

INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Not for submission under 37 CFR 1.99)	Application Number		
	Filing Date		
	First Named Inventor	Anurag Agarwal	
	Art Unit		
	Examiner Name		
	Attorney Docket Number	ThreatM008	

U.S.PATENTS						Remove
Examiner Initial*	Cite No	Patent Number	Kind Code ¹	Issue Date	Name of Patentee or Applicant of cited Document	Pages,Columns,Lines where Relevant Passages or Relevant Figures Appear
	1	7891003	B2	2011-02-15	Mir	
	2	7900259	B2	2011-03-01	Jeschke et al.	
	3	9774613	B2	2017-09-26	Thomas et al.	
	4	4959015		1990-09-25	Rasinski et al.	
	5	7433829	B2	2008-10-07	Borgia et al.	
	6	7096502	B1	2006-08-22	Fox et al.	
	7	8191139	B2	2012-05-29	Heimerdinger et al.	
	8	8255995	B2	2012-08-28	Kraemer et al.	

**INFORMATION DISCLOSURE
STATEMENT BY APPLICANT**
(Not for submission under 37 CFR 1.99)

Application Number		
Filing Date		
First Named Inventor	Anurag Agarwal	
Art Unit		
Examiner Name		
Attorney Docket Number	ThreatM008	

9	8413237	B2	2013-04-02	O'Rourke et al.	
10	9497203	B2	2016-11-15	Honig et al.	

If you wish to add additional U.S. Patent citation information please click the Add button.

Add

U.S.PATENT APPLICATION PUBLICATIONS

Remove

Examiner Initial*	Cite No	Publication Number	Kind Code ¹	Publication Date	Name of Patentee or Applicant of cited Document	Pages, Columns, Lines where Relevant Passages or Relevant Figures Appear
	1	20060241991	A1	2006-10-26	Pudhukottai et al.	
	2	20110178942	A1	2011-07-21	Watters et al.	
	3	20140137257	A1	2014-05-15	Martinez et al.	
	4	20110126111	A1	2011-05-26	Gill et al.	
	5	20180324207	A1	2018-11-08	Reybok, Jr. et al.	
	6	20060015941	A1	2006-01-19	McKenna	

If you wish to add additional U.S. Published Application citation information please click the Add button.

Add

FOREIGN PATENT DOCUMENTS

Remove

**INFORMATION DISCLOSURE
STATEMENT BY APPLICANT**
(Not for submission under 37 CFR 1.99)

Application Number		
Filing Date		
First Named Inventor	Anurag Agarwal	
Art Unit		
Examiner Name		
Attorney Docket Number	ThreatM008	

Examiner Initial*	Cite No	Foreign Document Number ³	Country Code ² i	Kind Code ⁴	Publication Date	Name of Patentee or Applicant of cited Document	Pages, Columns, Lines where Relevant Passages or Relevant Figures Appear	T ⁵
	1							

If you wish to add additional Foreign Patent Document citation information please click the Add button

NON-PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc), date, pages(s), volume-issue number(s), publisher, city and/or country where published.	T ⁵
	1	Threat Risk Modeling, published online at least as early as Feb. 3, 2018 by OWASP, available at https://www.owasp.org/index.php/Threat_Risk_Modeling .	
	2	Threat Model, Wikipedia, published online at least as early as Feb. 3, 2018, available at https://en.wikipedia.org/wiki/Threat_model (note that some of the material in this Wikipedia article appears to be posted by one or more inventors of the present application or its parent provisional applications - some of the material appears to be posted by Brian Beyst, for example, who is an inventor on two of the parent provisional applications).	
	3	Comparisons of Threat Model Modeling Methodologies, published online by ThreatModeler at least as early as Apr. 15, 2016, available at http://threatmodeler.com/comparison-threat-modeling-methodologies/ (note that this article appears to be affiliated with a business organization of, and may have been published under the direction of, one or more inventors of the present application or its parent provisional applications).	
	4	Microsoft Threat Modeling Tool, published online by Microsoft, different sections published at different times from Aug. 22, 2017 to Jan. 24, 2018, available online at https://docs.microsoft.com/en-us/azure/opbuildpdf/security/TOC.pdf?branch=live	
	5	A description of an on-sale version of systems and methods, on sale in the U.S. by applicants at least as early as 12/31/2011, which on-sale version disclosed some of the elements disclosed in the present application. The description includes a detailed discussion of which claim elements of the claims as originally filed in the present application were disclosed in the 2011 version and which were not, and includes screenshots of user interfaces used in the system/methods.	
	6	Lockheed Martin Corporation, "Seven Ways to Apply the Cyber Kill Chain with a Threat Intelligence Platform," published at least as early as 2015.	

If you wish to add additional non-patent literature document citation information please click the Add button

**INFORMATION DISCLOSURE
STATEMENT BY APPLICANT**
(Not for submission under 37 CFR 1.99)

Application Number	
Filing Date	
First Named Inventor	Anurag Agarwal
Art Unit	
Examiner Name	
Attorney Docket Number	ThreatM008

EXAMINER SIGNATURE

Examiner Signature		Date Considered	
--------------------	--	-----------------	--

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through a citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ See Kind Codes of USPTO Patent Documents at www.USPTO.GOV or MPEP 901.04. ² Enter office that issued the document, by the two-letter code (WIPO Standard ST.3). ³ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁴ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁵ Applicant is to place a check mark here if English language translation is attached.

**INFORMATION DISCLOSURE
STATEMENT BY APPLICANT**
(Not for submission under 37 CFR 1.99)

Application Number	
Filing Date	
First Named Inventor	Anurag Agarwal
Art Unit	
Examiner Name	
Attorney Docket Number	ThreatM008

CERTIFICATION STATEMENT

Please see 37 CFR 1.97 and 1.98 to make the appropriate selection(s):

That each item of information contained in the information disclosure statement was first cited in any communication from a foreign patent office in a counterpart foreign application not more than three months prior to the filing of the information disclosure statement. See 37 CFR 1.97(e)(1).

OR

☐ That no item of information contained in the information disclosure statement was cited in a communication from a foreign patent office in a counterpart foreign application, and, to the knowledge of the person signing the certification after making reasonable inquiry, no item of information contained in the information disclosure statement was known to any individual designated in 37 CFR 1.56(c) more than three months prior to the filing of the information disclosure statement. See 37 CFR 1.97(e)(2).

See attached certification statement.

The fee set forth in 37 CFR 1.17 (p) has been submitted herewith.

☒ A certification statement is not submitted herewith.

SIGNATURE

A signature of the applicant or representative is required in accordance with CFR 1.33, 10.18. Please see CFR 1.4(d) for the form of the signature.

Signature	/Paul B. Johnson/	Date (YYYY-MM-DD)	2018-12-06
Name/Print	Paul B. Johnson	Registration Number	59375

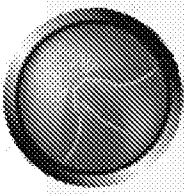
This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 1 hour to complete, including gathering, preparing and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these records.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.



- Home
- About OWASP
- Acknowledgements
- Advertising
- AppSec Events
- Books
- Brand Resources
- Chapters
- Donate to OWASP
- Downloads
- Funding
- Governance
- Initiatives
- Mailing Lists
- Membership
- Merchandise
- News
- Community portal
- Presentations
- Press
- Projects
- Video
- Volunteer
- Reference
- Activities
- Attacks
- Code Snippets
- Controls
- Glossary
- How To...
- Java Project
- .NET Project
- Principles

Threat Risk Modeling

- 1 Threat Risk Modeling
- 2 Performing threat risk modeling using the Microsoft Threat Modeling Process
 - 2.1 Identify Security Objectives
 - 2.2 Application Overview
 - 2.3 Decompose Application
 - 2.4 Identify Threats
 - 2.5 STRIDE
 - 2.6 DREAD
- 3 Alternative Threat Modeling Systems
 - 3.1 Trike
 - 3.2 AS/NZS 4360:2004 Risk Management
 - 3.3 CVSS
 - 3.4 OCTAVE
- 4 ThreatModel SDK
- 5 Conclusion
- 6 Further Reading
- 7 Appendix: Alternative open-source Risk Management tools
- 8 Reference

When you start a web application design, it is essential to apply threat modeling; otherwise you will squander resources, time, and money on useless controls that fail to focus on the real threats. There are multiple approaches to threat modeling, as listed below:

- Software centric threat modeling
- Security centric threat modeling
- Asset or risk centric threat modeling. Below represents a mixture of Threat Modeling tools and industry references. The method used to assess risk is not nearly as important as actually performing a structured threat risk modeling. Microsoft notes that the single most important factor in their security improvement program was the corporate adoption of threat risk modeling. One of many considerations is Microsoft's threat modeling process. It is simple to adopt by designers, developers, code reviewers, and the quality assurance team. The following sections provide some overview information (or see Section 6.9,

Further Reading, for additional resources).

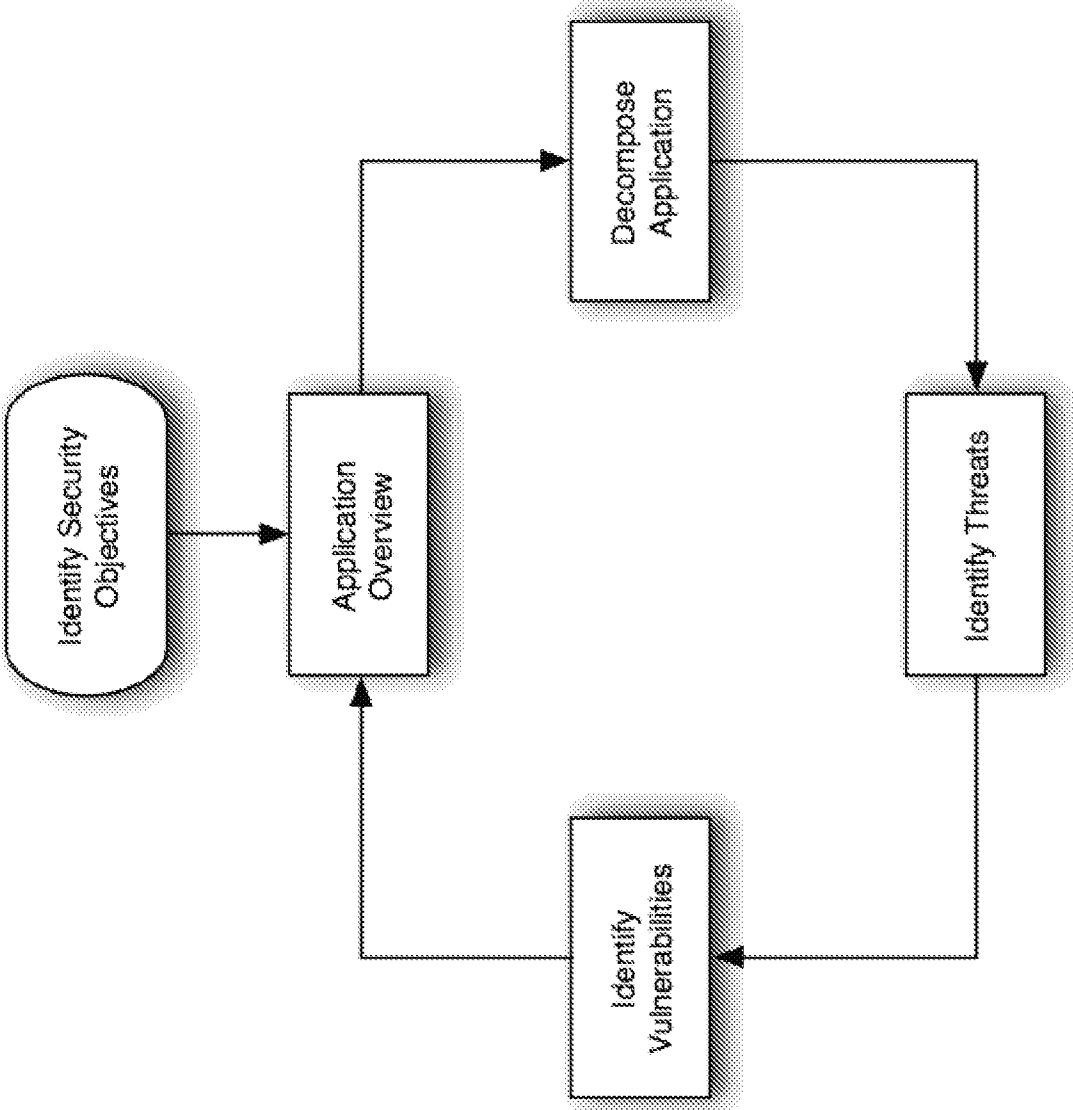
Threat Risk Modeling

Threat risk modeling is an essential process for secure web application development. It allows organizations to determine the correct controls and to produce effective countermeasures within budget. For example, there is little point in spending \$100,000 for fraud control for a system that has negligible fraud risk.

Performing threat risk modeling using the Microsoft Threat Modeling Process

The threat risk modeling process has five steps, enumerated below and shown graphically in Figure 1. They are:

1. Identify Security Objectives
2. Survey the Application
3. Decompose it
4. Identify Threats
5. Identify Vulnerabilities



Let's consider the steps in more detail.

Identify Security Objectives

The business (or project management) leadership, in concert with the software development and quality assurance teams, all need to understand the security objectives. To facilitate this, start by breaking down the application's security objectives into the following categories:

- **Identity:** Does the application protect user identity from abuse? Are there adequate controls in place to ensure evidence of identity (as required for many banking applications?)

- **Financial:** Assess the level of risk the organization is prepared to absorb in remediation, as a potential financial loss. For example, forum software may have a lower estimated financial risk than an Internet banking application.
- **Reputation:** Quantify or estimate of the loss of reputation derived from the application being misused or successfully attacked.
- **Privacy and Regulatory:** To what extent will the application have to protect user data? Forum software by its nature is public, but a tax preparation application is subject to tax regulations and privacy legislation requirements in most countries.
- **Availability Guarantees:** Is the application required to be available per a **Service Level Agreement (SLA)** or similar guarantee? Is it a nationally protected infrastructure? To what level will the application have to be available? High availability techniques are significantly more expensive, so applying the correct controls up front will save a great deal of time, resources, and money.

This is by no means an exhaustive list, but it gives an idea of some of the business risk decisions leading into selecting and building security controls.

Other sources of risk guidance come from:

- Laws (such as privacy or finance laws)
- Regulations (such as banking or e-commerce regulations)
- Standards (such as ISO/IEC 27001)
- Legal Agreements (such as payment card industry standards or merchant agreements)
- Corporate Information Security Policy

Application Overview

Once the security objectives have been defined, analyze the application design to identify the **components**, **data flows**, and **trust boundaries**.

Do this by surveying the application's architecture and design documentation. In particular, look for UML component diagrams. Such high level component diagrams are generally sufficient to understand how and why data flows to various places. For example, data movement across a trust boundary (such as from the Internet to the web tier, or from the business logic to the database server), needs to be carefully analyzed, whereas data that flows within the same trust level does not need as much scrutiny.

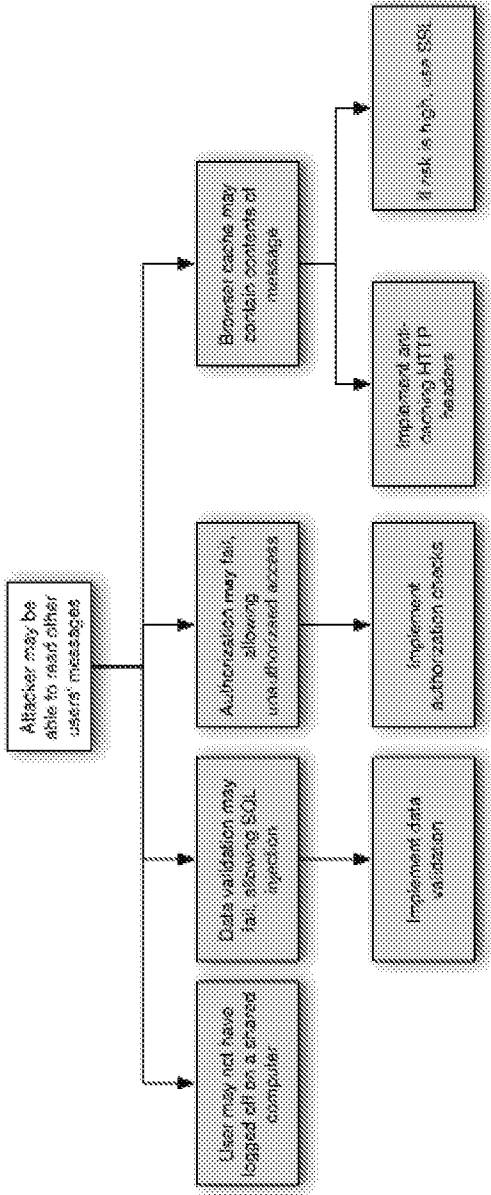
Decompose Application

Once the application architecture is understood then decompose it further, to identify the features and modules with a security impact that need to be evaluated. For example, when investigating the authentication module, it is necessary to understand how data enters the module, how the module validates and processes the data, where the data flows, how the data is stored, and what fundamental decisions and assumptions are made by the module.

Identify Threats

It is impossible to write down unknown threats, but it is likewise unlikely that new malware will be created to exploit new vulnerabilities within custom systems. Therefore, concentrate on known risks, which can be easily demonstrated using tools or techniques from Bugtraq.

Microsoft suggests two different approaches for writing up threats. One is a threat graph, as shown in Figure 2, and the other is a structured list.



Typically, a threat graph imparts more information quickly but it takes longer to construct, while a structured list is easier to create but it will take longer for the threat impacts to become obvious.

1. Attacker may be able to read other user's messages
2. User may not have logged off on a shared PC
3. Data validation may allow SQL injection
4. Implement data validation
5. Authorization may fail, allowing unauthorized access
6. Implement authorization checks
7. Browser cache may contain contents of message
8. Implement anti-caching directive in HTTP headers
9. If eavesdropping risk is high, use SSL

Note that it takes a motivated attacker to exploit a threat; they generally want something from your application or to obviate controls. To understand the relevant threats, use the following categories to understand who might attack the application:

- **Accidental Discovery:** An ordinary user stumbles across a functional mistake in your application, just using a web browser, and gains access to privileged information or functionality.
- **Automated Malware:** Programs or scripts, which are searching for known vulnerabilities, and then report them back to a central collection site.
- **The Curious Attacker:** a security researcher or ordinary user, who notices something wrong with the application, and

decides to pursue further.

- **Script Kiddies:** Common renegades, seeking to compromise or deface applications for collateral gain, notoriety, or a political agenda, perhaps using the attack categories described in the *OWASP Web Application Penetration Checklist*.
- **The Motivated Attacker:** Potentially, a disgruntled staff member with inside knowledge or a paid professional attacker.
- **Organized Crime:** Criminals seeking high stake payouts, such as cracking e-commerce or corporate banking applications, for financial gain.

It is vital to understand the level of attacker you are defending against. For example, a motivated attacker, who understands your internal processes is often more dangerous than script kiddies.

STRIDE

STRIDE is a classification scheme for characterizing known threats according to the kinds of exploit that are used (or motivation of the attacker). The STRIDE acronym is formed from the first letter of each of the following categories.

Spoofing Identity "Identity spoofing" is a key risk for applications that have many users but provide a single execution context at the application and database level. In particular, users should not be able to become any other user or assume the attributes of another user.

Tampering with Data Users can potentially change data delivered to them, return it, and thereby potentially manipulate client-side validation, GET and POST results, cookies, HTTP headers, and so forth. The application should not send data to the user, such as interest rates or periods, which are obtainable only from within the application itself. The application should also carefully check data received from the user and validate that it is sane and applicable before storing or using it.

Repudiation Users may dispute transactions if there is insufficient auditing or recordkeeping of their activity. For example, if a user says, "But I didn't transfer any money to this external account", and you cannot track his/her activities through the application, then it is extremely likely that the transaction will have to be written off as a loss.

Therefore, consider if the application requires non-repudiation controls, such as web access logs, audit trails at each tier, or the same user context from top to bottom. Preferably, the application should run with the user's privileges, not more, but this may not be possible with many off-the-shelf application frameworks.

Information Disclosure Users are rightfully wary of submitting private details to a system. If it is possible for an attacker to publicly reveal user data at large, whether anonymously or as an authorized user, there will be an immediate loss of confidence and a substantial period of reputation loss. Therefore, applications must include strong controls to prevent user ID tampering and abuse, particularly if they use a single context to run the entire application.

Also, consider if the user's web browser may leak information. Some web browsers may ignore the no caching directives in HTTP headers or handle them incorrectly. In a corresponding fashion, every secure application has a responsibility to minimize the amount of information stored by the web browser, just in case it leaks or leaves information behind, which can be used by an attacker to learn details about the application, the user, or to potentially become that user.

Finally, in implementing persistent values, keep in mind that the use of hidden fields is insecure by nature. Such storage should not be relied on to secure sensitive information or to provide adequate personal privacy safeguards.

Denial of Service Application designers should be aware that their applications may be subject to a denial of service attack. Therefore, the use of expensive resources such as large files, complex calculations, heavy-duty searches, or long queries should be reserved for authenticated and authorized users, and not available to anonymous users.

For applications that do not have this luxury, every facet of the application should be engineered to perform as little work as possible, to use fast and few database queries, to avoid exposing large files or unique links per user, in order to prevent simple denial of service attacks.

Elevation of Privilege If an application provides distinct user and administrative roles, then it is vital to ensure that the user cannot elevate his/her role to a higher privilege one. In particular, simply not displaying privileged role links is insufficient. Instead, all actions should be gated through an authorization matrix, to ensure that only the permitted roles can access privileged functionality.

DREAD

DREAD is a classification scheme for quantifying, comparing and prioritizing the amount of risk presented by each evaluated threat. The DREAD acronym is formed from the first letter of each category below.

DREAD modeling influences the thinking behind setting the risk rating, and is also used directly to sort the risks. The DREAD algorithm, shown below, is used to compute a risk value, which is an average of all five categories.

Risk_DREAD = (DAMAGE + REPRODUCIBILITY + EXPLOITABILITY + AFFECTED USERS + DISCOVERABILITY) / 5

The calculation always produces a number between 0 and 10; the higher the number, the more serious the risk.

Here are some examples of how to quantify the DREAD categories.

Damage Potential

- If a threat exploit occurs, how much damage will be caused?
 - 0 = Nothing
 - 5 = Individual user data is compromised or affected.
 - 10 = Complete system or data destruction

Reproducibility

- How easy is it to reproduce the threat exploit?
 - 0 = Very hard or impossible, even for administrators of the application.
 - 5 = One or two steps required, may need to be an authorized user.
 - 10 = Just a web browser and the address bar is sufficient, without authentication.

Exploitability

- What is needed to exploit this threat?
 - 0 = Advanced programming and networking knowledge, with custom or advanced attack tools.

- 5 = Malware exists on the Internet, or an exploit is easily performed, using available attack tools.
- 10 = Just a web browser

Affected Users

- How many users will be affected?
 - 0 = None
 - 5 = Some users, but not all
 - 10 = All users

Discoverability

- How easy is it to discover this threat?
 - 0 = Very hard to impossible; requires source code or administrative access.
 - 5 = Can figure it out by guessing or by monitoring network traces.
 - 9 = Details of faults like this are already in the public domain and can be easily discovered using a search engine.
 - 10 = The information is visible in the web browser address bar or in a form.

Note: When performing a security review of an existing application, "Discoverability" will often be set to 10 by convention, as it is assumed the threat issues will be discovered.

Note: Using DREAD can be difficult at first. It may be helpful to think of Damage Potential and Affected Users in terms of Impact, while thinking of Reproducibility, Exploitability, and Discoverability in terms of Probability. Using the Impact vs Probability approach (which follows best practices such as defined in NIST-800-30), I would alter the formula to make the Impact score equal to the Probability score. Otherwise the probability scores have more weight in the total.

Alternative Threat Modeling Systems

OWASP recognizes that the adoption of the Microsoft modeling process may not fit all organizations. If STRIDE and DREAD are unacceptable for some reason, we recommend that your organization "dry run" the other threat risk models discussed against an existing application or design. This will allow you to determine which approach works best for you, and to adopt the most appropriate threat modeling tools for your organization.

In summary, performing threat modeling provides a far greater return than most any other control in this Guide. Therefore, make threat risk modeling an early priority in your application design process.

Trike

Trike is a threat modeling framework with similarities to the Microsoft threat modeling processes. However, Trike differs because it uses a risk based approach with distinct implementation, threat, and risk models, instead of using the STRIDE/DREAD aggregated threat model (attacks, threats, and weaknesses). From the Trike paper, Trike's goals are:

- With assistance from the system stakeholders, to ensure that the risk this system entails to each asset is acceptable to all stakeholders.

- Be able to tell whether we have done this.
- Communicate what we've done and its effects to the stakeholders.
- Empower stakeholders to understand and reduce the risks to them and other stakeholders implied by their actions within their domains.

For more information on Trike, please see Section 6.9, reference 8.

AS/NZS 4360:2004 Risk Management

The Australian/New Zealand Standard AS/NZS 4360, first issued in 1999, and revised in 2004, is the world's first formal standard for documenting and managing risk and is still one of the few formal standards for managing it. The standard's approach is simple (it's only 28 pages long), flexible, and iterative. Furthermore, it does not lock organizations into a particular risk management methodology, provided the methodology fulfils the AS/NZS 4360 five steps. It also provides several sets of risk tables as examples, and allows organizations to freely develop and adopt their own.

The five steps of the AS/NZS 4360 process are:

- **Establish Context:** Establish the risk domain, i.e., which assets/systems are important?
- **Identify the Risks:** Within the risk domain, what specific risks are apparent?
- **Analyze the Risks:** Look at the risks and determine if there are any supporting controls in place.
- **Evaluate the Risks:** Determine the residual risk.
- **Treat the Risks:** Describe the method to treat the risks so that risks selected by the business will be mitigated.

AS/NZS 4360 assumes that risk will be managed by an *operational risk group*, and that the organization has adequate skills and risk management resources in house to identify, analyze, and treat the risks.

The advantages of AS/NZS 4360:

- AS/NZS 4360 works well as a risk management methodology for organizations requiring Sarbanes-Oxley compliance.
- AS/NZS 4360 works well for organizations that prefer to manage risks in a traditional way, such as just using likelihood and consequence to determine an overall risk.
- AS/NZS 4360 is familiar to most risk managers worldwide, and your organization may already have implemented an AS/NZS 4360 compatible approach.
- You are an Australian organization, and may be required to use it if you are audited on a regular basis, or to justify why you aren't using it. Luckily, the STRIDE/DREAD model discussed earlier is AS/NZS 4360 compatible.

The limitations of AS/NZS 4360:

- The AS/NZS 4360 approach works best for business or systemic risks than for technical risks.
- AS/NZS 4360 does not define the methodology to perform a structured threat risk modeling exercise.
- As AS/NZS 4360 is a generic framework for managing risk, it does not provide any structured method to enumerate web application security risks.

Although AS/NZS 4360 may be used to rank risks for security reviews, the lack of structured methods of enumerating threats

for web applications makes it less desirable than other methodologies described earlier.

CVSS

The US Department of Homeland Security (DHS) established the NIAC Vulnerability Disclosure Working Group, which incorporates input from Cisco Systems, Symantec, ISS, Qualys, Microsoft, CERT/CC, and eBay. One of the group's outputs is the **Common Vulnerability Scoring System (CVSS)**.

The advantages of CVSS:

- You have just received notification from a security researcher or other source that your product has vulnerability, and you wish to ensure that it has an accurate and normalized severity rating, so as to alert your customers to the appropriate level of action required when you release the patch.
- You are a security researcher, and have found several threat exploits within an application. You would like to use the CVSS ranking system to produce reliable risk rankings, to ensure that the ISV will take the exploits seriously as indicated by their rating.
- CVSS has been recommended by the working group for use by US Government departments. However, it is unclear if it will become policy or be widely adopted at the time of this writing.

The limitations of CVSS:

- CVSS does not find or reduce the attack surface area (i.e. design flaws), or help enumerate risks within any arbitrary piece of code, as it is just a scoring system, not a modeling methodology.
- CVSS is more complex than STRIDE/DREAD, as it aims to calculate the risk of announced vulnerabilities as applied to deployed software and environmental factors.
- The CVSS risk ranking is complex -- a spreadsheet is required to calculate the risk components as the assumption behind CVSS is that a specific vulnerability has been identified and announced, or a worm or Trojan has been released targeting a small number of attack vectors.
- The overhead of calculating the CVSS risk ranking is quite high if applied to a thorough code review, which may have 250 or more threats to rank.

OCTAVE

OCTAVE is a heavyweight risk methodology approach originating from Carnegie Mellon University's Software Engineering Institute (SEI) in collaboration with CERT. OCTAVE focuses on organizational risk, not technical risk. OCTAVE comes in two versions: Full OCTAVE, for large organizations, and OCTAVE-S for small organizations, both of which have specific catalogs of practices, profiles, and worksheets to document the modeling outcomes.

OCTAVE is popular with many sites and is useful when:

- Implementing an organizational culture of risk management and controls becomes necessary.
- Documenting and measuring business risk becomes timely.
- Documenting and measuring the overall IT security risk, particularly as it relates to the corporate IT risk management,

becomes necessary.

- When documenting risks surrounding complete systems becomes necessary.
- To accommodate a fundamental reorganization, such as when an organization does not have a working risk methodology in place, and requires a robust risk management framework to be put in place.

The limitations of OCTAVE are:

- OCTAVE is incompatible with AS/NZS 4360, as it mandates Likelihood = 1 (i.e., It assumes a threat will always occur) and this is inappropriate for many organizations. OCTAVE-S makes the inclusion of this probability optional, but this is not part of the more comprehensive OCTAVE standard.
- Consisting of 18 volumes, OCTAVE is large and complex, with many worksheets and practices to implement.
- It does not provide a list of “out of the box” practices for assessing and mitigating web application security risks.

Because of these issues, OWASP does not anticipate that OCTAVE will be used at large by application designers or developers, because it fails to take threat risk modeling into consideration, which is useful during all stages of development, by all participants, to reduce the overall risk of an application becoming vulnerable to attack.

ThreatModel SDK

The ThreatModel SDK is a minimalistic Java library that provides a basic vendor-neutral object model along with the ability to parse reports generated from common threat modeling tools. Supported Threat Modeling Tools:

- Microsoft Threat Modeling Tool 2016

Planned Threat Modeling Tools:

- Mozilla SeaSponge

For more information visit: <https://github.com/stevspringett/threatmodel-sdk>

Conclusion

In this chapter, we have touched on the basic principles of threat risk modeling, risk management, and web application security. Applications that leverage the underlying intent of these principles will be more secure than their counterparts, which will only be minimally compliant just by including specific controls.

Further Reading

- Threat Analysis & Modeling v2.1.2, © Microsoft Corporation, 2007.
- Threat Modeling Web Applications, J.D. Meier, Alex Mackman, Blaine Wastell, © Microsoft Corporation, May 2005.
- Improving Web Application Security: Threats and Countermeasures, J.D. Meier, Alex Mackman, Michael Dunner, Srinath Vasireddy, Ray Escamilla and Anandha Murukan, © Microsoft Corporation, June 2003.
- Threat Modeling, Frank Swiderski and Window Snyder, Microsoft Press, June 2004, ISBN 0-7356-1991-3.

- Writing Secure Code, 2nd Edition, Howard and LeBlanc, (pp. 69 – 124), Microsoft Press, 2003, ISBN 0-7356-1722-8.
- The STRIDE Threat Model, © Microsoft Corporation, 2005.
- DREADful - the DREAD system, © Microsoft Corporation, 2005.
- A Conceptual Model for Threat Modeling Applications, Saitta, Larcom, and Michael Eddington, July 2005, <http://dymaxion.org/trike/>.
- AS/NZS 4360:2004 Risk Management, Standards Australia and Standards New Zealand.
- CVSS, U.S. Department of Homeland Security library, February 2005.
- OCTAVE, CERT library.

Appendix: Alternative open-source Risk Management tools

- OSMR
- MARCO
- CORAS Risk Assessment Platform
- ISO 17799 Risk Assessment Toolkit
- Easy Threat Risk Assessment
- ARMS
- Minaccia
- ThreatMind
- Open Source Requirements Management Tool

Reference

Development Guide Table of Contents

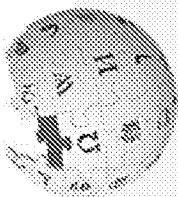
Categories: [FIXME](#) | [OWASP Guide Project](#) | [Activity](#) | [Externally Linked Page](#) | [Threat Modeling](#) | [SAMM-TA-1](#)

This page was last modified on 13 July 2017, at 04:20.

Content is available under Creative Commons Attribution-ShareAlike unless otherwise noted.

[Privacy policy](#) | [About OWASP](#) | [Disclaimers](#)





WIKIPEDIA
The Free Encyclopedia

- Main page
- Contents
- Featured content
- Current events
- Random article
- Donate to Wikipedia
- Wikipedia store

- Interaction
- Help
- About Wikipedia
- Community portal
- Recent changes
- Contact page

- Tools
- What links here
- Related changes
- Upload file
- Special pages
- Permanent link
- Page information
- Wikidata item
- Cite this page

- Print/export
- Create a book
- Download as PDF
- Printable version

Languages

Français
中文

ArticleTalk

Threat model

From Wikipedia, the free encyclopedia

Threat modeling is a process by which potential threats, such as structural vulnerabilities can be identified, enumerated, and prioritized -- all from a hypothetical attacker's point of view. The purpose of threat modeling is to provide defenders with a systematic analysis of the probable attacker's profile, the most likely attack vectors, and the assets most desired by an attacker. Threat modeling answers the questions "Where are the high-value assets?" "Where am I most vulnerable to attack?" "What are the most relevant threats?" "Is there an attack vector that might go unnoticed?"

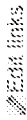
Conceptually, most people incorporate some form of threat modeling in their daily life and don't even realize it. Commuters use threat modeling to consider what might go wrong during the morning drive to work and to take preemptive action to avoid possible accidents. Children engage in threat modeling when determining the best path toward an intended goal while avoiding the playground bully. In a more formal sense, threat modeling has been used to prioritize military defensive preparations since antiquity.

Contents

- 1 Evolution of IT-Based Threat Modeling
- 2 Threat Modeling Methodologies for IT Purposes
 - 2.1 STRIDE Methodology
 - 2.2 P.A.S.T.A.
 - 2.3 Trike
 - 2.4 VAST
- 3 Generally Accepted IT Threat Modeling Processes
 - 3.1 Visual Representations based on Data Flow Diagrams
 - 3.2 Visual Representations based on Process Flow Diagrams
- 4 Threat Modeling Tools
- 5 Further fields of application
- 6 References

Evolution of IT-Based Threat Modeling [edit]

Shortly after shared computing made its debut in the early 1960s individuals began seeking ways to exploit security vulnerabilities for personal gain.^[1] As a result, engineers and computer scientists soon began developing threat modeling



concepts for information technology systems.

Early IT-based threat modeling methodologies were based on the concept of architectural patterns^[2] first presented by Christopher Alexander in 1977. In 1988 Robert Barnard developed and successfully applied the first profile for an IT-system attacker.

In 1994, Edward Amoroso put forth the concept of a "threat tree" in his book, "Fundamentals of Computer Security Technology."^[3] The concept of a threat tree was based on decision tree diagrams. Threat trees graphically represent how a potential threat to an IT system can be exploited.

Independently, similar work was conducted by the NSA and DARPA on a structured graphical representation of how specific attacks against IT-systems could be executed. The resulting representation were called "attack trees." In 1998 Bruce Schneier published his analysis of cyber risks utilizing attack trees in his paper entitled "Toward a Secure System Engineering Methodology."^[4] The paper proved to be a seminal contribution in the evolution of threat modeling for IT-systems. In Schneier's analysis, the attacker's goal is represented as a "root node," with the potential means of reaching the goal represented as "leaf nodes." Utilizing the attack tree in this way allowed cybersecurity professionals to systematically consider multiple attack vectors against any defined target.

In 1999, Microsoft cybersecurity professionals Loren Kohnfelder and Praerit Gard applied Schneier's attack tree analysis to develop a methodical means to consider potential attacks relevant to the Microsoft Windows development environment. The resultant STRIDE^[5] threat model (STRIDE is an acronym for Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service, Elevation of privilege) required security professionals to systematically determine how a potential attacker could utilize any threat categorized by the STRIDE acronym at each node of a Schneier-type attack tree.

In 2003, OCTAVE^[6] (Operationally Critical Threat, Asset, and Vulnerability Evaluation) method, a risk-based assessment threat modeling methodology, was first introduced with a primary focus on organizational risk management.

In 2004, Frank Swiderski and Window Snyder wrote "Threat Modeling," by Microsoft press. In it they developed the concept of using threat models to create secure applications.

In 2014 Ryan Stillions expressed the idea that cyber threats should be expressed with different semantic levels, and proposed the DML (Detection Maturity Level) model.^[7] An attack is an instantiation of a threat scenario which is caused by a specific attacker with a specific goal in mind and a strategy for reaching that goal. The goal and strategy represent the highest semantic levels of the DML model. This is followed by the TTP (Tactics, Techniques and Procedures) which represent intermediate semantic levels. The lowest semantic levels of the DML model are the tools used by the attacker, host and observed network artefacts such as packets and payloads, and finally atomic indicators such as IP addresses at the lowest semantic level. Current SIEM tools typically only provide indicators at the lowest semantic levels. There is therefore a need to develop SIEM tools that can provide threat indicators at higher semantic levels.^[8]

Threat Modeling Methodologies for IT Purposes [edit]

Conceptually a threat modeling practice flows from a methodology. Numerous threat modeling methodologies are available for

implementation. Based on volume of published online content, the four methodologies discussed below are the most well known.

STRIDE Methodology [edit]

The STRIDE approach to threat modeling was introduced in 1999 at Microsoft, providing a mnemonic for developers to find 'threats to our products' ^[9]. STRIDE, Patterns and Practices, and Asset/entry point were amongst the threat modeling approaches developed and published by Microsoft. References to "the" Microsoft methodology commonly mean STRIDE.

P.A.S.T.A. [edit]

The Process for Attack Simulation and Threat Analysis (PASTA) is a seven-step, risk-centric methodology.^[10] It provides a seven-step process for aligning business objectives and technical requirements, taking into account compliance issues and business analysis. The intent of the method is to provide a dynamic threat identification, enumeration, and scoring process. Once the threat model is completed security subject matter experts develop a detailed analysis of the identified threats. Finally, appropriate security controls can be enumerated. This methodology is intended to provide an attacker-centric view of the application and infrastructure from which defenders can develop an asset-centric mitigation strategy.

Trike [edit]

The focus of the Trike methodology^[11] is using threat models as a risk-management tool. Within this framework, threat models are used to satisfy the security auditing process. Threat models are based on a "requirements model." The requirements model establishes the stakeholder-defined "acceptable" level of risk assigned to each asset class. Analysis of the requirements model yields a threat model form which threats are enumerated and assigned risk values. The completed threat model is used to construct a risk model based on asset, roles, actions, and calculated risk exposure.

VAST [edit]

VAST is an acronym for Visual, Agile, and Simple Threat modeling.^[12] The underlying principle of this methodology is the necessity of scaling the threat modeling process across the infrastructure and entire SDLC, and integrating it seamlessly into an Agile software development methodology. The methodology seeks to provide actionable outputs for the unique needs of various stakeholders: application architects and developers, cybersecurity personnel, and senior executives. The methodology provides a unique application and infrastructure visualization scheme such that the creation and use of threat models do not require specific security subject matter expertise.

Generally Accepted IT Threat Modeling Processes [edit]

All IT-related threat modeling processes start with creating a visual representation of the application and / or infrastructure being analyzed. The application / infrastructure is decomposed into various elements to aid in the analysis. Once completed, the visual representation is used to identify and enumerate potential threats. Further analysis of the model regarding risks associated with identified threats, prioritization of threats, and enumeration of the appropriate mitigating controls depends on

the methodological basis for the threat model process being utilized.

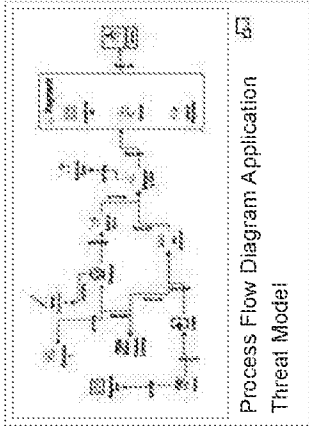
Visual Representations based on Data Flow Diagrams [edit]

The Microsoft methodology, PASTA, and Trike each develop a visual representation of the application-infrastructure utilizing data flow diagrams (DFD).^[13] DFDs were developed in the 1970s as tool for system engineers to communicate, on a high level, how an application caused data to flow, be stored, and manipulated by the infrastructure upon which the application runs. Traditionally, DFDs utilize only four unique symbols: data flows, data stores, processes, and interactors. In the early 2000s, an additional symbol, trust boundaries, were added to allow DFDs to be utilized for threat modeling.

Once the application-infrastructure system is decomposed into its five elements, security experts consider each identified threat entry point against all known threat categories. Once the potential threats are identified, mitigating security controls can be enumerated or additional analysis can be performed.

Visual Representations based on Process Flow Diagrams [edit]

The VAST methodology distinguished between application threat models (ATM) and operational or infrastructure threat models (OTM).^[14]

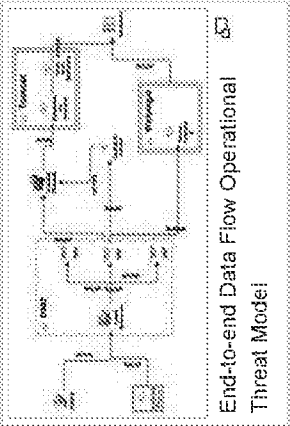


ATMs are built with process flow diagrams (PFD).^[13] PFDs were developed in 2011 as a tool to allow Agile software development teams to create threat models based on the application design process. Applications are decomposed into their various features or use cases. Each feature is described in terms of the widgets or code building blocks required to build that feature. Features are then connected by communication protocols. The resulting visualization is a map of how a user moves through the various features of an application.

OTMs are built with end-to-end data flow diagrams very similar to traditional DFDs.

End to end data flow diagrams decompose an IT system into its various independent, grouped, and shared components. Each component is described in terms of specific attributes. Components are then connected by communication pathways and protocols.

Once the ATM or OTM is constructed, the methodology specifies how potential threats are identified, enumerated, prioritized, and associated with their relevant risks and mitigating security controls.



[edit]

Threat Modeling Tools

There are currently five tools available for organizational threat modeling:

- Microsoft's free threat modeling tool – the Threat Modeling Tool (formerly SDL Threat Modeling Tool).^[15] This tool also utilizes the Microsoft threat modeling methodology, is DFD-based, and identifies threats based on the STRIDE threat classification scheme. It is intended primarily for general use.
- MyAppSecurity offers the first commercially available threat modeling tool - ThreatModeler^[16] It utilizes the VAST methodology, is PFD-based, and identifies threats based on a customizable comprehensive threat library.^[17] It is intended for collaborative use across all organizational stakeholders.
- IriusRisk offers both a community and a commercial version of the tool. This tool focus on the creation and maintenance of a live Threat Model through the entire SDLC. It drives the process by using fully customizable questionnaires and Risk Pattern Libraries, and connects with other several different tools (OWASP ZAP, BDD-Security, Threadfix...) to empower automation.^[18]
- securiCAD is a threat modelling and risk management tool by the Scandinavian company foreseei. It is intended for company cyber security management, from CISO, to security engineer, to technician. securiCAD conducts automated attack simulations to current and future IT architectures, identifies and quantifies risks holistically including structural vulnerabilities, and provides decision support based on the findings. securiCAD is offered in both commercial and community editions.^[19]
- SD Elements by Security Compass is a software security requirements management platform that includes automated threat modeling capabilities. A set of threats is generated by completing a short questionnaire about the technical details and compliance drivers of the application. Countermeasures are included in the form of actionable tasks for developers that can be tracked and managed throughout the entire SDLC.^[20]

Further fields of application [edit]

Threat modeling is being applied not only to IT but also to other areas such as building and home automation.^[21] In this context, threats to security and privacy like information about the inhabitant's movement profiles, working times, and health situations are modeled as well as physical or network-based attacks. The latter could make use of more and more available smart building features, i.e., sensors (e.g., to spy on the inhabitant) and actuators (e.g., to unlock doors).^[21]

References [edit]

- ↑ McMillan, Robert (2012). "The World's First Computer Password? It Was Useless Too". *Wired Business*.
- ↑ Shostack, Adam (2014). "Threat Modeling: Designing for Security". John Wiley & Sons Inc: Indianapolis.
- ↑ Amoroso, Edward G (1994). "Fundamentals of Computer Security Technology". AT&T Bell Labs. Prentice-Hall: Upper Saddle River.
- ↑ Agarwal, Anurag "Archie," etal. *VAST Methodology: Visual, Agile, and Simple Threat Modeling*. Various interviews. Transformational Opportunities: Prescott Valley. 2016
- ↑ ^a ^b Agarwal, Anurag "Archie" (2016). "Threat Modeling – Data Flow Diagram vs Process Flow Diagram". *ThreatModeler.com*.

4. [^] [^] Schneier, Bruce; et al. (1998). "Toward A Secure System Engineering Methodology" (PDF). National Security Agency: Washington.

5. [^] "The STRIDE Threat Mode" . Microsoft. 2016.

6. [^] Alberts, Christopher (2003). "Introduction to the OCTAVE® Approach" (PDF). Software Engineering Institute, Carnegie Mellon: Pitsburg.

7. [^] Stillions, Ryan (2014). "The DMIL Model" . Ryan Stillions.

8. [^] Bromander, Siri (2016). "Semantic Cyberthreat Modelling" (PDF). Semantic Technology for Intelligence, Defence and Security (STIDS 2016).

9. [^] Kohnfelder, Loren; Gaig, Praert. "Threats to Our Products" . Microsoft. Retrieved 20 September 2016.

10. [^] Ucedavélez, Tony and Marco M. Morana (2015). "Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis" . Hobekin.

11. [^] Eddington, Michael, Brenda Larcom, and Eleanor Saitta (2005). "Trike v1 Methodology Document" . Octotrike.org.

14. [^] Agarwal, Anurag "Archie" (2016). "Application Threat Modeling vs Operational Threat Modeling" . ThreatModeler.com.

15. [^] "What's New with Microsoft Threat Modeling Tool 2016" . Microsoft. 2015.

16. [^] "ThreatModeler Home" . ThreatModeler.

17. [^] Agarwal, Anurag "Archie," etal. *Comprehensive Threat Library*. Various Interviews. Transformational Opportunities: Prescott Valley. 2016

18. [^] "Irius Risk Risk Management Tool" . Continuum Security. 2016.

19. [^] "Cyber Threat Modelling and Risk Management - securiCAD by foreseeit" . foreseeit.

20. [^] "SD Elements by Security Compass" . Retrieved 2017-03-24.

21. [^] ^a ^b Meyer, D.; Haase, J.; Eckert, M.; Klauer, B. (2016-07-01). "A threat-model for building and home automation" . 2016 *IEEE 14th International Conference on Industrial Informatics (INDIN)*: 860–866. doi:10.1109/INDIN.2016.7819280 .

Categories: Computer security | Computer security exploits

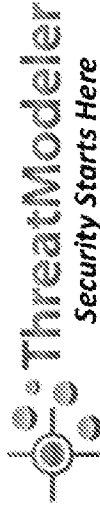
This page was last edited on 10 January 2018, at 09:09.

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, inc., a non-profit organization.

Privacy policy About Wikipedia Disclaimers Contact Wikipedia Developers Cookie statement Mobile view

IriusRisk
Ex. 1002 - Page 24

https://en.wikipedia.org/wiki/Threat_model[2/3/2018 7:49:50 PM]



Comparison of Threat Modeling Methodologies

Posted on *April 15, 2016* by *Reef DSouza*

Microsoft Threat Modeling Methodology: Stride Threat Model

Microsoft's threat modeling practice aligns with their Trustworthy Computing directive of January 2002 and its primary focus is to help ensure secure software begins during the design phase. In order for an application to meet the security properties of Confidentiality, Integrity, and Availability (CIA), along with Authorization, Authentication and Non-Repudiation, Microsoft uses its "STRIDE" methodology. STRIDE is an acronym that defines a threat classification system that stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. Stride threat model designed by Microsoft.

Data Flow Diagrams (DFD) are used to graphically represent the system and the DFDs use a standard set of symbols consisting of four elements: data flows, data stores, processes, and interactors, and to incorporate threat modeling, trust boundaries are also included. An application is decomposed into individual components and the relevant threats from the STRIDE system can be applied to each component. The mitigating controls for each component are then determined.

From a methodology point of view, scalability with a software development process such as AGILE is a challenge. And, since the Trustworthy Computing directives apply specifically to a Microsoft Windows development environment, it is limited in scope, since it does not address the many other development platforms in use by organizations. Also worth noting is that practitioners of this process require subject matter expertise in order to fully understand the threats, threat classifications, and relevance of security properties to the elements of a threat model. Compare Stride Threat Model with P.A.S.T.A. and Trike below.

P.A.S.T.A (Process for Attack Simulation and Threat Analysis):

This is a new application threat modeling methodology introduced by Marco Morana and Tony "UV." P.A.S.T.A is a seven-step process that is applicable to most application development methodologies and is platform agnostic. It not only aligns business objectives with technical requirements, but also takes into account compliance requirements, business impact analysis, and a dynamic approach to threat management, enumeration, and scoring.

The process begins with a clear definition of business objectives, security and compliance requirements, along with business impact analysis. Similar to the Microsoft process, an application is decomposed into components with use case diagrams and DFDs that are used to illustrate the threat model, from which threat and vulnerability analysis can be performed.

The next step involves the use of threat trees, abuse cases, scoring systems, and enumerations for more detailed analysis. The user can then view a threat model from an attacker's perspective by performing attack modeling in attack trees, along with attack surface analysis. In the final step, risk and business impact can be qualified and quantified, in conjunction with the necessary countermeasures.

This process combines the best of various threat modeling approaches using attack trees to provide an attacker-centric view of threats, in combination with risk and impact analysis. This helps organizations establish an asset-centric approach to planning its mitigation strategy. The use of threat trees to map threats to existing vulnerabilities lends itself to a much easier and scalable threat management process.

Beyond the technical aspect of this approach, the risk and business impact analysis elevates threat modeling from a "software development only" exercise by linking vulnerability management to business relevancy, since it involves the participation of key decision makers in the process. What differentiates this methodology from Trike (see below) is that it includes risk management steps in the final stage of the process to help ensure P.A.S.T.A is not limited to a specific risk calculation formula.

Trike:

Trike is a methodology used to build threat models in order to help satisfy the security auditing process from a risk management perspective. The creators of Trike also provide a tool to accompany this methodology.

The foundation of this methodology is a "requirements model" designed to ensure the level of risk assigned to each asset is classified as "acceptable" by the system's stakeholders. Once the requirements model is in place, DFDs are used to create an implementation model that illustrates data flows, including the actions that can be performed in real time and within a system state.

Finally, the implementation model is analyzed to produce a threat model. Once threats are enumerated, appropriate risk values are assigned to them and attack graphs are created. Mitigating controls are also assigned to the vulnerabilities that lead to these threats. Once this process is completed, a risk model is created based on assets, roles, actions and threat exposure.

While Trike has similarities to the Microsoft threat modeling process, it differs in that it uses a risk-based approach with distinct implementation, threat, and risk models. The Trike methodology appears to be in an experimental stage and with lack of significant documentation and support, it seems difficult to implement. Systems with more than roughly a dozen actors and assets are difficult to model as a single system using Trike v1, as this current version suffers from performance issues, even at these relatively low thresholds.

ThreatModeler, MyAppSecurity's flagship offering, is the industry's first automated, scalable, and repeatable threat modeling product. Please contact us to learn more about ThreatModeler™

Contact us for a ThreatModeler Demo 

References:

http://msdn.microsoft.com/en-us/library/ff648006.aspx#tmwa_approach

<http://msdn.microsoft.com/en-gb/magazine/cc163519.aspx>

<http://msdn.microsoft.com/en-us/library/cc307748.aspx>

<http://securesoftware.blogspot.com/2011/06/attack-simulation-and-threat-analysis.html>

<http://octotrike.org/papers/>

This entry was posted in *Threat Modeling* and tagged *Application Security, Architecture Risk Assessment, Threat Modeling*. Bookmark the *permalink*.

3 Approaches to Threat Modeling →

BLOG

- Why Implement Enterprise Threat Modeling
- Meltdown and Spectre Addressed by ThreatModeler
- Improving DevOps: Including Legacy Systems
- Improving DevOps: Shifting Security Right
- DevOps Data-Driven Transformational Leadership

□ □ □ □ □

© MyAppSecurity 2010-2020. All Rights Reserved.

Microsoft Threat Modeling Tool

8/25/2017 • 1 min to read • [Edit Online](#)

The Threat Modeling Tool is a core element of the Microsoft Security Development Lifecycle (SDL). It allows software architects to identify and mitigate potential security issues early, when they are relatively easy and cost-effective to resolve. As a result, it greatly reduces the total cost of development. Also, we designed the tool with non-security experts in mind, making threat modeling easier for all developers by providing clear guidance on creating and analyzing threat models.

The tool enables anyone to:

- Communicate about the security design of their systems
- Analyze those designs for potential security issues using a proven methodology
- Suggest and manage mitigations for security issues

Here are some tooling capabilities and innovations, just to name a few:

- **Automation:** Guidance and feedback in drawing a model
- **STRIDE per Element:** Guided analysis of threats and mitigations
- **Reporting:** Security activities and testing in the verification phase
- **Unique Methodology:** Enables users to better visualize and understand threats
- **Designed for Developers and Centered on Software:** many approaches are centered on assets or attackers. We are centered on software. We build on activities that all software developers and architects are familiar with -- such as drawing pictures for their software architecture
- **Focused on Design Analysis:** The term "threat modeling" can refer to either a requirements or a design analysis technique. Sometimes, it refers to a complex blend of the two. The Microsoft SDL approach to threat modeling is a focused design analysis technique

Next steps

The table below contains important links to get you started with the Threat Modeling Tool:

STEP	DESCRIPTION
1	Download the Threat Modeling Tool
2	Read Our getting started guide
3	Get familiar with the features
4	Learn about generated threat categories
5	Find mitigations to generated threats

Resources

Here are a few older articles still relevant to threat modeling today:

- [Article on the importance of Threat Modeling](#)
- [Training Published by Trustworthy Computing](#)

Check out what a few Threat Modeling Tool experts have done:

- [Threats Manager](#)
- [Simone Curzi Security Blog](#)

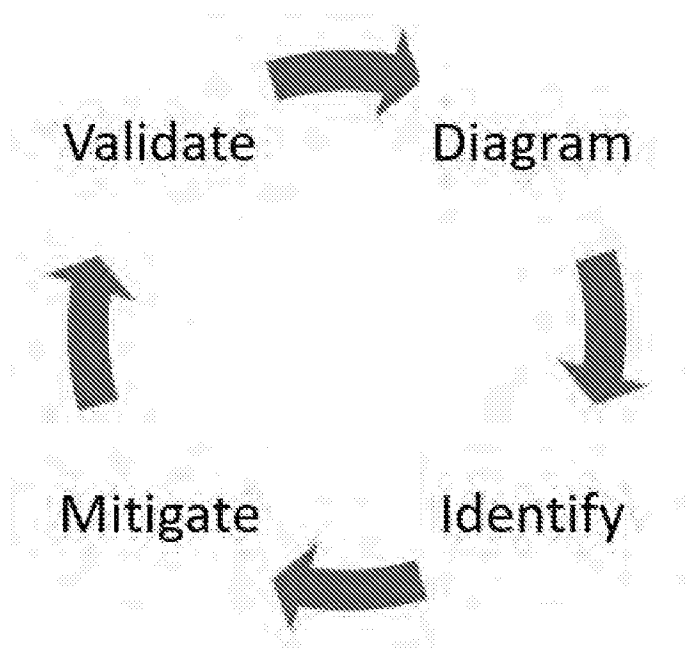
Getting started with the Threat Modeling Tool

12/13/2017 • 7 min to read • [Edit Online](#)

The Cloud and Enterprise Security Tools team released the Threat Modeling Tool Preview earlier this year as a free **click-to-download**. The change in delivery mechanism allows us to push the latest improvements and bug fixes to customers each time they open the tool, making it easier to maintain and use. This article takes you through the process of getting started with the Microsoft SDL threat modeling approach and shows you how to use the tool to develop great threat models as a backbone of your security process.

This article builds on existing knowledge of the SDL threat modeling approach. For a quick review, refer to [Threat Modeling Web Applications](#) and an archived version of [Uncover Security Flaws Using the STRIDE Approach](#) MSDN article published in 2006.

To quickly summarize, the approach involves creating a diagram, identifying threats, mitigating them and validating each mitigation. Here's a diagram that highlights this process:



Starting the threat modeling process

When you launch the Threat Modeling Tool, you'll notice a few things, as seen in the picture:

MICROSOFT MICROSOFT THREAT MODELING TOOL (PREVIEW)

Threat Model

[Feedback, Suggestions and Issues](#)

Create A Model

Model your system by drawing diagram in. Make sure you capture important details.

Open A Model

Open an existing model and analyze threats against your system. Do not worry, the tool will help you identify them.

Getting Started Guide

A step by step guide to help you get up and running now.

Template For New Models

[Azure Threat Model Template\(1.0.0.20\)](#)
[Browse...](#)

Recently Opened Models

[Basic Web App \(NEW\).tm7](#)
[New Threat Model.tm7](#)
[Library Sample.tm7](#)
[Basic Web App Sample.tm7](#)
[CSP: complete IS filtered.tm7](#)
[Downloadable Threat Model Sample and New Threat Model](#)

Threat Modeling Workflow

1. Select your template.
2. Create your data flow diagram model.
3. Analyze the model for potential threats.
4. Determine mitigations.

Template:

Create New Template

Define threats, threat types and custom threat properties for your threat model from scratch.

Open Template

Open an existing Template and make modifications to focus on your specific threat analysis.

Template Workflow

Use templates to define threats that applications should look for.

1. Define threats
2. Define categories
3. Define threat properties
4. Define threat
5. Share your template

Threat model section

COMPONENT	DETAILS
Feedback, Suggestions and Issues Button	Takes you the MSDN Forum for all things SDL. It gives you an opportunity to read through what other users are doing, along with workarounds and recommendations. If you still can't find what you're looking for, email tmtextsupport@microsoft.com for our support team to help you
Create a Model	Opens a blank canvas for you to draw your diagram. Make sure to select which template you'd like to use for your model
Template for New Models	You must select which template to use before creating a model. Our main template is the Azure Threat Model Template, which contains Azure-specific stencils, threats and mitigations. For generic models, select the SDL TM Knowledge Base from the drop-down menu. Want to create your own template or submit a new one for all users? Check out our Template Repository GitHub Page to learn more
Open a Model	Opens previously saved threat models. The Recently Opened Models feature is great if you need to open your most recent files. When you hover over the selection, you'll find 2 ways to open models: • Open From this Computer – classic way of opening a file using local storage • Open from OneDrive – teams can use folders in OneDrive to save and share all their threat models in a single location to help increase productivity and collaboration

COMPONENT	DETAILS
Getting Started Guide	Opens the Microsoft Threat Modeling Tool main page

Template section

COMPONENT	DETAILS
Create New Template	Opens a blank template for you to build on. Unless you have extensive knowledge in building templates from scratch, we recommend you to build from existing ones
Open Template	Opens existing templates for you to make changes to

The Threat Modeling Tool team is constantly working to improve tool functionality and experience. A few minor changes might take place over the course of the year, but all major changes require rewrites in the guide. Refer to it often to ensure you get the latest announcements.

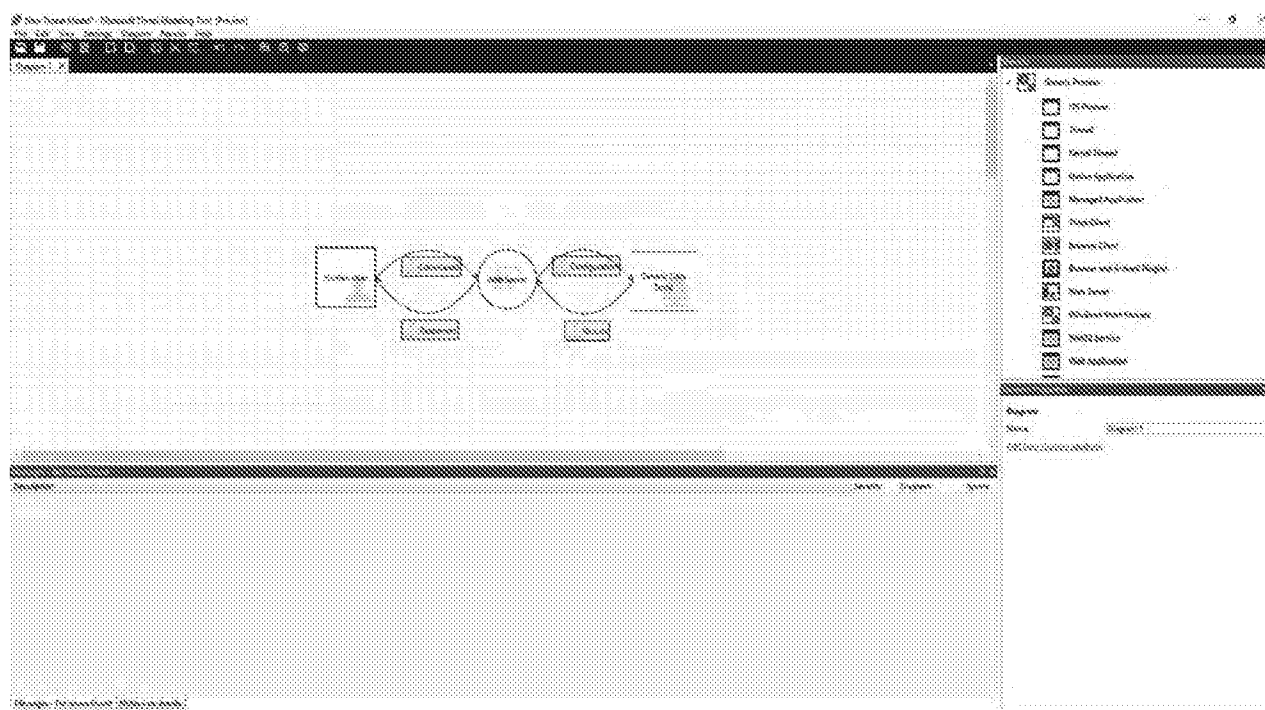
Building a model

In this section, we follow:

- Cristina (a developer)
- Ricardo (a program manager) and
- Ashish (a tester)

They are going through the process of developing their first threat model.

Ricardo: Hi Cristina, I worked on the threat model diagram and wanted to make sure we got the details right. Can you help me look it over? Cristina: Absolutely. Let's take a look. Ricardo opens the tool and shares his screen with Cristina.



Cristina: Ok, looks straightforward, but can you walk me through it? Ricardo: Sure! Here is the breakdown:

- Our human user is drawn as an outside entity—a square

- They're sending commands to our Web server—the circle
- The Web server is consulting a database (two parallel lines)

What Ricardo just showed Cristina is a DFD, short for Data Flow Diagram. The Threat Modeling Tool allows users to specify trust boundaries, indicated by the red dotted lines, to show where different entities are in control. For example, IT administrators require an Active Directory system for authentication purposes, so the Active Directory is outside of their control.

Cristina: Looks right to me. What about the threats? Ricardo: Let me show you.

Analyzing threats

Once he clicks on the analysis view from the icon menu selection (file with magnifying glass), he is taken to a list of generated threats the Threat Modeling Tool found based on the default template, which uses the SDL approach called **STRIDE (Spoofing, Tampering, Info Disclosure, Denial of Service and Elevation of Privilege)**. The idea is that software comes under a predictable set of threats, which can be found using these 6 categories.

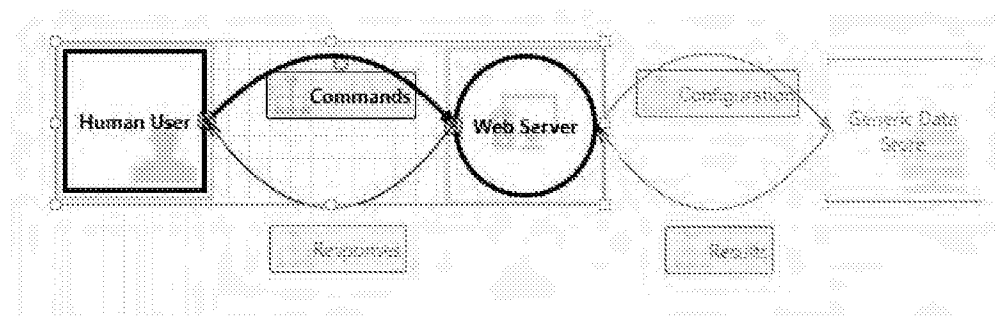
This approach is like securing your house by ensuring each door and window has a locking mechanism in place before adding an alarm system or chasing after the thief.

The screenshot shows the Microsoft Threat Modeling Tool interface. At the top, there's a diagram of a system with components like 'Human User', 'Commands', 'Web Server', 'Configuration', and 'Generic Data Store'. Below the diagram is a table of generated threats.

ID	Severity	Target ID	Trust Boundary	Task	STRIDE	Severity	Justification	Severity	Priority
1	Severe	Target1	Untrusted	Access to Data	Information Disclosure	Severe	Information Disclosure	Severe	High
2	Severe	Target1	Untrusted	Access to Data	Information Disclosure	Severe	Information Disclosure	Severe	High
3	Severe	Target1	Untrusted	Access to Data	Information Disclosure	Severe	Information Disclosure	Severe	High
4	Severe	Target1	Untrusted	Access to Data	Information Disclosure	Severe	Information Disclosure	Severe	High
5	Severe	Target1	Untrusted	Access to Data	Information Disclosure	Severe	Information Disclosure	Severe	High
6	Severe	Target1	Untrusted	Access to Data	Information Disclosure	Severe	Information Disclosure	Severe	High
7	Severe	Target1	Untrusted	Access to Data	Information Disclosure	Severe	Information Disclosure	Severe	High
8	Severe	Target1	Untrusted	Access to Data	Information Disclosure	Severe	Information Disclosure	Severe	High
9	Severe	Target1	Untrusted	Access to Data	Information Disclosure	Severe	Information Disclosure	Severe	High
10	Severe	Target1	Untrusted	Access to Data	Information Disclosure	Severe	Information Disclosure	Severe	High

Ricardo begins by selecting the first item on the list. Here's what happens:

First, the interaction between the two stencils is enhanced



Second, additional information about the threat appears in the Threat Properties window

Threat Properties		
ID:	0	Diagram: Diagram 1 Status: Not Started
Title:	Spoofing the Human User External Entity	
Category:	Spoofing	
Description:	Human User may be spoofed by an attacker and this may lead to unauthorized access to Web Server. Consider using a standard authentication mechanism to identify the external entity.	
Justification:		
Interaction:	Commands	
Priority:	High	

The generated threat helps him understand potential design flaws. The STRIDE categorization gives him an idea on potential attack vectors, while the additional description tells him exactly what's wrong, along with potential ways to mitigate it. He can use editable fields to write notes in the justification details or change priority ratings depending on his organization's bug bar.

Azure templates have additional details to help users understand not only what's wrong, but also how to fix it by adding descriptions, examples and hyperlinks to Azure-specific documentation.

The description made him realize the importance of adding an authentication mechanism to prevent users from being spoofed, revealing the first threat to be worked on. A few minutes into the discussion with Cristina, they understood the importance of implementing access control and roles. Ricardo filled in some quick notes to make sure these were implemented.

As Ricardo went into the threats under Information Disclosure, he realized the access control plan required some read-only accounts for audit and report generation. He wondered whether this should be a new threat, but the mitigations were the same, so he noted the threat accordingly. He also thought about information disclosure a bit more and realized that the backup tapes were going to need encryption, a job for the operations team.

Threats not applicable to the design due to existing mitigations or security guarantees can be changed to "Not Applicable" from the Status drop-down. There are three other choices: Not Started – default selection, Needs Investigation – used to follow up on items and Mitigated – once it's fully worked on.

Reports & sharing

Once Ricardo goes through the list with Cristina and adds important notes, mitigations/justifications, priority and status changes, he selects Reports -> Create Full Report -> Save Report, which prints out a nice report for him to go through with colleagues to ensure the proper security work is implemented.

Threat Modeling Report

Created on 7/31/2017 12:35:43 PM

Threat Model Name:

Owner:

Reviewer:

Contributors:

Description:

Assumptions:

External Dependencies:

Threat Model Summary:

Not Started: 0
Not Applicable: 0
Needs Investigation: 0
Mitigation Implemented: 0
Total: 0
Total Migrated: 0

Diagram: Diagram 1

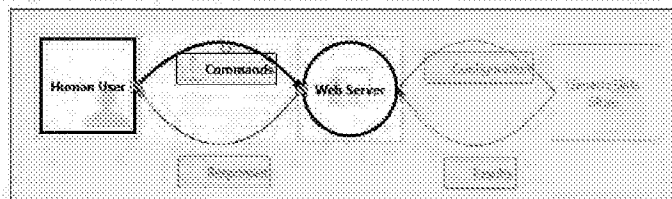
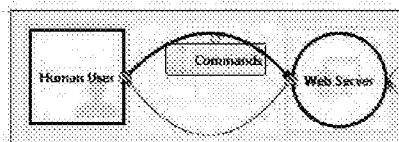


Diagram 1 Diagram Summary:

Not Started: 0
Not Applicable: 0
Needs Investigation: 0
Mitigation Implemented: 0
Total: 0
Total Migrated: 0

Interaction: Commands



1. Spoofing the Human User External Entity [State: Not Started] [Priority: High]

Category: Spoofing
Description: Human User may be spoofed by an attacker and this may lead to unauthorized access to Web Server. Consider using a standard authentication mechanism to identify the external entity.
Justification: (no mitigation provided)
Possible Mitigation(s):
SDL Phase: Design

2. Cross Site Scripting [State: Not Started] [Priority: High]

Category: Tampering
Description: The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: (no mitigation provided)
Possible Mitigation(s):
SDL Phase: Design

If Ricardo wants to share the file instead, he can easily do so by saving in his organization's OneDrive account. Once he does that, he can copy the document link and share it with his colleagues.

Threat modeling meetings

When Ricardo sent his threat model to his colleague using OneDrive, Ashish, the tester, was underwhelmed. Seemed like Ricardo and Cristina missed quite a few important corner cases, which could be easily compromised. His skepticism is a complement to threat models.

In this scenario, after Ashish took over the threat model, he called for two threat modeling meetings: one meeting to synchronize on the process and walk through the diagrams and then a second meeting for threat review and sign-off.

In the first meeting, Ashish spent 10 minutes walking everyone through the SDL threat modeling process. He then pulled up the threat model diagram and started explaining it in detail. Within five minutes, an important missing component had been identified.

A few minutes later, Ashish and Ricardo got into an extended discussion of how the Web server was built. It was not the ideal way for a meeting to proceed, but everyone eventually agreed that discovering the discrepancy early was going to save them time in the future.

In the second meeting, the team walked through the threats, discussed some ways to address them, and signed off on the threat model. They checked the document into source control and continued with development.

Thinking about assets

Some readers who have threat modeled may notice that we haven't talked about assets at all. We've discovered that many software engineers understand their software better than they understand the concept of assets and what assets an attacker may be interested in.

If you're going to threat model a house, you might start by thinking about your family, irreplaceable photos or valuable artwork. Perhaps you might start by thinking about who might break in and the current security system. Or you might start by considering the physical features, like the pool or the front porch. These are analogous to thinking about assets, attackers, or software design. Any of these three approaches work.

The approach to threat modeling we've presented here is substantially simpler than what Microsoft has done in the past. We found that the software design approach works well for many teams. We hope that include yours.

Next Steps

Send your questions, comments and concerns to tmtextsupport@microsoft.com. [Download](#) the Threat Modeling Tool to get started.

Threat Modeling Tool feature overview

12/18/2017 • 5 min to read • [Edit Online](#)

The Threat Modeling Tool can help you with your threat modeling needs. For a basic introduction to the tool, see [Get started with the Threat Modeling Tool](#).

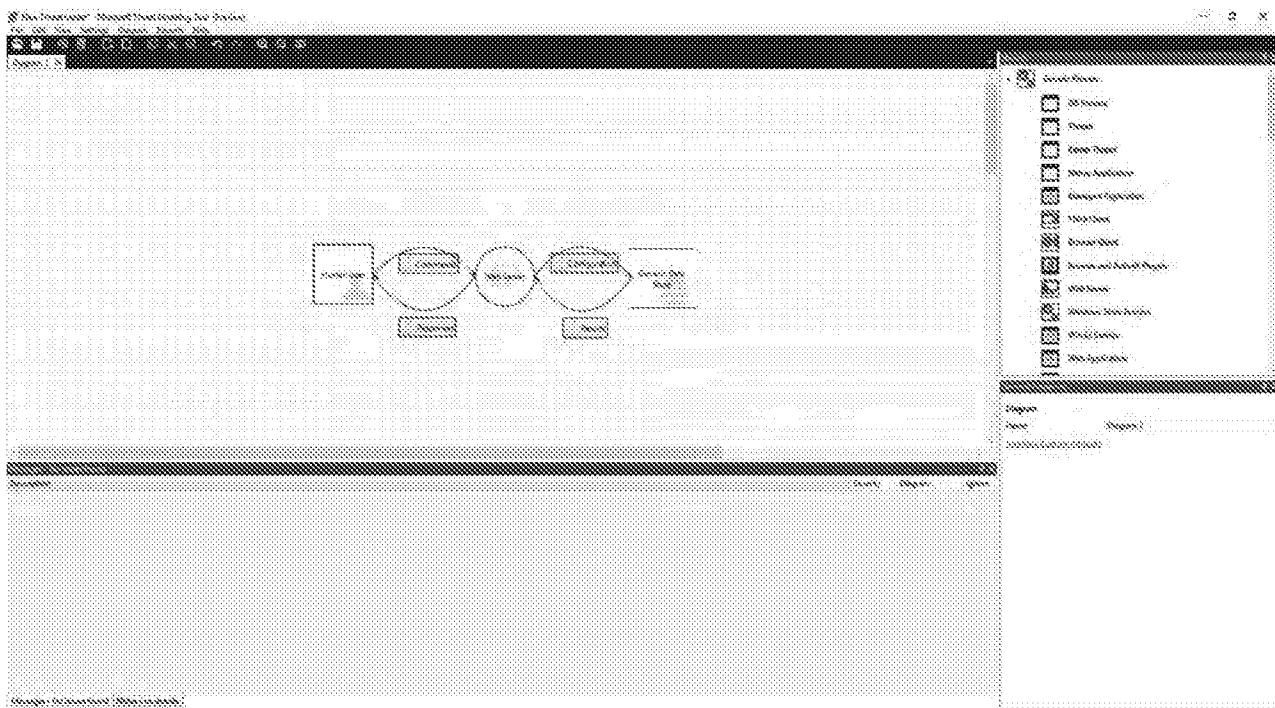
NOTE

The Threat Modeling Tool is updated frequently, so check this guide often to see our latest features and improvements.

To open a blank page, select **Create A Model**.



To see the features currently available in the tool, use the threat model created by our team in the [Get started](#) example.



Navigation

Before we discuss the built-in features, let's review the main components found in the tool.

Menu items

The experience is similar to other Microsoft products. Let's review the top-level menu items.



LABEL	DETAILS
File	• Open, save, and close files • Sign in and sign out of OneDrive accounts. • Share links (view and edit). • View file information. • Apply a new template to existing models.
Edit	Undo and redo actions, as well as copy, paste, and delete.
View	• Switch between Analysis and Design views. • Open closed windows (for example, stencils, element properties, and messages). • Reset layout to default settings.
Diagram	Add and delete diagrams, and move through tabs of diagrams.
Reports	Create HTML reports to share with others.
Help	Find guides to help you use the tool.

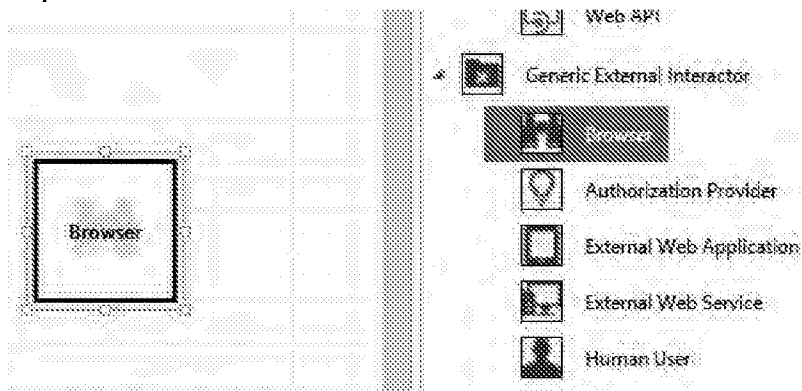
The symbols are shortcuts for the top-level menus:

SYMBOL	DETAILS
Open	Opens a new file.
Save	Saves the current file.
Design	Opens the Design view, where you can create models.
Analyze	Shows generated threats and their properties.
Add diagram	Adds a new diagram (similar to new tabs in Excel).
Delete diagram	Deletes the current diagram.
Copy/Cut/Paste	Copies, cuts, and pastes elements.
Undo/Redo	Undoes and redoes actions.
Zoom in/Zoom out	Zooms in and out of the diagram for a better view.
Feedback	Opens the MSDN Forum.

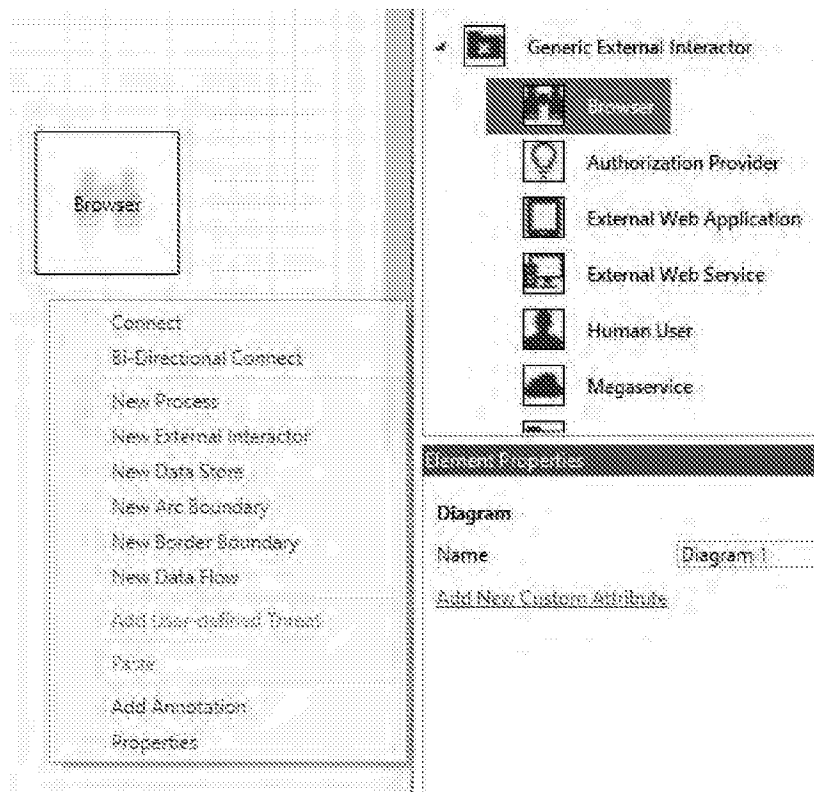
Canvas

The canvas is the space where you drag and drop elements. Drag and drop is the quickest and most efficient way to build models. You can also right-click and select items from the menu to add generic versions of elements, as shown:

Drop the stencil on the canvas



Select the stencil



Stencils

Based on the template you select, you can find all the stencils available to use. If you can't find the right elements, use another template. Or you can modify a template to fit your needs. Generally, you can find a combination of categories like these:

STENCIL NAME	DETAILS
Process	Applications, browser plug-ins, threads, virtual machines
External interactor	Authentication providers, browsers, users, web applications
Data store	Cache, storage, configuration files, databases, registry
Data flow	Binary, ALPC, HTTP, HTTPS/TLS/SSL, IOCTL, IPsec, named pipe, RPC/DCOM, SMB, UDP
Trust line/Border boundary	Corporate networks, internet, machine, sandbox, user/kernel mode

Notes/messages

COMPONENT	DETAILS
Messages	Internal tool logic that alerts users whenever there's an error, such as no data flows between elements.
Notes	Manual notes are added to the file by engineering teams throughout the design and review process.

Element properties

Element properties vary by the elements you select. Apart from trust boundaries, all other elements contain three general selections:

ELEMENT PROPERTY	DETAILS
Name	Useful for naming your processes, stores, interactors, and flows so that they're easily recognized.
Out of scope	If selected, the element is taken out of the threat-generation matrix (not recommended).
Reason for out of scope	Justification field to let users know why out of scope was selected.

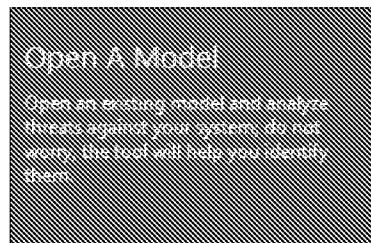
Properties are changed under each element category. Select each element to inspect the available options. Or you can open the template to learn more. Let's review the features.

Welcome screen

When you open the app, you see the **Welcome** screen.

Open a model

Hover over **Open A Model** to reveal two options: **Open From This Computer** and **Open From OneDrive**. The first option opens the **File Open** screen. The second option takes you through the sign-in process for OneDrive. After successful authentication, you can select folders and files.



Feedback, suggestions, and issues

When you select **Feedback, Suggestions and Issues**, you go to the MSDN Forum for SDL Tools. You can read what other people are saying about the tool, including workarounds and new ideas.



Design view

When you open or create a new model, the **Design** view opens.

Add elements

You can add elements on the grid in two ways:

- **Drag and drop:** Drag the desired element to the grid. Then use the element properties to provide additional information.
- **Right-click:** Right-click anywhere on the grid, and select items from the drop-down menu. A generic

representation of the element you select appears on the screen.

Connect elements

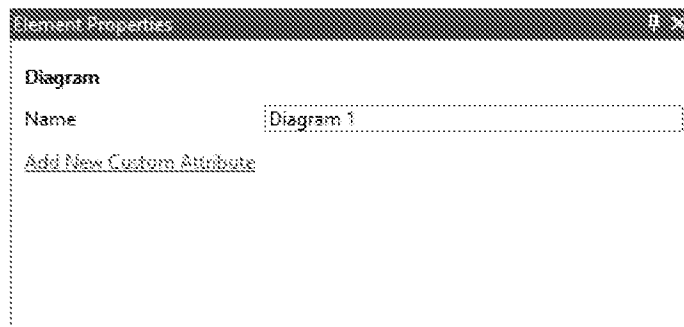
You can connect elements in two ways:

- **Drag and drop:** Drag the desired dataflow to the grid, and connect both ends to the appropriate elements.
- **Click + Shift:** Click the first element (sending data), press and hold the Shift key, and then select the second element (receiving data). Right-click, and select **Connect**. If you use a bi-directional data flow, the order is not as important.

Properties

To see the properties that can be modified on the stencils, select the stencil and the information populates accordingly. The following example shows before and after a **Database** stencil is dragged onto the diagram:

Before



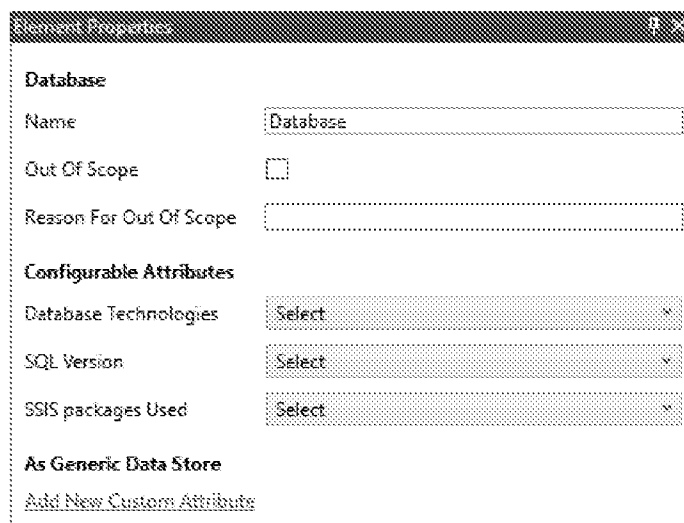
Element Properties

Diagram

Name:

[Add New Custom Attribute](#)

After



Element Properties

Database

Name:

Out Of Scope: ☐

Reason For Out Of Scope:

Configurable Attributes

Database Technologies:

SQL Version:

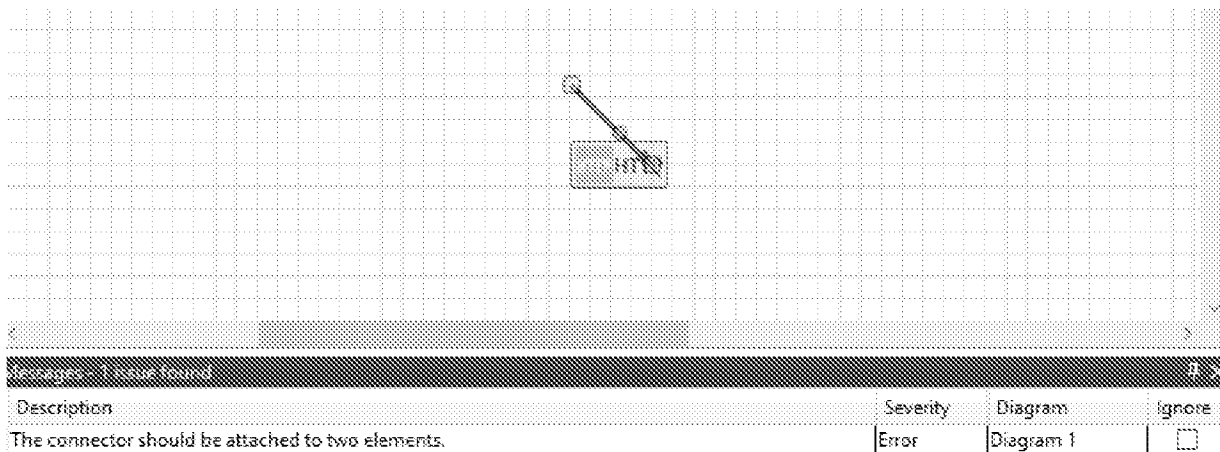
SSIS packages Used:

As Generic Data Store

[Add New Custom Attribute](#)

Messages

If you create a threat model and forget to connect data flows to elements, you get a notification. You can ignore the message, or you can follow the instructions to fix the issue.



Notes

To add notes to your diagram, switch from the **Messages** tab to the **Notes** tab.

Analysis view

After you build your diagram, select the **Analysis** symbol (the magnifying glass) on the shortcuts toolbar to switch to the **Analysis** view.

New Threat Model - Microsoft Threat Modeling Tool (Preview)

File Edit View Settings Diagram Reports Help

Diagram 1 X

Threat List

ID	Diagram	Changed By	Last Modified	State	Title	Category	Description	Justification	Interaction	Priority
1	Diagram 1	Generated	Not Started	Cross Site Scr...	Tampering	The web serv...	Commands	High		
2	Diagram 1	Generated	Not Started	Elevation of Pr...	Elevation of Pr...	Web Server...	Commands	High		
3	Diagram 1	Generated	Not Started	Spawning of D...	Spawning	Generic Data...	Configuration	High		
4	Diagram 1	Generated	Not Started	Potential for...	Denial of Ser...	Does Web Ser...	Configuration	High		
5	Diagram 1	Generated	Not Started	Spawning of S...	Spawning	Generic Data...	Results	High		
6	Diagram 1	Generated	Not Started	Cross Site Scr...	Tampering	The web serv...	Results	High		
7	Diagram 1	Generated	Not Started	Persistent Cr...	Tampering	The web serv...	Results	High		
8	Diagram 1	Generated	Not Started	Weak Accest...	Information...	Improper dat...	Results	High		
9	Diagram 1	Generated	Not Started	Spawning the...	Spawning	Human User...	Commands	High		

9 Threats Displayed, 3 Total

No threats are selected

Threat Properties Notes - no entries

Generated threat selection

When you select a threat, you can use three distinct functions:

FEATURE

Read indicator

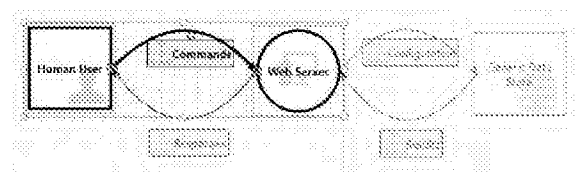
INFORMATION

The threat is marked as read, which helps you keep track of the items you reviewed.

ID	Diagram	Changed By	Last Modified	State	Title	Category	Description	Justification	Interaction	Priority
1	Diagram 1	Generated	Not Started	Cross Site Scr...	Tampering	The web serv...	Commands	High		
2	Diagram 1	Generated	Not Started	Elevation of Pr...	Elevation of Pr...	Web Server...	Commands	High		
3	Diagram 1	Generated	Not Started	Spawning of D...	Spawning	Generic Data...	Configuration	High		

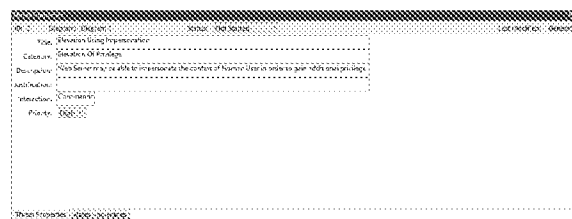
Interaction focus

Interaction in the diagram that belongs to a threat is highlighted.



Threat properties

Additional information about the threat appears in the **Threat Properties** window.



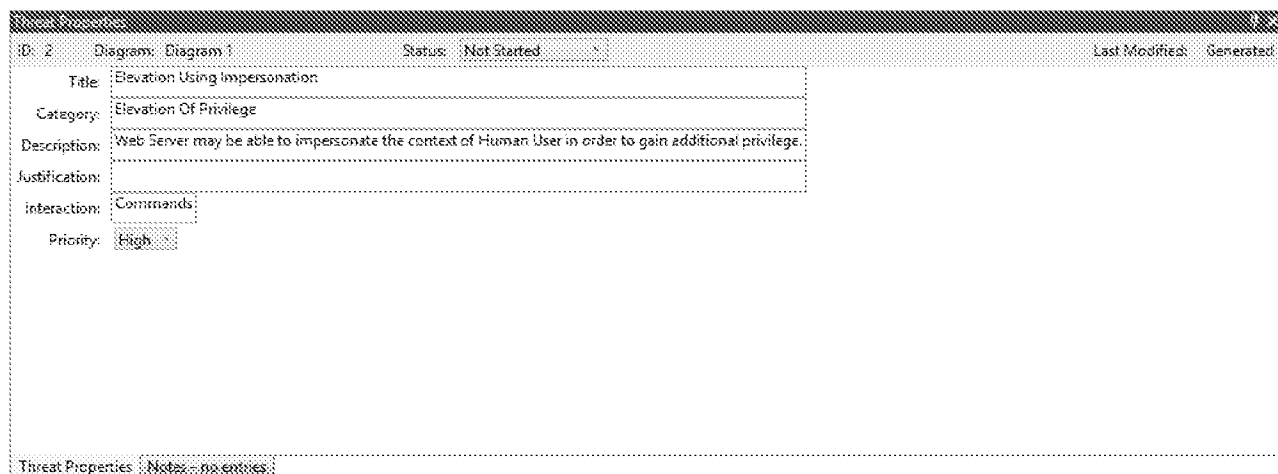
Priority change

You can change the priority level of each generated threat. Different colors make it easy to identify high-, medium-, and low-priority threats.

ID	Diagram	Changed By	Last Modified	State	Title	Category	Description	Justification	Interaction	Priority
1	Diagram 1	ESD&W&ND&W...	8/16/2017 3:13...	Not Started	Command Line S...	Tampering	Changing comm...		Commands	Medium
2	Diagram 1	ESD&W&ND&W...	8/16/2017 3:13...	Not Started	Elevation Using...	Elevation Of Pr...	Web Server may...		Commands	Low
3	Diagram 1	ESD&W&ND&W...	8/16/2017 3:13...	Not Started	Spooling of De...	Spooling	Changing Data L...		Configuration	Medium
4	Diagram 1	ESD&W&ND&W...	8/16/2017 3:13...	Not Started	Potential Escal...	Privileged Data...	Using Tools S...		Configuration	Medium
5	Diagram 1	Generated	Generated	Not Started	Spooling of Se...	Spooling	Changing Data L...		Results	High
6	Diagram 1	Generated	Generated	Not Started	Cross Site Scri...	Tampering	The web server...		Results	High
7	Diagram 1	ESD&W&ND&W...	8/16/2017 3:13...	Not Started	Persistent Coo...	Tampering	The web server...		Results	Low
8	Diagram 1	Generated	Generated	Not Started	Weak Access	Information	Inappropriate al...		Results	High
9	Diagram 1	Generated	Generated	Not Started	Spooling the...	Spooling	Human User...		Commands	High

Threat properties editable fields

As seen in the preceding image, you can change the information generated by the tool. You can also add information to certain fields, such as justification. These fields are generated by the template. If you need more information for each threat, you can make modifications.



Reports

After you finish changing priorities and updating the status of each generated threat, you can save the file and/or print out a report. Go to **Report > Create Full Report**. Name the report, and you should see something similar to the following image:

Threat Modeling Report

Created on 7/31/2017 12:35:42 PM

Threat Model Name:

Owner:

Reviewer:

Contributors:

Description:

Assumptions:

External Dependencies:

Threat Model Summary:

Not Started	0
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	0
Total Mitigated	0

Diagram: Diagram 1

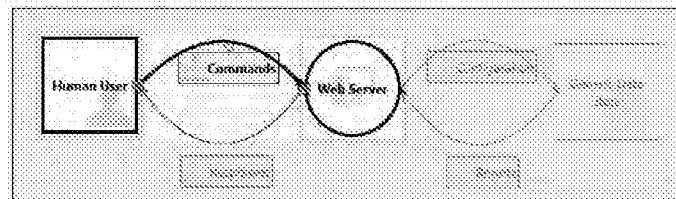
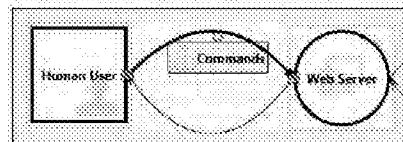


Diagram 1 Diagram Summary:

Not Started	0
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	0
Total Mitigated	0

Interaction: Commands



1. Spoofing the Human User External Entity [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	Human user may be spoofed by an attacker and this may lead to unauthorized access to Web Server. Consider using a standard authentication mechanism to identify the external entity.
Justification:	<no mitigation provided>
Possible Mitigation(s):	
SDL Phase:	Design

2. Cross Site Scripting [State: Not Started] [Priority: High]

Category:	Tampering
Description:	The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification:	<no mitigation provided>
Possible Mitigation(s):	
SDL Phase:	Design

Next steps

- To contribute a template for the community, go to our [GitHub page](#).
- To get started with the tool, go to the [Download page](#).

Microsoft Threat Modeling Tool threats

8/25/2017 • 2 min to read • [Edit Online](#)

The Threat Modeling Tool is a core element of the Microsoft Security Development Lifecycle (SDL). It allows software architects to identify and mitigate potential security issues early, when they are relatively easy and cost-effective to resolve. As a result, it greatly reduces the total cost of development. Also, we designed the tool with non-security experts in mind, making threat modeling easier for all developers by providing clear guidance on creating and analyzing threat models.

Visit the **Threat Modeling Tool** to get started today!

The Threat Modeling Tool helps you answer certain questions, such as the ones below:

- How can an attacker change the authentication data?
- What is the impact if an attacker can read the user profile data?
- What happens if access is denied to the user profile database?

STRIDE model

To better help you formulate these kinds of pointed questions, Microsoft uses the STRIDE model, which categorizes different types of threats and simplifies the overall security conversations.

CATEGORY	DESCRIPTION
Spoofing	Involves illegally accessing and then using another user's authentication information, such as username and password
Tampering	Involves the malicious modification of data. Examples include unauthorized changes made to persistent data, such as that held in a database, and the alteration of data as it flows between two computers over an open network, such as the Internet
Repudiation	Associated with users who deny performing an action without other parties having any way to prove otherwise—for example, a user performs an illegal operation in a system that lacks the ability to trace the prohibited operations. Non-Repudiation refers to the ability of a system to counter repudiation threats. For example, a user who purchases an item might have to sign for the item upon receipt. The vendor can then use the signed receipt as evidence that the user did receive the package
Information Disclosure	Involves the exposure of information to individuals who are not supposed to have access to it—for example, the ability of users to read a file that they were not granted access to, or the ability of an intruder to read data in transit between two computers
Denial of Service	Denial of service (DoS) attacks deny service to valid users—for example, by making a Web server temporarily unavailable or unusable. You must protect against certain types of Denial of Service threats simply to improve system availability and reliability

CATEGORY	DESCRIPTION
Elevation of Privilege	An unprivileged user gains privileged access and thereby has sufficient access to compromise or destroy the entire system. Elevation of privilege threats include those situations in which an attacker has effectively penetrated all system defenses and become part of the trusted system itself, a dangerous situation indeed

Next steps

Proceed to **Threat Modeling Tool Mitigations** to learn the different ways you can mitigate these threats with Azure.

Microsoft Threat Modeling Tool mitigations

8/25/2017 • 2 min to read • [Edit Online](#)

The Threat Modeling Tool is a core element of the Microsoft Security Development Lifecycle (SDL). It allows software architects to identify and mitigate potential security issues early, when they are relatively easy and cost-effective to resolve. As a result, it greatly reduces the total cost of development. Also, we designed the tool with non-security experts in mind, making threat modeling easier for all developers by providing clear guidance on creating and analyzing threat models.

Visit the **Threat Modeling Tool** to get started today!

Mitigation categories

The Threat Modeling Tool mitigations are categorized according to the Web Application Security Frame, which consists of the following:

CATEGORY	DESCRIPTION
Auditing and Logging	Who did what and when? Auditing and logging refer to how your application records security-related events
Authentication	Who are you? Authentication is the process where an entity proves the identity of another entity, typically through credentials, such as a user name and password
Authorization	What can you do? Authorization is how your application provides access controls for resources and operations
Communication Security	Who are you talking to? Communication Security ensures all communication done is as secure as possible
Configuration Management	Who does your application run as? Which databases does it connect to? How is your application administered? How are these settings secured? Configuration management refers to how your application handles these operational issues
Cryptography	How are you keeping secrets (confidentiality)? How are you tamper-proofing your data or libraries (integrity)? How are you providing seeds for random values that must be cryptographically strong? Cryptography refers to how your application enforces confidentiality and integrity
Exception Management	When a method call in your application fails, what does your application do? How much do you reveal? Do you return friendly error information to end users? Do you pass valuable exception information back to the caller? Does your application fail gracefully?

CATEGORY	DESCRIPTION
Input Validation	How do you know that the input your application receives is valid and safe? Input validation refers to how your application filters, scrubs, or rejects input before additional processing. Consider constraining input through entry points and encoding output through exit points. Do you trust data from sources such as databases and file shares?
Sensitive Data	How does your application handle sensitive data? Sensitive data refers to how your application handles any data that must be protected either in memory, over the network, or in persistent stores
Session Management	How does your application handle and protect user sessions? A session refers to a series of related interactions between a user and your Web application

This helps you identify:

- Where are the most common mistakes made
- Where are the most actionable improvements

As a result, you use these categories to focus and prioritize your security work, so that if you know the most prevalent security issues occur in the input validation, authentication and authorization categories, you can start there. For more information visit [this patent link](#)

Next steps

Visit [Threat Modeling Tool Threats](#) to learn more about the threat categories the tool uses to generate possible design threats.

Security Frame: Auditing and Logging | Mitigations

8/22/2017 • 8 min to read • [Edit Online](#)

PRODUCT/SERVICE	ARTICLE
Dynamics CRM	• Identify sensitive entities in your solution and implement change auditing
Web Application	• Ensure that auditing and logging is enforced on the application • Ensure that log rotation and separation are in place • Ensure that the application does not log sensitive user data • Ensure that Audit and Log Files have Restricted Access • Ensure that User Management Events are Logged • Ensure that the system has inbuilt defenses against misuse • Enable diagnostics logging for web apps in Azure App Service
Database	• Ensure that login auditing is enabled on SQL Server • Enable Threat detection on Azure SQL
Azure Storage	• Use Azure Storage Analytics to audit access of Azure Storage
WCF	• Implement sufficient Logging • Implement sufficient Audit Failure Handling
Web API	• Ensure that auditing and logging is enforced on Web API
IoT Field Gateway	• Ensure that appropriate auditing and logging is enforced on Field Gateway
IoT Cloud Gateway	• Ensure that appropriate auditing and logging is enforced on Cloud Gateway

Identify sensitive entities in your solution and implement change auditing

TITLE	DETAILS
Component	Dynamics CRM

TITLE	DETAILS
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Identify entities in your solution containing sensitive data and implement change auditing on those entities and fields

Ensure that auditing and logging is enforced on the application

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Enable auditing and logging on all components. Audit logs should capture user context. Identify all important events and log those events. Implement centralized logging

Ensure that log rotation and separation are in place

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A

TITLE	DETAILS
Steps	Log rotation is an automated process used in system administration in which dated log files are archived. Servers which run large applications often log every request: in the face of bulky logs, log rotation is a way to limit the total size of the logs while still allowing analysis of recent events. Log separation basically means that you have to store your log files on a different partition as where your OS/application is running on in order to avert a Denial of service attack or the downgrading of your application its performance

Ensure that the application does not log sensitive user data

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Check that you do not log any sensitive data that a user submits to your site. Check for intentional logging as well as side effects caused by design issues. Examples of sensitive data include: • User Credentials • Social Security number or other identifying information • Credit card numbers or other financial information • Health information • Private keys or other data that could be used to decrypt encrypted information • System or application information that can be used to more effectively attack the application

Ensure that Audit and Log Files have Restricted Access

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A

TITLE	DETAILS
References	N/A
Steps	Check to ensure access rights to log files are appropriately set. Application accounts should have write-only access and operators and support personnel should have read-only access as needed. Administrators accounts are the only accounts which should have full access. Check Windows ACL on log files to ensure they are properly restricted: • Application accounts should have write-only access • Operators and support personnel should have read-only access as needed • Administrators are the only accounts that should have full access

Ensure that User Management Events are Logged

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure that the application monitors user management events such as successful and failed user logins, password resets, password changes, account lockout, user registration. Doing this helps to detect and react to potentially suspicious behavior. It also enables to gather operations data; for example, to track who is accessing the application

Ensure that the system has inbuilt defenses against misuse

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A

TITLE	DETAILS
References	N/A
Steps	Controls should be in place which throw security exception in case of application misuse. E.g., If input validation is in place and an attacker attempts to inject malicious code that does not match the regex, a security exception can be thrown which can be an indicative of system misuse For example, it is recommended to have security exceptions logged and actions taken for the following issues: • Input validation • CSRF violations • Brute force (upper limit for number of requests per user per resource) • File upload violations

Enable diagnostics logging for web apps in Azure App Service

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	EnvironmentType - Azure
References	N/A
Steps	Azure provides built-in diagnostics to assist with debugging an App Service web app. It also applies to API apps and mobile apps. App Service web apps provide diagnostic functionality for logging information from both the web server and the web application. These are logically separated into web server diagnostics and application diagnostics

Ensure that login auditing is enabled on SQL Server

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A

TITLE	DETAILS
References	Configure Login Auditing
Steps	Database Server login auditing must be enabled to detect/confirm password guessing attacks. It is important to capture failed login attempts. Capturing both successful and failed login attempts provides additional benefit during forensic investigations

Enable Threat detection on Azure SQL

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	SQL Azure
Attributes	SQL Version - V12
References	Get Started with SQL Database Threat Detection
Steps	Threat Detection detects anomalous database activities indicating potential security threats to the database. It provides a new layer of security, which enables customers to detect and respond to potential threats as they occur by providing security alerts on anomalous activities. Users can explore the suspicious events using Azure SQL Database Auditing to determine if they result from an attempt to access, breach or exploit data in the database. Threat Detection makes it simple to address potential threats to the database without the need to be a security expert or manage advanced security monitoring systems

Use Azure Storage Analytics to audit access of Azure Storage

TITLE	DETAILS
Component	Azure Storage
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	Using Storage Analytics to monitor authorization type

TITLE	DETAILS
Steps	For each storage account, one can enable Azure Storage Analytics to perform logging and store metrics data. The storage analytics logs provide important information such as authentication method used by someone when they access storage. This can be really helpful if you are tightly guarding access to storage. For example, in Blob Storage you can set all of the containers to private and implement the use of an SAS service throughout your applications. Then you can check the logs regularly to see if your blobs are accessed using the storage account keys, which may indicate a breach of security, or if the blobs are public but they shouldn't be.

Implement sufficient Logging

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	.NET Framework
Attributes	N/A
References	MSDN, Fortify Kingdom
Steps	The lack of a proper audit trail after a security incident can hamper forensic efforts. Windows Communication Foundation (WCF) offers the ability to log successful and/or failed authentication attempts. Logging failed authentication attempts can warn administrators of potential brute-force attacks. Similarly, logging successful authentication events can provide a useful audit trail when a legitimate account is compromised. Enable WCF's service security audit feature

Example

The following is an example configuration with auditing enabled

```
<system.serviceModel>
  <behaviors>
    <serviceBehaviors>
      <behavior name=""NewBehavior"">
        <serviceSecurityAudit auditLogLocation=""Default""
          suppressAuditFailure=""false""
          serviceAuthorizationAuditLevel=""SuccessAndFailure""
          messageAuthenticationAuditLevel=""SuccessAndFailure"" />
        ...
      </behavior>
    </serviceBehaviors>
  </behaviors>
</system.serviceModel>
```

Implement sufficient Audit Failure Handling

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	.NET Framework
Attributes	N/A
References	MSDN, Fortify Kingdom
Steps	Developed solution is configured not to generate an exception when it fails to write to an audit log. If WCF is configured not to throw an exception when it is unable to write to an audit log, the program will not be notified of the failure and auditing of critical security events may not occur.

Example

The `<behavior/>` element of the WCF configuration file below instructs WCF to not notify the application when WCF fails to write to an audit log.

```
<behaviors>
  <serviceBehaviors>
    <behavior name="NewBehavior">
      <serviceSecurityAudit auditLogLocation="Application"
        suppressAuditFailure="true"
        serviceAuthorizationAuditLevel="Success"
        messageAuthenticationAuditLevel="Success" />
    </behavior>
  </serviceBehaviors>
</behaviors>
```

Configure WCF to notify the program whenever it is unable to write to an audit log. The program should have an alternative notification scheme in place to alert the organization that audit trails are not being maintained.

Ensure that auditing and logging is enforced on Web API

TITLE	DETAILS
Component	Web API
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A

TITLE	DETAILS
Steps	Enable auditing and logging on Web APIs. Audit logs should capture user context. Identify all important events and log those events. Implement centralized logging

Ensure that appropriate auditing and logging is enforced on Field Gateway

TITLE	DETAILS
Component	IoT Field Gateway
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	When multiple devices connect to a Field Gateway, ensure that connection attempts and authentication status (success or failure) for individual devices are logged and maintained on the Field Gateway. Also, in cases where Field Gateway is maintaining the IoT Hub credentials for individual devices, ensure that auditing is performed when these credentials are retrieved. Develop a process to periodically upload the logs to Azure IoT Hub/storage for long term retention.

Ensure that appropriate auditing and logging is enforced on Cloud Gateway

TITLE	DETAILS
Component	IoT Cloud Gateway
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Introduction to IoT Hub operations monitoring

TITLE	DETAILS
Steps	Design for collecting and storing audit data gathered through IoT Hub Operations Monitoring. Enable the following monitoring categories: • Device identity operations • Device-to-cloud communications • Cloud-to-device communications • Connections • File uploads

Security Frame: Authentication | Mitigations

1/24/2018 • 25 min to read • [Edit Online](#)

PRODUCT/SERVICE	ARTICLE
Web Application	• Consider using a standard authentication mechanism to authenticate to Web Application • Applications must handle failed authentication scenarios securely • Enable step up or adaptive authentication • Ensure that administrative interfaces are appropriately locked down • Implement forgot password functionalities securely • Ensure that password and account policy are implemented • Implement controls to prevent username enumeration
Database	• When possible, use Windows Authentication for connecting to SQL Server • When possible use Azure Active Directory Authentication for Connecting to SQL Database • When SQL authentication mode is used, ensure that account and password policy are enforced on SQL server • Do not use SQL Authentication in contained databases
Azure Event Hub	• Use per device authentication credentials using SaaS tokens
Azure Trust Boundary	• Enable Azure Multi-Factor Authentication for Azure Administrators
Service Fabric Trust Boundary	• Restrict anonymous access to Service Fabric Cluster • Ensure that Service Fabric client-to-node certificate is different from node-to-node certificate • Use AAD to authenticate clients to service fabric clusters • Ensure that service fabric certificates are obtained from an approved Certificate Authority (CA)
Identity Server	• Use standard authentication scenarios supported by Identity Server • Override the default Identity Server token cache with a scalable alternative
Machine Trust Boundary	• Ensure that deployed application's binaries are digitally signed

PRODUCT/SERVICE	ARTICLE
WCF	• Enable authentication when connecting to MSMQ queues in WCF • WCF-Do not set Message clientCredentialType to none • WCF-Do not set Transport clientCredentialType to none
Web API	• Ensure that standard authentication techniques are used to secure Web APIs
Azure AD	• Use standard authentication scenarios supported by Azure Active Directory • Override the default ADAL token cache with a scalable alternative • Ensure that TokenReplayCache is used to prevent the replay of ADAL authentication tokens • Use ADAL libraries to manage token requests from OAuth2 clients to AAD (or on-premises AD)
IoT Field Gateway	• Authenticate devices connecting to the Field Gateway
IoT Cloud Gateway	• Ensure that devices connecting to Cloud gateway are authenticated • Use per-device authentication credentials
Azure Storage	• Ensure that only the required containers and blobs are given anonymous read access • Grant limited access to objects in Azure storage using SAS or SAP

Consider using a standard authentication mechanism to authenticate to Web Application

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A

TITLE	DETAILS
Details	Authentication is the process where an entity proves its identity, typically through credentials, such as a user name and password. There are multiple authentication protocols available which may be considered. Some of them are listed below: • Client certificates • Windows based • Forms based • Federation - ADFS • Federation - Azure AD • Federation - Identity Server Consider using a standard authentication mechanism to identify the source process

Applications must handle failed authentication scenarios securely

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Details	Applications that explicitly authenticate users must handle failed authentication scenarios securely. The authentication mechanism must: • Deny access to privileged resources when authentication fails • Display a generic error message after failed authentication and access denied occurs Test for: • Protection of privileged resources after failed logins • A generic error message is displayed on failed authentication and access denied event(s) • Accounts are disabled after an excessive number of failed attempts

Enable step up or adaptive authentication

TITLE	DETAILS
Component	Web Application
SDL Phase	Build

TITLE	DETAILS
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Details	Verify the application has additional authorization (such as step up or adaptive authentication, via multi-factor authentication such as sending OTP in SMS, email etc. or prompting for re-authentication) so the user is challenged before being granted access to sensitive information. This rule also applies for making critical changes to an account or action This also means that the adaptation of authentication has to be implemented in such a manner that the application correctly enforces context-sensitive authorization so as to not allow unauthorized manipulation by means of in example, parameter tampering

Ensure that administrative interfaces are appropriately locked down

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Details	The first solution is to grant access only from a certain source IP range to the administrative interface. If that solution would not be possible than it is always recommended to enforce a step-up or adaptive authentication for logging in into the administrative interface

Implement forgot password functionalities securely

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A

TITLE	DETAILS
References	N/A
Details	The first thing is to verify that forgot password and other recovery paths send a link including a time-limited activation token rather than the password itself. Additional authentication based on soft-tokens (e.g. SMS token, native mobile applications, etc.) can be required as well before the link is sent over. Second, you should not lock out the users account whilst the process of getting a new password is in progress. This could lead to a Denial of service attack whenever an attacker decides to intentionally lock out the users with an automated attack. Third, whenever the new password request was set in progress, the message you display should be generalized in order to prevent username enumeration. Fourth, always disallow the use of old passwords and implement a strong password policy.

Ensure that password and account policy are implemented

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A

TITLE	DETAILS
Details	Password and account policy in compliance with organizational policy and best practices should be implemented. To defend against brute-force and dictionary based guessing: Strong password policy must be implemented to ensure that users create complex password (e.g., 12 characters minimum length, alphanumeric and special characters). Account lockout policies may be implemented in the following manner: • Soft lock-out: This can be a good option for protecting your users against brute force attacks. For example, whenever the user enters a wrong password three times the application could lock down the account for a minute in order to slow down the process of brute forcing his password making it less profitable for the attacker to proceed. If u were to implement hard lock-out countermeasures for this example you would achieve a "Dos" by permanently locking out accounts. Alternatively, application may generate an OTP (One Time Password) and send it out-of-band (through email, sms etc.) to the user. Another approach may be to implement CAPTCHA after a threshold number of failed attempts is reached. • Hard lock-out: This type of lockout should be applied whenever you detect a user attacking your application and counter him by means of permanently locking out his account until a response team had time to do their forensics. After this process you can decide to give the user back his account or take further legal actions against him. This type of approach prevents the attacker from further penetrating your application and infrastructure. To defend against attacks on default and predictable accounts, verify that all keys and passwords are replaceable, and are generated or replaced after installation time. If the application has to auto-generate passwords, ensure that the generated passwords are random and have high entropy.

Implement controls to prevent username enumeration

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A

TITLE	DETAILS
References	N/A
Steps	All error messages should be generalized in order to prevent username enumeration. Also sometimes you cannot avoid information leaking in functionalities such as a registration page. Here you need to use rate-limiting methods like CAPTCHA to prevent an automated attack by an attacker.

When possible, use Windows Authentication for connecting to SQL Server

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	OnPrem
Attributes	SQL Version - All
References	SQL Server - Choose an Authentication Mode
Steps	Windows Authentication uses Kerberos security protocol, provides password policy enforcement with regard to complexity validation for strong passwords, provides support for account lockout, and supports password expiration.

When possible use Azure Active Directory Authentication for Connecting to SQL Database

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	SQL Azure
Attributes	SQL Version - V12
References	Connecting to SQL Database By Using Azure Active Directory Authentication
Steps	Minimum version: Azure SQL Database V12 required to allow Azure SQL Database to use AAD Authentication against the Microsoft Directory

When SQL authentication mode is used, ensure that account and password are stored securely. **HighRisk**
Ex. 1002 - Page 69

password policy are enforced on SQL server

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	SQL Server password policy
Steps	When using SQL Server Authentication, logins are created in SQL Server that are not based on Windows user accounts. Both the user name and the password are created by using SQL Server and stored in SQL Server. SQL Server can use Windows password policy mechanisms. It can apply the same complexity and expiration policies used in Windows to passwords used inside SQL Server.

Do not use SQL Authentication in contained databases

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	OnPrem, SQL Azure
Attributes	SQL Version - MSSQL2012, SQL Version - V12
References	Security Best Practices with Contained Databases
Steps	The absence of an enforced password policy may increase the likelihood of a weak credential being established in a contained database. Leverage Windows Authentication.

Use per device authentication credentials using SaS tokens

TITLE	DETAILS
Component	Azure Event Hub
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A

TITLE	DETAILS
References	Event Hubs authentication and security model overview
Steps	The Event Hubs security model is based on a combination of Shared Access Signature (SAS) tokens and event publishers. The publisher name represents the DeviceID that receives the token. This would help associate the tokens generated with the respective devices. All messages are tagged with originator on service side allowing detection of in-payload origin spoofing attempts. When authenticating devices, generate a per device SaS token scoped to a unique publisher.

Enable Azure Multi-Factor Authentication for Azure Administrators

TITLE	DETAILS
Component	Azure Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	What is Azure Multi-Factor Authentication?
Steps	Multi-factor authentication (MFA) is a method of authentication that requires more than one verification method and adds a critical second layer of security to user sign-ins and transactions. It works by requiring any two or more of the following verification methods: • Something you know (typically a password) • Something you have (a trusted device that is not easily duplicated, like a phone) • Something you are (biometrics)

Restrict anonymous access to Service Fabric Cluster

TITLE	DETAILS
Component	Service Fabric Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	Environment - Azure
References	Service Fabric cluster security scenarios

TITLE	DETAILS
Steps	Clusters should always be secured to prevent unauthorized users from connecting to your cluster, especially when it has production workloads running on it. While creating a service fabric cluster, ensure that the security mode is set to "secure" and configure the required X.509 server certificate. Creating an "insecure" cluster will allow any anonymous user to connect to it if it exposes management endpoints to the public Internet.

Ensure that Service Fabric client-to-node certificate is different from node-to-node certificate

TITLE	DETAILS
Component	Service Fabric Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	Environment - Azure, Environment - Stand alone
References	Service Fabric Client-to-node certificate security, Connect to a secure cluster using client certificate
Steps	Client-to-node certificate security is configured while creating the cluster either through the Azure portal, Resource Manager templates or a standalone JSON template by specifying an admin client certificate and/or a user client certificate. The admin client and user client certificates you specify should be different than the primary and secondary certificates you specify for Node-to-node security.

Use AAD to authenticate clients to service fabric clusters

TITLE	DETAILS
Component	Service Fabric Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	Environment - Azure
References	Cluster security scenarios - Security Recommendations

TITLE	DETAILS
Steps	Clusters running on Azure can also secure access to the management endpoints using Azure Active Directory (AAD), apart from client certificates. For Azure clusters, it is recommended that you use AAD security to authenticate clients and certificates for node-to-node security.

Ensure that service fabric certificates are obtained from an approved Certificate Authority (CA)

TITLE	DETAILS
Component	Service Fabric Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	Environment - Azure
References	X.509 certificates and Service Fabric
Steps	Service Fabric uses X.509 server certificates for authenticating nodes and clients. Some important things to consider while using certificates in service fabrics: • Certificates used in clusters running production workloads should be created by using a correctly configured Windows Server certificate service or obtained from an approved Certificate Authority (CA). The CA can be an approved external CA or a properly managed internal Public Key Infrastructure (PKI) • Never use any temporary or test certificates in production that are created with tools such as MakeCert.exe • You can use a self-signed certificate, but should only do so for test clusters and not in production

Use standard authentication scenarios supported by Identity Server

TITLE	DETAILS
Component	Identity Server
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	IdentityServer3 - The Big Picture IriusRisk Ex. 1002 - Page 73

TITLE	DETAILS
Steps	Below are the typical interactions supported by Identity Server: • Browsers communicate with web applications • Web applications communicate with web APIs (sometimes on their own, sometimes on behalf of a user) • Browser-based applications communicate with web APIs • Native applications communicate with web APIs • Server-based applications communicate with web APIs • Web APIs communicate with web APIs (sometimes on their own, sometimes on behalf of a user)

Override the default Identity Server token cache with a scalable alternative

TITLE	DETAILS
Component	Identity Server
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	Identity Server Deployment - Caching
Steps	IdentityServer has a simple built-in in-memory cache. While this is good for small scale native apps, it does not scale for mid tier and backend applications for the following reasons: • These applications are accessed by many users at once. Saving all access tokens in the same store creates isolation issues and presents challenges when operating at scale: many users, each with as many tokens as the resources the app accesses on their behalf, can mean huge numbers and very expensive lookup operations • These applications are typically deployed on distributed topologies, where multiple nodes must have access to the same cache • Cached tokens must survive process recycles and deactivations • For all the above reasons, while implementing web apps, it is recommended to override the default Identity Server's token cache with a scalable alternative such as Azure Redis cache

TITLE	DETAILS
Component	Machine Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure that deployed application's binaries are digitally signed so that the integrity of the binaries can be verified

Enable authentication when connecting to MSMQ queues in WCF

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	Generic, .NET Framework 3
Attributes	N/A
References	MSDN
Steps	Program fails to enable authentication when connecting to MSMQ queues, an attacker can anonymously submit messages to the queue for processing. If authentication is not used to connect to an MSMQ queue used to deliver a message to another program, an attacker could submit an anonymous message that is malicious.

Example

The `<netMsmqBinding/>` element of the WCF configuration file below instructs WCF to disable authentication when connecting to an MSMQ queue for message delivery.

```
<bindings>
  <netMsmqBinding>
    <binding>
      <security>
        <transport msmqAuthenticationMode="None" />
      </security>
    </binding>
  </netMsmqBinding>
</bindings>
```

Configure MSMQ to require Windows Domain or Certificate authentication at all times for any incoming or outgoing messages.

Example

The `<netMsmqBinding/>` element of the WCF configuration file below instructs WCF to enable certificate authentication when connecting to an MSMQ queue. The client is authenticated using X.509 certificates. The client certificate must be present in the certificate store of the server.

```
<bindings>
  <netMsmqBinding>
    <binding>
      <security>
        <transport msmqAuthenticationMode=""Certificate"" />
      </security>
    </binding>
  </netMsmqBinding>
</bindings>
```

WCF-Do not set Message clientCredentialType to none

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	.NET Framework 3
Attributes	Client Credential Type - None
References	MSDN, Fortify
Steps	The absence of authentication means everyone is able to access this service. A service that does not authenticate its clients allows access to all users. Configure the application to authenticate against client credentials. This can be done by setting the message clientCredentialType to Windows or Certificate.

Example

```
<message clientCredentialType=""Certificate""/>
```

WCF-Do not set Transport clientCredentialType to none

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	Generic, .NET Framework 3
Attributes	Client Credential Type - None
References	MSDN, Fortify

TITLE	DETAILS
Steps	The absence of authentication means everyone is able to access this service. A service that does not authenticate its clients allows all users to access its functionality. Configure the application to authenticate against client credentials. This can be done by setting the transport clientCredentialType to Windows or Certificate.

Example

```
<transport clientCredentialType=""Certificate""/>
```

Ensure that standard authentication techniques are used to secure Web APIs

TITLE	DETAILS
Component	Web API
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Authentication and Authorization in ASP.NET Web API, External Authentication Services with ASP.NET Web API (C#)
Steps	Authentication is the process where an entity proves its identity, typically through credentials, such as a user name and password. There are multiple authentication protocols available which may be considered. Some of them are listed below: • Client certificates • Windows based • Forms based • Federation - ADFS • Federation - Azure AD • Federation - Identity Server Links in the references section provide low-level details on how each of the authentication schemes can be implemented to secure a Web API.

Use standard authentication scenarios supported by Azure Active Directory

TITLE	DETAILS
Component	Azure AD

TITLE	DETAILS
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Authentication Scenarios for Azure AD, Azure Active Directory Code Samples, Azure Active Directory developer's guide
Steps	Azure Active Directory (Azure AD) simplifies authentication for developers by providing identity as a service, with support for industry-standard protocols such as OAuth 2.0 and OpenID Connect. Below are the five primary application scenarios supported by Azure AD: • Web Browser to Web Application: A user needs to sign in to a web application that is secured by Azure AD • Single Page Application (SPA): A user needs to sign in to a single page application that is secured by Azure AD • Native Application to Web API: A native application that runs on a phone, tablet, or PC needs to authenticate a user to get resources from a web API that is secured by Azure AD • Web Application to Web API: A web application needs to get resources from a web API secured by Azure AD • Daemon or Server Application to Web API: A daemon application or a server application with no web user interface needs to get resources from a web API secured by Azure AD Please refer to the links in the references section for low-level implementation details

Override the default ADAL token cache with a scalable alternative

TITLE	DETAILS
Component	Azure AD
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Modern Authentication with Azure Active Directory for Web Applications, Using Redis as ADAL token cache

TITLE	DETAILS
Steps	The default cache that ADAL (Active Directory Authentication Library) uses is an in-memory cache that relies on a static store, available process-wide. While this works for native applications, it does not scale for mid tier and backend applications for the following reasons: • These applications are accessed by many users at once. Saving all access tokens in the same store creates isolation issues and presents challenges when operating at scale: many users, each with as many tokens as the resources the app accesses on their behalf, can mean huge numbers and very expensive lookup operations • These applications are typically deployed on distributed topologies, where multiple nodes must have access to the same cache • Cached tokens must survive process recycles and deactivations For all the above reasons, while implementing web apps, it is recommended to override the default ADAL token cache with a scalable alternative such as Azure Redis cache.

Ensure that TokenReplayCache is used to prevent the replay of ADAL authentication tokens

TITLE	DETAILS
Component	Azure AD
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Modern Authentication with Azure Active Directory for Web Applications

TITLE	DETAILS
Steps	The TokenReplayCache property allows developers to define a token replay cache, a store that can be used for saving tokens for the purpose of verifying that no token can be used more than once. This is a measure against a common attack, the aptly called token replay attack: an attacker intercepting the token sent at sign-in might try to send it to the app again ("replay" it) for establishing a new session. E.g., In OIDC code-grant flow, after successful user authentication, a request to "/signin-oidc" endpoint of the relying party is made with "id_token", "code" and "state" parameters. The relying party validates this request and establishes a new session. If an adversary captures this request and replays it, he/she can establish a successful session and spoof the user. The presence of the nonce in OpenID Connect can limit but not fully eliminate the circumstances in which the attack can be successfully enacted. To protect their applications, developers can provide an implementation of ITokenReplayCache and assign an instance to TokenReplayCache.

Example

```
// ITokenReplayCache defined in ADAL
public interface ITokenReplayCache
{
    bool TryAdd(string securityToken, DateTime expiresOn);
    bool TryFind(string securityToken);
}
```

Example

Here is a sample implementation of the ITokenReplayCache interface. (Please customize and implement your project-specific caching framework)

```
public class TokenReplayCache : ITokenReplayCache
{
    private readonly ICacheProvider cache; // Your project-specific cache provider
    public TokenReplayCache(ICacheProvider cache)
    {
        this.cache = cache;
    }
    public bool TryAdd(string securityToken, DateTime expiresOn)
    {
        if (this.cache.Get<string>(securityToken) == null)
        {
            this.cache.Set(securityToken, securityToken);
            return true;
        }
        return false;
    }
    public bool TryFind(string securityToken)
    {
        return this.cache.Get<string>(securityToken) != null;
    }
}
```

follows.

```
OpenIdConnectOptions openIdConnectOptions = new OpenIdConnectOptions
{
    AutomaticAuthenticate = true,
    ... // other configuration properties follow..
    TokenValidationParameters = new TokenValidationParameters
    {
        TokenReplayCache = new TokenReplayCache(/*Inject your cache provider*/);
    }
}
```

Please note that to test the effectiveness of this configuration, login into your local OIDC-protected application and capture the request to `"/signin-oidc"` endpoint in fiddler. When the protection is not in place, replaying this request in fiddler will set a new session cookie. When the request is replayed after the TokenReplayCache protection is added, the application will throw an exception as follows:

```
SecurityTokenReplayDetectedException: IDX10228: The securityToken has previously been validated, securityToken:
'eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIngldCI6Ikp1eQ19wMmNBVGZNNXBpWlKSE1iYTlhb0VLWSIsImtpZCI6Ikp1eQ1.....
```

Use ADAL libraries to manage token requests from OAuth2 clients to AAD (or on-premises AD)

TITLE	DETAILS
Component	Azure AD
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	ADAL
Steps	The Azure AD authentication Library (ADAL) enables client application developers to easily authenticate users to cloud or on-premises Active Directory (AD), and then obtain access tokens for securing API calls. ADAL has many features that make authentication easier for developers, such as asynchronous support, a configurable token cache that stores access tokens and refresh tokens, automatic token refresh when an access token expires and a refresh token is available, and more. By handling most of the complexity, ADAL can help a developer focus on business logic in their application and easily secure resources without being an expert on security. Separate libraries are available for .NET, JavaScript (client and Node.js), iOS, Android and Java.

Authenticate devices connecting to the Field Gateway

TITLE	DETAILS
Component	IoT Field Gateway

TITLE	DETAILS
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure that each device is authenticated by the Field Gateway before accepting data from them and before facilitating upstream communications with the Cloud Gateway. Also, ensure that devices connect with a per device credential so that individual devices can be uniquely identified.

Ensure that devices connecting to Cloud gateway are authenticated

TITLE	DETAILS
Component	IoT Cloud Gateway
SDL Phase	Build
Applicable Technologies	Generic, C#, NodeJS,
Attributes	N/A, Gateway choice - Azure IoT Hub
References	N/A, Azure IoT hub with .NET, Getting Started with IoT hub and Node JS, Securing IoT with SAS and certificates, Git repository
Steps	• Generic: Authenticate the device using Transport Layer Security (TLS) or IPSec. Infrastructure should support using pre-shared key (PSK) on those devices that cannot handle full asymmetric cryptography. Leverage Azure AD, OAuth. • C#: When creating a DeviceClient instance, by default, the Create method creates a DeviceClient instance that uses the AMQP protocol to communicate with IoT Hub. To use the HTTPS protocol, use the override of the Create method that enables you to specify the protocol. If you use the HTTPS protocol, you should also add the <code>Microsoft.AspNet.WebApi.Client</code> NuGet package to your project to include the <code>System.Net.Http.Formatting</code> namespace.

Example

```

static DeviceClient deviceClient;

static string deviceKey = "{device key}";
static string iotHubUri = "{iot hub hostname}";

var messageString = "{message in string format}";
var message = new Message(Encoding.ASCII.GetBytes(messageString));

deviceClient = DeviceClient.Create(iotHubUri, new DeviceAuthenticationWithRegistrySymmetricKey("myFirstDevice",
deviceKey));

await deviceClient.SendEventAsync(message);

```

Example

Node.JS: Authentication

Symmetric key

- Create a IoT hub on azure
- Create an entry in the device identity registry

```
javascript var device = new iothub.Device(null); device.deviceId = <DeviceId> registry.create(device,
function(err, deviceInfo, res) {})
```

- Create a simulated device

```
javascript var clientFromConnectionString = require('azure-iot-device-amqp').clientFromConnectionString; var
Message = require('azure-iot-device').Message; var connectionString = 'HostName=<HostName>DeviceId=
<DeviceId>SharedAccessKey=<SharedAccessKey>'; var client = clientFromConnectionString(connectionString);
```

SAS Token

- Gets internally generated when using symmetric key but we can generate and use it explicitly as well
- Define a protocol : `var Http = require('azure-iot-device-http').Http;`
- Create a sas token : `""javascript resourceUri =
encodeURIComponent(resourceUri.toLowerCase()).toLowerCase(); var deviceName = ""; var expires =
(Date.now() / 1000) + expiresInMins * 60; var toSign = resourceUri + '\n' + expires; // using crypto var
decodedPassword = new Buffer(signingKey, 'base64').toString('binary'); const hmac =
crypto.createHmac('sha256', decodedPassword); hmac.update(toSign); var base64signature =
hmac.digest('base64'); var base64UriEncoded = encodeURIComponent(base64signature); // construct
authorization string var token = "SharedAccessSignature sr=" + resourceUri +
"%2fdevices%2f"+deviceName+"&sig="

 - base64UriEncoded + "&se=" + expires; if (policyName) token += "&skn="+policyName; return token; ""`
- Connect using sas token: `javascript Client.fromSharedAccessSignature(sas, Http);` #### Certificates
- Generate a self signed X509 certificate using any tool such as OpenSSL to generate a .cert and .key files to store the certificate and the key respectively
- Provision a device that accepts secured connection using certificates.

```
javascript var connectionString = '<connectionString>'; var registry =
iothub.Registry.fromConnectionString(connectionString); var deviceJSON = {deviceId:"<deviceId>",
authentication: { x509Thumbprint: { primaryThumbprint: "<PrimaryThumbprint>", secondaryThumbprint: "
<SecondaryThumbprint>" } }} var device = deviceJSON; registry.create(device, function (err) {});
```

- Connect a device using a certificate

```
javascript var Protocol = require('azure-iot-device-http').Http; var Client = require('azure-iot-
device').Client; var connectionString = 'HostName=<HostName>DeviceId=<DeviceId>x509=true'; var client =
Client.fromConnectionString(connectionString, Protocol); var options = { key: fs.readFileSync('./key.pem',
'utf8'), cert: fs.readFileSync('./server.crt', 'utf8') }; // Calling setOptions with the x509 certificate and
key (and optionally, passphrase) will configure the client //transport to use x509 when connecting to IoT Hub
client.setOptions(options); //call fn to execute after the connection is set up client.open(fn);
```

Use per-device authentication credentials

TITLE	DETAILS
Component	IoT Cloud Gateway
SDL Phase	Build
Applicable Technologies	Generic
Attributes	Gateway choice - Azure IoT Hub
References	Azure IoT Hub Security Tokens
Steps	Use per device authentication credentials using SaS tokens based on Device key or Client Certificate, instead of IoT Hub-level shared access policies. This prevents the reuse of authentication tokens of one device or field gateway by another

Ensure that only the required containers and blobs are given anonymous read access

TITLE	DETAILS
Component	Azure Storage
SDL Phase	Build
Applicable Technologies	Generic
Attributes	StorageType - Blob
References	Manage anonymous read access to containers and blobs, Shared Access Signatures, Part 1: Understanding the SAS model

TITLE	DETAILS
Steps	By default, a container and any blobs within it may be accessed only by the owner of the storage account. To give anonymous users read permissions to a container and its blobs, one can set the container permissions to allow public access. Anonymous users can read blobs within a publicly accessible container without authenticating the request. Containers provide the following options for managing container access: • Full public read access: Container and blob data can be read via anonymous request. Clients can enumerate blobs within the container via anonymous request, but cannot enumerate containers within the storage account. • Public read access for blobs only: Blob data within this container can be read via anonymous request, but container data is not available. Clients cannot enumerate blobs within the container via anonymous request • No public read access: Container and blob data can be read by the account owner only Anonymous access is best for scenarios where certain blobs should always be available for anonymous read access. For finer-grained control, one can create a shared access signature, which enables to delegate restricted access using different permissions and over a specified time interval. Ensure that containers and blobs, which may potentially contain sensitive data, are not given anonymous access accidentally

Grant limited access to objects in Azure storage using SAS or SAP

TITLE	DETAILS
Component	Azure Storage
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Shared Access Signatures, Part 1: Understanding the SAS model, Shared Access Signatures, Part 2: Create and use a SAS with Blob storage, How to delegate access to objects in your account using Shared Access Signatures and Stored Access Policies

TITLE	DETAILS
Steps	Using a shared access signature (SAS) is a powerful way to grant limited access to objects in a storage account to other clients, without having to expose account access key. The SAS is a URI that encompasses in its query parameters all of the information necessary for authenticated access to a storage resource. To access storage resources with the SAS, the client only needs to pass in the SAS to the appropriate constructor or method. You can use a SAS when you want to provide access to resources in your storage account to a client that can't be trusted with the account key. Your storage account keys include both a primary and secondary key, both of which grant administrative access to your account and all of the resources in it. Exposing either of your account keys opens your account to the possibility of malicious or negligent use. Shared access signatures provide a safe alternative that allows other clients to read, write, and delete data in your storage account according to the permissions you've granted, and without need for the account key. If you have a logical set of parameters that are similar each time, using a Stored Access Policy (SAP) is a better idea. Because using a SAS derived from a Stored Access Policy gives you the ability to revoke that SAS immediately, it is the recommended best practice to always use Stored Access Policies when possible.

Security Frame: Authorization | Mitigations

1/24/2018 • 17 min to read • [Edit Online](#)

PRODUCT/SERVICE	ARTICLE
Machine Trust Boundary	• Ensure that proper ACLs are configured to restrict unauthorized access to data on the device • Ensure that sensitive user-specific application content is stored in user-profile directory • Ensure that the deployed applications are run with least privileges
Web Application	• Enforce sequential step order when processing business logic flows • Implement rate limiting mechanism to prevent enumeration • Ensure that proper authorization is in place and principle of least privileges is followed • Business logic and resource access authorization decisions should not be based on incoming request parameters • Ensure that content and resources are not enumerable or accessible via forceful browsing
Database	• Ensure that least-privileged accounts are used to connect to Database server • Implement Row Level Security RLS to prevent tenants from accessing each other's data • Sysadmin role should only have valid necessary users
IoT Cloud Gateway	• Connect to Cloud Gateway using least-privileged tokens
Azure Event Hub	• Use a send-only permissions SAS Key for generating device tokens • Do not use access tokens that provide direct access to the Event Hub • Connect to Event Hub using SAS keys that have the minimum permissions required
Azure Document DB	• Use resource tokens to connect to Azure Cosmos DB whenever possible
Azure Trust Boundary	• Enable fine-grained access management to Azure Subscription using RBAC
Service Fabric Trust Boundary	• Restrict client's access to cluster operations using RBAC

PRODUCT/SERVICE	ARTICLE
Dynamics CRM	Perform security modeling and use Field Level Security where required
Dynamics CRM Portal	Perform security modeling of portal accounts keeping in mind that the security model for the portal differs from the rest of CRM
Azure Storage	- Grant fine-grained permission on a range of entities in Azure Table Storage - Enable Role-Based Access Control (RBAC) to Azure storage account using Azure Resource Manager
Mobile Client	Implement implicit jailbreak or rooting detection
WCF	- Weak Class Reference in WCF - WCF-Implement Authorization control
Web API	Implement proper authorization mechanism in ASP.NET Web API
IoT Device	Perform authorization checks in the device if it supports various actions that require different permission levels
IoT Field Gateway	Perform authorization checks in the Field Gateway if it supports various actions that require different permission levels

Ensure that proper ACLs are configured to restrict unauthorized access to data on the device

TITLE	DETAILS
Component	Machine Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure that proper ACLs are configured to restrict unauthorized access to data on the device

Ensure that sensitive user-specific application content is stored in user-profile directory

TITLE	DETAILS
Component	Machine Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure that sensitive user-specific application content is stored in user-profile directory. This is to prevent multiple users of the machine from accessing each other's data.

Ensure that the deployed applications are run with least privileges

TITLE	DETAILS
Component	Machine Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure that the deployed application is run with least privileges.

Enforce sequential step order when processing business logic flows

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A

TITLE	DETAILS
Steps	In order to verify that this stage was run through by a genuine user you want to enforce the application to only process business logic flows in sequential step order, with all steps being processed in realistic human time, and not process out of order, skipped steps, processed steps from another user, or too quickly submitted transactions.

Implement rate limiting mechanism to prevent enumeration

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure that sensitive identifiers are random. Implement CAPTCHA control on anonymous pages. Ensure that error and exception should not reveal specific data

Ensure that proper authorization is in place and principle of least privileges is followed

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A

TITLE	DETAILS
Steps	The principle means giving a user account only those privileges which are essential to that users work. For example, a backup user does not need to install software: hence, the backup user has rights only to run backup and backup-related applications. Any other privileges, such as installing new software, are blocked. The principle applies also to a personal computer user who usually does work in a normal user account, and opens a privileged, password protected account (that is, a superuser) only when the situation absolutely demands it. This principle can also be applied to your web-applications. Instead of solely depending on role-based authentication methods using sessions, we rather want to assign privileges to users by means of a Database-Based Authentication system. We still use sessions in order to identify if the user was logged in correctly, only now instead of assigning that user with a specific role we assign him with privileges to verify which actions he is privileged to perform on the system. Also a big pro of this method is, whenever a user has to be assigned fewer privileges your changes will be applied on the fly since the assigning does not depend on the session which otherwise had to expire first.

Business logic and resource access authorization decisions should not be based on incoming request parameters

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Whenever you are checking whether a user is restricted to review certain data, the access restrictions should be processed server-side. The userID should be stored inside of a session variable on login and should be used to retrieve user data from the database

Example

```
SELECT data
FROM personaldata
WHERE userID=:id < - session var
```

Now an possible attacker can not tamper and change the application operation since the identifier for retrieving the data is handled server-side.

Ensure that content and resources are not enumerable or accessible via forceful browsing

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Sensitive static and configuration files should not be kept in the web-root. For content not required to be public, either proper access controls should be applied or removal of the content itself. Also, forceful browsing is usually combined with Brute Force techniques to gather information by attempting to access as many URLs as possible to enumerate directories and files on a server. Attackers may check for all variations of commonly existing files. For example, a password file search would encompass files including psswd.txt, password.htm, password.dat, and other variations. To mitigate this, capabilities for detection of brute force attempts should be included.

Ensure that least-privileged accounts are used to connect to Database server

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	SQL Database permissions hierarchy, SQL database securables
Steps	Least-privileged accounts should be used to connect to the database. Application login should be restricted in the database and should only execute selected stored procedures. Application's login should have no direct table access.

Implement Row Level Security RLS to prevent tenants from accessing each other's data

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	Sql Azure, OnPrem
Attributes	SQL Version - V12, SQL Version - MsSQL2016
References	SQL Server Row-Level Security (RLS)
Steps	Row-Level Security enables customers to control access to rows in a database table based on the characteristics of the user executing a query (e.g., group membership or execution context). Row-Level Security (RLS) simplifies the design and coding of security in your application. RLS enables you to implement restrictions on data row access. For example ensuring that workers can access only those data rows that are pertinent to their department, or restricting a customer's data access to only the data relevant to their company. The access restriction logic is located in the database tier rather than away from the data in another application tier. The database system applies the access restrictions every time that data access is attempted from any tier. This makes the security system more reliable and robust by reducing the surface area of the security system.

Please note that RLS as an out-of-the-box database feature is applicable only to SQL Server starting 2016 and Azure SQL database. If the out-of-the-box RLS feature is not implemented, it should be ensured that data access is restricted Using Views and Procedures

Sysadmin role should only have valid necessary users

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	SQL Database permissions hierarchy, SQL database securables
Steps	Members of the SysAdmin fixed server role should be very limited and never contain accounts used by applications. Please review the list of users in the role and remove any unnecessary accounts

Connect to Cloud Gateway using least-privileged tokens

TITLE	DETAILS
Component	IoT Cloud Gateway
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	Gateway choice - Azure IoT Hub
References	IoT Hub Access Control
Steps	Provide least privilege permissions to various components that connect to Cloud Gateway (IoT Hub). Typical example is -- Device management/provisioning component uses registryread/write, Event Processor (ASA) uses Service Connect. Individual devices connect using Device credentials

Use a send-only permissions SAS Key for generating device tokens

TITLE	DETAILS
Component	Azure Event Hub
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Event Hubs authentication and security model overview
Steps	A SAS key is used to generate individual device tokens. Use a send-only permissions SAS key while generating the device token for a given publisher

Do not use access tokens that provide direct access to the Event Hub

TITLE	DETAILS
Component	Azure Event Hub
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Event Hubs authentication and security model overview

TITLE	DETAILS
Steps	A token that grants direct access to the event hub should not be given to the device. Using a least privileged token for the device that gives access only to a publisher would help identify and blacklist it if found to be a rogue or compromised device.

Connect to Event Hub using SAS keys that have the minimum permissions required

TITLE	DETAILS
Component	Azure Event Hub
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Event Hubs authentication and security model overview
Steps	Provide least privilege permissions to various back-end applications that connect to the Event Hub. Generate separate SAS keys for each back-end application and only provide the required permissions - Send, Receive or Manage to them.

Use resource tokens to connect to Cosmos DB whenever possible

TITLE	DETAILS
Component	Azure Document DB
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	A resource token is associated with an Azure Cosmos DB permission resource and captures the relationship between the user of a database and the permission that user has for a specific Azure Cosmos DB application resource (e.g. collection, document). Always use a resource token to access the Azure Cosmos DB if the client cannot be trusted with handling master or read-only keys - like an end user application like a mobile or desktop client. Use Master key or read-only keys from backend applications which can store these keys securely.

Enable fine-grained access management to Azure Subscription using

RBAC

TITLE	DETAILS
Component	Azure Trust Boundary
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Use role assignments to manage access to your Azure subscription resources
Steps	Azure Role-Based Access Control (RBAC) enables fine-grained access management for Azure. Using RBAC, you can grant only the amount of access that users need to perform their jobs.

Restrict client's access to cluster operations using RBAC

TITLE	DETAILS
Component	Service Fabric Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	Environment - Azure
References	Role-based access control for Service Fabric clients
Steps	Azure Service Fabric supports two different access control types for clients that are connected to a Service Fabric cluster: administrator and user. Access control allows the cluster administrator to limit access to certain cluster operations for different groups of users, making the cluster more secure. Administrators have full access to management capabilities (including read/write capabilities). Users, by default, have only read access to management capabilities (for example, query capabilities), and the ability to resolve applications and services. You specify the two client roles (administrator and client) at the time of cluster creation by providing separate certificates for each.

Perform security modeling and use Field Level Security where required

TITLE	DETAILS
Component	Dynamics CRM
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Perform security modeling and use Field Level Security where required

Perform security modeling of portal accounts keeping in mind that the security model for the portal differs from the rest of CRM

TITLE	DETAILS
Component	Dynamics CRM Portal
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Perform security modeling of portal accounts keeping in mind that the security model for the portal differs from the rest of CRM

Grant fine-grained permission on a range of entities in Azure Table Storage

TITLE	DETAILS
Component	Azure Storage
SDL Phase	Build
Applicable Technologies	Generic
Attributes	StorageType - Table
References	How to delegate access to objects in your Azure storage account using SAS

TITLE	DETAILS
Steps	In certain business scenarios, Azure Table Storage may be required to store sensitive data that caters to different parties. E.g., sensitive data pertaining to different countries. In such cases, SAS signatures can be constructed by specifying the partition and row key ranges, such that a user can access data specific to a particular country.

Enable Role-Based Access Control (RBAC) to Azure storage account using Azure Resource Manager

TITLE	DETAILS
Component	Azure Storage
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	How to secure your storage account with Role-Based Access Control (RBAC)
Steps	When you create a new storage account, you select a deployment model of Classic or Azure Resource Manager. The Classic model of creating resources in Azure only allows all-or-nothing access to the subscription, and in turn, the storage account. With the Azure Resource Manager model, you put the storage account in a resource group and control access to the management plane of that specific storage account using Azure Active Directory. For example, you can give specific users the ability to access the storage account keys, while other users can view information about the storage account, but cannot access the storage account keys.

Implement implicit jailbreak or rooting detection

TITLE	DETAILS
Component	Mobile Client
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A

TITLE	DETAILS
Steps	Application should safeguard its own configuration and user data in case if phone is rooted or jail broken. Rooting/jail breaking implies unauthorized access, which normal users won't do on their own phones. Therefore application should have implicit detection logic on application startup, to detect if the phone has been rooted. The detection logic can be simply accessing files which normally only root user can access, for example: • /system/app/Superuser.apk • /sbin/su • /system/bin/su • /system/sbin/su • /data/local/sbin/su • /data/local/bin/su • /system/sd/sbin/su • /system/bin/failsafe/su • /data/local/su If the application can access any of these files, it denotes that the application is running as root user.

Weak Class Reference in WCF

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	Generic, NET Framework 3
Attributes	N/A
References	MSDN, Fortify Kingdom

TITLE	DETAILS
Steps	The system uses a weak class reference, which might allow an attacker to execute unauthorized code. The program references a user-defined class that is not uniquely identified. When .NET loads this weakly identified class, the CLR type loader searches for the class in the following locations in the specified order: 1. If the assembly of the type is known, the loader searches the configuration file's redirect locations, GAC, the current assembly using configuration information, and the application base directory 2. If the assembly is unknown, the loader searches the current assembly, mscorlib, and the location returned by the TypeResolve event handler 3. This CLR search order can be modified with hooks such as the Type Forwarding mechanism and the AppDomain.TypeResolve event If an attacker exploits the CLR search order by creating an alternative class with the same name and placing it in an alternative location that the CLR will load first, the CLR will unintentionally execute the attacker-supplied code

Example

The `<behaviorExtensions/>` element of the WCF configuration file below instructs WCF to add a custom behavior class to a particular WCF extension.

```
<system.serviceModel>
  <extensions>
    <behaviorExtensions>
      <add name="myBehavior" type="MyBehavior" />
    </behaviorExtensions>
  </extensions>
</system.serviceModel>
```

Using fully qualified (strong) names uniquely identifies a type and further increases security of your system. Use fully qualified assembly names when registering types in the machine.config and app.config files.

Example

The `<behaviorExtensions/>` element of the WCF configuration file below instructs WCF to add strongly-referenced custom behavior class to a particular WCF extension.

```
<system.serviceModel>
  <extensions>
    <behaviorExtensions>
      <add name="myBehavior" type="Microsoft.ServiceModel.Samples.MyBehaviorSection, MyBehavior,
        Version=1.0.0.0, Culture=neutral, PublicKeyToken=null" />
    </behaviorExtensions>
  </extensions>
</system.serviceModel>
```

WCF-Implement Authorization control

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	Generic, NET Framework 3
Attributes	N/A
References	MSDN, Fortify Kingdom
Steps	This service does not use an authorization control. When a client calls a particular WCF service, WCF provides various authorization schemes that verify that the caller has permission to execute the service method on the server. If authorization controls are not enabled for WCF services, an authenticated user can achieve privilege escalation.

Example

The following configuration instructs WCF to not check the authorization level of the client when executing the service:

```
<behaviors>
  <serviceBehaviors>
    <behavior>
      ...
      <serviceAuthorization principalPermissionMode=""None"" />
    </behavior>
  </serviceBehaviors>
</behaviors>
```

Use a service authorization scheme to verify that the caller of the service method is authorized to do so. WCF provides two modes and allows the definition of a custom authorization scheme. The UseWindowsGroups mode uses Windows roles and users and the UseAspNetRoles mode uses an ASP.NET role provider, such as SQL Server, to authenticate.

Example

The following configuration instructs WCF to make sure that the client is part of the Administrators group before executing the Add service:

```
<behaviors>
  <serviceBehaviors>
    <behavior>
      ...
      <serviceAuthorization principalPermissionMode=""UseWindowsGroups"" />
    </behavior>
  </serviceBehaviors>
</behaviors>
```

The service is then declared as the following:

```
[PrincipalPermission(SecurityAction.Demand,
Role = ""Builtin\Administrators"")]
public double Add(double n1, double n2)
{
double result = n1 + n2;
return result;
}
```

Implement proper authorization mechanism in ASP.NET Web API

TITLE	DETAILS
Component	Web API
SDL Phase	Build
Applicable Technologies	Generic, MVC5
Attributes	N/A, Identity Provider - ADFS, Identity Provider - Azure AD
References	Authentication and Authorization in ASP.NET Web API
Steps	Role information for the application users can be derived from Azure AD or ADFS claims if the application relies on them as Identity provider or the application itself might provided it. In any of these cases, the custom authorization implementation should validate the user role information. Role information for the application users can be derived from Azure AD or ADFS claims if the application relies on them as Identity provider or the application itself might provided it. In any of these cases, the custom authorization implementation should validate the user role information.

Example

```

[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, Inherited = true, AllowMultiple = true)]
public class ApiAuthorizeAttribute : System.Web.Http.AuthorizeAttribute
{
    public async override Task OnAuthorizationAsync(HttpContext actionContext, CancellationToken
cancellation_token)
    {
        if (actionContext == null)
        {
            throw new Exception();
        }

        if (!string.IsNullOrEmpty(base.Roles))
        {
            bool isAuthorized = ValidateRoles(actionContext);
            if (!isAuthorized)
            {
                HandleUnauthorizedRequest(actionContext);
            }
        }

        base.OnAuthorization(actionContext);
    }

    public bool ValidateRoles(HttpContext actionContext)
    {
        //Authorization logic here; returns true or false
    }
}

```

All the controllers and action methods which needs to protected should be decorated with above attribute.

```

[ApiAuthorize]
public class CustomController : ApiController
{
    //Application code goes here
}

```

Perform authorization checks in the device if it supports various actions that require different permission levels

TITLE	DETAILS
Component	IoT Device
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A

TITLE	DETAILS
Steps	The Device should authorize the caller to check if the caller has the required permissions to perform the action requested. For e.g. Lets say the device is a Smart Door Lock that can be monitored from the cloud, plus it provides functionalities like Remotely locking the door. The Smart Door Lock provides unlocking functionality only when someone physically comes near the door with a Card. In this case, the implementation of the remote command and control should be done in such a way that it does not provide any functionality to unlock the door as the cloud gateway is not authorized to send a command to unlock the door.

Perform authorization checks in the Field Gateway if it supports various actions that require different permission levels

TITLE	DETAILS
Component	IoT Field Gateway
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	The Field Gateway should authorize the caller to check if the caller has the required permissions to perform the action requested. For e.g. there should be different permissions for an admin user interface/API used to configure a field gateway v/s devices that connect to it.

Security Frame: Communication Security | Mitigations

1/24/2018 • 14 min to read • [Edit Online](#)

PRODUCT/SERVICE	ARTICLE
Azure Event Hub	Secure communication to Event Hub using SSL/TLS
Dynamics CRM	Check service account privileges and check that the custom Services or ASP.NET Pages respect CRM's security
Azure Data Factory	Use Data management gateway while connecting On Prem SQL Server to Azure Data Factory
Identity Server	Ensure that all traffic to Identity Server is over HTTPS connection
Web Application	- Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections - Configure SSL certificate for custom domain in Azure App Service - Force all traffic to Azure App Service over HTTPS connection - Enable HTTP Strict Transport Security (HSTS)
Database	- Ensure SQL server connection encryption and certificate validation - Force Encrypted communication to SQL server
Azure Storage	- Ensure that communication to Azure Storage is over HTTPS - Validate MD5 hash after downloading blob if HTTPS cannot be enabled - Use SMB 3.0 compatible client to ensure in-transit data encryption to Azure File Shares
Mobile Client	Implement Certificate Pinning
WCF	- Enable HTTPS - Secure Transport channel - WCF: Set Message security Protection level to EncryptAndSign - WCF: Use a least-privileged account to run your WCF service
Web API	Force all traffic to Web APIs over HTTPS connection

PRODUCT/SERVICE	ARTICLE
Azure Redis Cache	Ensure that communication to Azure Redis Cache is over SSL
IoT Field Gateway	Secure Device to Field Gateway communication
IoT Cloud Gateway	Secure Device to Cloud Gateway communication using SSL/TLS

Secure communication to Event Hub using SSL/TLS

TITLE	DETAILS
Component	Azure Event Hub
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Event Hubs authentication and security model overview
Steps	Secure AMQP or HTTP connections to Event Hub using SSL/TLS

Check service account privileges and check that the custom Services or ASP.NET Pages respect CRM's security

TITLE	DETAILS
Component	Dynamics CRM
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Check service account privileges and check that the custom Services or ASP.NET Pages respect CRM's security

Use Data management gateway while connecting On Prem SQL Server to Azure Data Factory

TITLE	DETAILS
Component	Azure Data Factory
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	Linked Service Types - Azure and On Prem
References	Moving data between On Prem and Azure Data Factory, Data management gateway
Steps	The Data Management Gateway (DMG) tool is required to connect to data sources which are protected behind corpnet or a firewall. 1. Locking down the machine isolates the DMG tool and prevents malfunctioning programs from damaging or snooping on the data source machine. (E.g. latest updates must be installed, enable minimum required ports, controlled accounts provisioning, auditing enabled, disk encryption enabled etc.) 2. Data Gateway key must be rotated at frequent intervals or whenever the DMG service account password renews 3. Data transits through Link Service must be encrypted

Ensure that all traffic to Identity Server is over HTTPS connection

TITLE	DETAILS
Component	Identity Server
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	IdentityServer3 - Keys, Signatures and Cryptography, IdentityServer3 - Deployment
Steps	By default, IdentityServer requires all incoming connections to come over HTTPS. It is absolutely mandatory that communication with IdentityServer is done over secured transports only. There are certain deployment scenarios like SSL offloading where this requirement can be relaxed. See the Identity Server deployment page in the references for more information.

Verify X.509 certificates used to authenticate SSL, TLS, and DTLS connections

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Applications that use SSL, TLS, or DTLS must fully verify the X.509 certificates of the entities they connect to. This includes verification of the certificates for: • Domain name • Validity dates (both beginning and expiration dates) • Revocation status • Usage (for example, Server Authentication for servers, Client Authentication for clients) • Trust chain. Certificates must chain to a root certification authority (CA) that is trusted by the platform or explicitly configured by the administrator • Key length of certificate's public key must be >2048 bits • Hashing algorithm must be SHA256 and above

Configure SSL certificate for custom domain in Azure App Service

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	EnvironmentType - Azure
References	Enable HTTPS for an app in Azure App Service
Steps	By default, Azure already enables HTTPS for every app with a wildcard certificate for the *.azurewebsites.net domain. However, like all wildcard domains, it is not as secure as using a custom domain with own certificate Refer. It is recommended to enable SSL for the custom domain which the deployed app will be accessed through

Force all traffic to Azure App Service over HTTPS connection

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	EnvironmentType - Azure
References	Enforce HTTPS on Azure App Service
Steps	Though Azure already enables HTTPS for Azure app services with a wildcard certificate for the domain *.azurewebsites.net, it do not enforce HTTPS. Visitors may still access the app using HTTP, which may compromise the app's security and hence HTTPS has to be enforced explicitly. ASP.NET MVC applications should use the RequireHttps filter that forces an unsecured HTTP request to be re-sent over HTTPS. Alternatively, the URL Rewrite module, which is included with Azure App Service can be used to enforce HTTPS. URL Rewrite module enables developers to define rules that are applied to incoming requests before the requests are handed to your application. URL Rewrite rules are defined in a web.config file stored in the root of the application

Example

The following example contains a basic URL Rewrite rule that forces all incoming traffic to use HTTPS

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <rewrite>
      <rules>
        <rule name="Force HTTPS" enabled="true">
          <match url="(.*)" ignoreCase="false" />
          <conditions>
            <add input="{HTTPS}" pattern="off" />
          </conditions>
          <action type="Redirect" url="https://{HTTP_HOST}/{R:1}" appendQueryString="true"
redirectType="Permanent" />
        </rule>
      </rules>
    </rewrite>
  </system.webServer>
</configuration>
```

This rule works by returning an HTTP status code of 301 (permanent redirect) when the user requests a page using HTTP. The 301 redirects the request to the same URL as the visitor requested, but replaces the HTTP portion of the request with HTTPS. For example, HTTP://contoso.com would be redirected to HTTPS://contoso.com.

Enable HTTP Strict Transport Security (HSTS)

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	OWASP HTTP Strict Transport Security Cheat Sheet
Steps	HTTP Strict Transport Security (HSTS) is an opt-in security enhancement that is specified by a web application through the use of a special response header. Once a supported browser receives this header that browser will prevent any communications from being sent over HTTP to the specified domain and will instead send all communications over HTTPS. It also prevents HTTPS click through prompts on browsers. To implement HSTS, the following response header has to be configured for a website globally, either in code or in config. Strict-Transport-Security: max-age=300; includeSubDomains HSTS addresses the following threats: • User bookmarks or manually types <code>http://example.com</code> and is subject to a man-in-the-middle attacker: HSTS automatically redirects HTTP requests to HTTPS for the target domain • Web application that is intended to be purely HTTPS inadvertently contains HTTP links or serves content over HTTP: HSTS automatically redirects HTTP requests to HTTPS for the target domain • A man-in-the-middle attacker attempts to intercept traffic from a victim user using an invalid certificate and hopes the user will accept the bad certificate: HSTS does not allow a user to override the invalid certificate message

Ensure SQL server connection encryption and certificate validation

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	SQL Azure
Attributes	SQL Version - V12
References	Best Practices on Writing Secure Connection Strings for SQL Database

TITLE	DETAILS
Steps	All communications between SQL Database and a client application are encrypted using Secure Sockets Layer (SSL) at all times. SQL Database doesn't support unencrypted connections. To validate certificates with application code or tools, explicitly request an encrypted connection and do not trust the server certificates. If your application code or tools do not request an encrypted connection, they will still receive encrypted connections However, they may not validate the server certificates and thus will be susceptible to "man in the middle" attacks. To validate certificates with ADO.NET application code, set <code>Encrypt=True</code> and <code>TrustServerCertificate=False</code> in the database connection string. To validate certificates via SQL Server Management Studio, open the Connect to Server dialog box. Click Encrypt connection on the Connection Properties tab

Force Encrypted communication to SQL server

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	OnPrem
Attributes	SQL Version - MsSQL2016, SQL Version - MsSQL2012, SQL Version - MsSQL2014
References	Enable Encrypted Connections to the Database Engine
Steps	Enabling SSL encryption increases the security of data transmitted across networks between instances of SQL Server and applications.

Ensure that communication to Azure Storage is over HTTPS

TITLE	DETAILS
Component	Azure Storage
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	Azure Storage Transport-Level Encryption -- Using HTTPS

TITLE	DETAILS
Steps	To ensure the security of Azure Storage data in-transit, always use the HTTPS protocol when calling the REST APIs or accessing objects in storage. Also, Shared Access Signatures, which can be used to delegate access to Azure Storage objects, include an option to specify that only the HTTPS protocol can be used when using Shared Access Signatures, ensuring that anybody sending out links with SAS tokens will use the proper protocol.

Validate MD5 hash after downloading blob if HTTPS cannot be enabled

TITLE	DETAILS
Component	Azure Storage
SDL Phase	Build
Applicable Technologies	Generic
Attributes	StorageType - Blob
References	Windows Azure Blob MD5 Overview
Steps	Windows Azure Blob service provides mechanisms to ensure data integrity both at the application and transport layers. If for any reason you need to use HTTP instead of HTTPS and you are working with block blobs, you can use MD5 checking to help verify the integrity of the blobs being transferred This will help with protection from network/transport layer errors, but not necessarily with intermediary attacks. If you can use HTTPS, which provides transport level security, then using MD5 checking is redundant and unnecessary.

Use SMB 3.0 compatible client to ensure in-transit data encryption to Azure File shares

TITLE	DETAILS
Component	Mobile Client
SDL Phase	Build
Applicable Technologies	Generic
Attributes	StorageType - File
References	Azure File Storage, Azure File Storage SMB Support for Windows Clients

TITLE	DETAILS
Steps	Azure File Storage supports HTTPS when using the REST API, but is more commonly used as an SMB file share attached to a VM. SMB 2.1 does not support encryption, so connections are only allowed within the same region in Azure. However, SMB 3.0 supports encryption, and can be used with Windows Server 2012 R2, Windows 8, Windows 8.1, and Windows 10, allowing cross-region access and even access on the desktop.

Implement Certificate Pinning

TITLE	DETAILS
Component	Azure Storage
SDL Phase	Build
Applicable Technologies	Generic, Windows Phone
Attributes	N/A
References	Certificate and Public Key Pinning
Steps	Certificate pinning defends against Man-In-The-Middle (MITM) attacks. Pinning is the process of associating a host with their expected X509 certificate or public key. Once a certificate or public key is known or seen for a host, the certificate or public key is associated or 'pinned' to the host. Thus, when an adversary attempts to do SSL MITM attack, during SSL handshake the key from attacker's server will be different from the pinned certificate's key, and the request will be discarded, thus preventing MITM. Certificate pinning can be achieved by implementing ServicePointManager's <code>ServerCertificateValidationCallback</code> delegate.

Example

```

using System;
using System.Net;
using System.Net.Security;
using System.Security.Cryptography;

namespace CertificatePinningExample
{
    class CertificatePinningExample
    {
        /* Note: In this example, we're hardcoding a the certificate's public key and algorithm for
        demonstration purposes. In a real-world application, this should be stored in a secure
        configuration area that can be updated as needed. */

        private static readonly string PINNED_ALGORITHM = "RSA";

        private static readonly string PINNED_PUBLIC_KEY = "3082010A0282010100B0E75B7C8E56D31658EF79B3A1" +
        "294D506A88DFCDD603F6EF15E7F5BC8DF32291EC5082B828A158E905FE6A83EE044A48258807FAC3D6356AF0982" +
        "3EDAB15D00507B70DB08DB9A20C7D1201417B3071A346D663A241061C151B6EC5B5B4ECCDCDBEA24F051962809" +
        "FEC499BF2D093C06E3BDA7D0B883CDC1C2C6660B8ECB2EA30A685ADE2DC83C88314010FFC7F4F0F895EDDBE5C02" +
        "ABF78E50B708E0A0EB984A9AA536BCE61A0C31D895425C6FEE5A564B158EE7C4F0693C439AE010EF83CA8155750" +
        "09817537C29F86071E5DD8CA50EBD8A409494F479807574D83EDCE6F68A8F7D40447471D05BC3F5EAD7862FA748" +
        "EA3C92A60A12834481CEF7A0B0D94E50203010001";

        public static void Main(string[] args)
        {
            HttpWebRequest request = (HttpWebRequest)WebRequest.Create("https://azure.microsoft.com");
            request.ServerCertificateValidationCallback = (sender, certificate, chain, sslPolicyErrors) =>
            {
                if (certificate == null || sslPolicyErrors != SslPolicyErrors.None)
                {
                    // Error getting certificate or the certificate failed basic validation
                    return false;
                }

                var targetKeyAlgorithm = new Oid(certificate.GetKeyAlgorithm()).FriendlyName;
                var targetPublicKey = certificate.GetPublicKeyString();

                if (targetKeyAlgorithm == PINNED_ALGORITHM &&
                    targetPublicKey == PINNED_PUBLIC_KEY)
                {
                    // Success, the certificate matches the pinned value.
                    return true;
                }
                // Reject, either the key or the algorithm does not match the expected value.
                return false;
            };

            try
            {
                var response = (HttpWebResponse)request.GetResponse();
                Console.WriteLine($"Success, HTTP status code: {response.StatusCode}");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Failure, {ex.Message}");
            }
            Console.WriteLine("Press any key to end.");
            Console.ReadKey();
        }
    }
}

```

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	NET Framework 3
Attributes	N/A
References	MSDN, Fortify Kingdom
Steps	The application configuration should ensure that HTTPS is used for all access to sensitive information. • EXPLANATION: If an application handles sensitive information and does not use message-level encryption, then it should only be allowed to communicate over an encrypted transport channel. • RECOMMENDATIONS: Ensure that HTTP transport is disabled and enable HTTPS transport instead. For example, replace the <code><httpTransport/></code> with <code><httpsTransport/></code> tag. Do not rely on a network configuration (firewall) to guarantee that the application can only be accessed over a secure channel. From a philosophical point of view, the application should not depend on the network for its security. From a practical point of view, the people responsible for securing the network do not always track the security requirements of the application as they evolve.

WCF: Set Message security Protection level to EncryptAndSign

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	.NET Framework 3
Attributes	N/A
References	MSDN

TITLE	DETAILS
Steps	• EXPLANATION: When Protection level is set to "none" it will disable message protection. Confidentiality and integrity is achieved with appropriate level of setting. • RECOMMENDATIONS: ◦ when <code>Mode=None</code> - Disables message protection ◦ when <code>Mode=Sign</code> - Signs but does not encrypt the message; should be used when data integrity is important ◦ when <code>Mode=EncryptAndSign</code> - Signs and encrypts the message Consider turning off encryption and only signing your message when you just need to validate the integrity of the information without concerns of confidentiality. This may be useful for operations or service contracts in which you need to validate the original sender but no sensitive data is transmitted. When reducing the protection level, be careful that the message does not contain any personally identifiable information (PII).

Example

Configuring the service and the operation to only sign the message is shown in the following examples. Service Contract Example of `ProtectionLevel.Sign`: The following is an example of using `ProtectionLevel.Sign` at the Service Contract level:

```
[ServiceContract(Protection Level=ProtectionLevel.Sign]
public interface IService
{
    string GetData(int value);
}
```

Example

Operation Contract Example of `ProtectionLevel.Sign` (for Granular Control): The following is an example of using `ProtectionLevel.Sign` at the OperationContract level:

```
[OperationContract(ProtectionLevel=ProtectionLevel.Sign]
string GetData(int value);
```

WCF: Use a least-privileged account to run your WCF service

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	.NET Framework 3
Attributes	N/A

TITLE	DETAILS
References	MSDN
Steps	• EXPLANATION: Do not run WCF services under admin or high privilege account. in case of services compromise it will result in high impact. • RECOMMENDATIONS: Use a least-privileged account to host your WCF service because it will reduce your application's attack surface and reduce the potential damage if you are attacked. If the service account requires additional access rights on infrastructure resources such as MSMQ, the event log, performance counters, and the file system, appropriate permissions should be given to these resources so that the WCF service can run successfully. If your service needs to access specific resources on behalf of the original caller, use impersonation and delegation to flow the caller's identity for a downstream authorization check. In a development scenario, use the local network service account, which is a special built-in account that has reduced privileges. In a production scenario, create a least-privileged custom domain service account.

Force all traffic to Web APIs over HTTPS connection

TITLE	DETAILS
Component	Web API
SDL Phase	Build
Applicable Technologies	MVC5, MVC6
Attributes	N/A
References	Enforcing SSL in a Web API Controller
Steps	If an application has both an HTTPS and an HTTP binding, clients can still use HTTP to access the site. To prevent this, use an action filter to ensure that requests to protected APIs are always over HTTPS.

Example

The following code shows a Web API authentication filter that checks for SSL:

```

public class RequireHttpsAttribute : AuthorizationFilterAttribute
{
    public override void OnAuthorization(HttpContext actionContext)
    {
        if (actionContext.Request.RequestUri.Scheme != Uri.UriSchemeHttps)
        {
            actionContext.Response = new HttpResponseMessage(System.Net.HttpStatusCode.Forbidden)
            {
                ReasonPhrase = "HTTPS Required"
            };
        }
        else
        {
            base.OnAuthorization(actionContext);
        }
    }
}

```

Add this filter to any Web API actions that require SSL:

```

public class ValuesController : ApiController
{
    [RequireHttps]
    public HttpResponseMessage Get() { ... }
}

```

Ensure that communication to Azure Redis Cache is over SSL

TITLE	DETAILS
Component	Azure Redis Cache
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Azure Redis SSL support
Steps	Redis server does not support SSL out of the box, but Azure Redis Cache does. If you are connecting to Azure Redis Cache and your client supports SSL, like StackExchange.Redis, then you should use SSL. By default non-SSL port is disabled for new Azure Redis Cache instances. Ensure that the secure defaults are not changed unless there is a dependency on SSL support for redis clients.

Please note that Redis is designed to be accessed by trusted clients inside trusted environments. This means that usually it is not a good idea to expose the Redis instance directly to the internet or, in general, to an environment where untrusted clients can directly access the Redis TCP port or UNIX socket.

Secure Device to Field Gateway communication

TITLE	DETAILS
Component	IoT Field Gateway
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	For IP based devices, the communication protocol could typically be encapsulated in a SSL/TLS channel to protect data in transit. For other protocols that do not support SSL/TLS investigate if there are secure versions of the protocol that provide security at transport or message layer.

Secure Device to Cloud Gateway communication using SSL/TLS

TITLE	DETAILS
Component	IoT Cloud Gateway
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Choose your Communication Protocol
Steps	Secure HTTP/AMQP or MQTT protocols using SSL/TLS.

Security Frame: Configuration Management | Mitigations

1/24/2016 • 20 min to read • [Edit Online](#)

PRODUCT/SERVICE	ARTICLE
Web Application	• Implement Content Security Policy (CSP), and disable inline javascript • Enable browser's XSS filter • ASP.NET applications must disable tracing and debugging prior to deployment • Access third-party javascripts from trusted sources only • Ensure that authenticated ASP.NET pages incorporate UI Redressing or click-jacking defenses • Ensure that only trusted origins are allowed if CORS is enabled on ASP.NET Web Applications • Enable ValidateRequest attribute on ASP.NET Pages • Use locally-hosted latest versions of JavaScript libraries • Disable automatic MIME sniffing • Remove standard server headers on Windows Azure Web Sites to avoid fingerprinting
Database	• Configure a Windows Firewall for Database Engine Access
Web API	• Ensure that only trusted origins are allowed if CORS is enabled on ASP.NET Web API • Encrypt sections of Web API's configuration files that contain sensitive data
IoT Device	• Ensure that all admin interfaces are secured with strong credentials • Ensure that unknown code cannot execute on devices • Encrypt OS and additional partitions of IoT Device with bit-locker • Ensure that only the minimum services/features are enabled on devices
IoT Field Gateway	• Encrypt OS and additional partitions of IoT Field Gateway with bit-locker • Ensure that the default login credentials of the field gateway are changed during installation
IoT Cloud Gateway	• Ensure that the Cloud Gateway implements a process to keep the connected devices firmware up to date

PRODUCT/SERVICE	ARTICLE
Machine Trust Boundary	• Ensure that devices have end-point security controls configured as per organizational policies
Azure Storage	• Ensure secure management of Azure storage access keys • Ensure that only trusted origins are allowed if CORS is enabled on Azure storage
WCF	• Enable WCF's service throttling feature • WCF-Information disclosure through metadata

Implement Content Security Policy (CSP), and disable inline javascript

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	An Introduction to Content Security Policy, Content Security Policy Reference, Security features, Introduction to content security policy, Can I use CSP?

TITLE	DETAILS
Steps	Content Security Policy (CSP) is a defense-in-depth security mechanism, a W3C standard, that enables web application owners to have control on the content embedded in their site. CSP is added as an HTTP response header on the web server and is enforced on the client side by browsers. It is a whitelist-based policy - a website can declare a set of trusted domains from which active content such as JavaScript can be loaded. CSP provides the following security benefits: • Protection against XSS: If a page is vulnerable to XSS, an attacker can exploit it in 2 ways: ◦ Inject <code><script>malicious_code</script></code> This exploit will not work due to CSP's Base Restriction-1 ◦ Inject <pre><script src="http://attacker.com/maliciousCode.js"/></pre> This exploit will not work since the attacker controlled domain will not be in CSP's whitelist of domains • Control over data exfiltration: If any malicious content on a webpage attempts to connect to an external website and steal data, the connection will be aborted by CSP. This is because the target domain will not be in CSP's whitelist • Defense against click-jacking: click-jacking is an attack technique using which an adversary can frame a genuine website and force users to click on UI elements. Currently defense against click-jacking is achieved by configuring a response header- X-Frame-Options. Not all browsers respect this header and going forward CSP will be a standard way to defend against click-jacking • Real-time attack reporting: If there is an injection attack on a CSP-enabled website, browsers will automatically trigger a notification to an endpoint configured on the webserver. This way, CSP serves as a real-time warning system.

Example

Example policy:

```
Content-Security-Policy: default-src 'self'; script-src 'self' www.google-analytics.com
```

This policy allows scripts to load only from the web application's server and google analytics server. Scripts loaded from any other site will be rejected. When CSP is enabled on a website, the following features are automatically disabled to mitigate XSS attacks.

Example

Inline scripts will not execute. Following are examples of inline scripts

```
<script> some Javascript code </script>
Event handling attributes of HTML tags (e.g., <button onclick="function(){}">
javascript:alert(1);
```

Example

Strings will not be evaluated as code.

```
Example: var str="alert(1)"; eval(str);
```

Enable browser's XSS filter

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	XSS Protection Filter
Steps	X-XSS-Protection response header configuration controls the browser's cross site script filter. This response header can have following values: • 0: This will disable the filter • 1: Filter enabled If a cross-site scripting attack is detected, in order to stop the attack, the browser will sanitize the page • 1: mode=block : Filter enabled Rather than sanitize the page, when a XSS attack is detected, the browser will prevent rendering of the page • 1: report=http://[YOURDOMAIN]/your_report_URI : Filter enabled The browser will sanitize the page and report the violation. This is a Chromium function utilizing CSP violation reports to send details to a URI of your choice. The last 2 options are considered safe values.

ASP.NET applications must disable tracing and debugging prior to deployment

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A

TITLE	DETAILS
References	ASP.NET Debugging Overview, ASP.NET Tracing Overview, How to: Enable Tracing for an ASP.NET Application, How to: Enable Debugging for ASP.NET Applications
Steps	When tracing is enabled for the page, every browser requesting it also obtains the trace information that contains data about internal server state and workflow. That information could be security sensitive. When debugging is enabled for the page, errors happening on the server result in a full stack trace data presented to the browser. That data may expose security-sensitive information about the server's workflow.

Access third-party javascripts from trusted sources only

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	third-party JavaScripts should be referenced only from trusted sources. The reference endpoints should always be on SSL.

Ensure that authenticated ASP.NET pages incorporate UI Redressing or click-jacking defenses

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	OWASP click-jacking Defense Cheat Sheet, IE Internals - Combating click-jacking With X-Frame-Options

TITLE	DETAILS
Steps	click-jacking, also known as a "UI redress attack", is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top-level page. This layering is achieved by crafting a malicious page with an iframe, which loads the victim's page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both. To prevent click-jacking attacks, set the proper X-Frame-Options HTTP response headers that instruct the browser to not allow framing from other domains

Example

The X-FRAME-OPTIONS header can be set via IIS web.config. Web.config code snippet for sites that should never be framed:

```
<system.webServer>
  <httpProtocol>
    <customHeader>
      <add name="X-FRAME-OPTIONS" value="DENY"/>
    </customHeader>
  </httpProtocol>
</system.webServer>
```

Example

Web.config code for sites that should only be framed by pages in the same domain:

```
<system.webServer>
  <httpProtocol>
    <customHeader>
      <add name="X-FRAME-OPTIONS" value="SAMEORIGIN"/>
    </customHeader>
  </httpProtocol>
</system.webServer>
```

Ensure that only trusted origins are allowed if CORS is enabled on ASP.NET Web Applications

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Web Forms, MVC5
Attributes	N/A
References	N/A

TITLE	DETAILS
Steps	Browser security prevents a web page from making AJAX requests to another domain. This restriction is called the same-origin policy, and prevents a malicious site from reading sensitive data from another site. However, sometimes it might be required to expose APIs securely which other sites can consume. Cross Origin Resource Sharing (CORS) is a W3C standard that allows a server to relax the same-origin policy. Using CORS, a server can explicitly allow some cross-origin requests while rejecting others. CORS is safer and more flexible than earlier techniques such as JSONP. At its core, enabling CORS translates to adding a few HTTP response headers (Access-Control-*) to the web application and this can be done in a couple of ways.

Example

If access to Web.config is available, then CORS can be added through the following code:

```
<system.webServer>
  <httpProtocol>
    <customHeaders>
      <clear />
      <add name="Access-Control-Allow-Origin" value="http://example.com" />
    </customHeaders>
  </httpProtocol>
```

Example

If access to web.config is not available, then CORS can be configured by adding the following CSharp code:

```
HttpContext.Response.AppendHeader("Access-Control-Allow-Origin", "http://example.com")
```

Please note that it is critical to ensure that the list of origins in "Access-Control-Allow-Origin" attribute is set to a finite and trusted set of origins. Failing to configure this inappropriately (e.g., setting the value as '*') will allow malicious sites to trigger cross origin requests to the web application >without any restrictions, thereby making the application vulnerable to CSRF attacks.

Enable ValidateRequest attribute on ASP.NET Pages

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Web Forms, MVC5
Attributes	N/A
References	Request Validation - Preventing Script Attacks

IriusRisk

TITLE	DETAILS
Steps	Request validation, a feature of ASP.NET since version 1.1, prevents the server from accepting content containing un-encoded HTML. This feature is designed to help prevent some script-injection attacks whereby client script code or HTML can be unknowingly submitted to a server, stored, and then presented to other users. We still strongly recommend that you validate all input data and HTML encode it when appropriate. Request validation is performed by comparing all input data to a list of potentially dangerous values. If a match occurs, ASP.NET raises an <code>HttpRequestValidationException</code>. By default, Request Validation feature is enabled.

Example

However, this feature can be disabled at page level:

```
<%@ Page validateRequest="false" %>
```

or, at application level

```
<configuration>
  <system.web>
    <pages validateRequest="false" />
  </system.web>
</configuration>
```

Please note that Request Validation feature is not supported, and is not part of MVC6 pipeline.

Use locally-hosted latest versions of JavaScript libraries

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A

TITLE	DETAILS
Steps	Developers using standard JavaScript libraries like JQuery must use approved versions of common JavaScript libraries that do not contain known security flaws. A good practice is to use the most latest version of the libraries, since they contain security fixes for known vulnerabilities in their older versions. If the most recent release cannot be used due to compatibility reasons, the below minimum versions should be used. Acceptable minimum versions: • JQuery ◦ JQuery 1.7.1 ◦ JQueryUI 1.10.0 ◦ JQuery Validate 1.9 ◦ JQuery Mobile 1.0.1 ◦ JQuery Cycle 2.99 ◦ JQuery DataTables 1.9.0 • Ajax Control Toolkit ◦ Ajax Control Toolkit 40412 • ASP.NET Web Forms and Ajax ◦ ASP.NET Web Forms and Ajax 4 ◦ ASP.NET Ajax 3.5 • ASP.NET MVC ◦ ASP.NET MVC 3.0 Never load any JavaScript library from external sites such as public CDNs

Disable automatic MIME sniffing

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	IE8 Security Part V: Comprehensive Protection, MIME type
Steps	The X-Content-Type-Options header is an HTTP header that allows developers to specify that their content should not be MIME-sniffed. This header is designed to mitigate MIME-Sniffing attacks. For each page that could contain user controllable content, you must use the HTTP Header X-Content-Type-Options:nosniff. To enable the required header globally for all pages in the application, you can do one of the following

Example

Add the header in the web.config file if the application is hosted by Internet Information Services (IIS) on Windows.

```

<system.webServer>
<httpProtocol>
<customHeaders>
<add name="X-Content-Type-Options" value="nosniff"/>
</customHeaders>
</httpProtocol>
</system.webServer>

```

Example

Add the header through the global Application_BeginRequest

```

void Application_BeginRequest(object sender, EventArgs e)
{
    this.Response.Headers["X-Content-Type-Options"] = "nosniff";
}

```

Example

Implement custom HTTP module

```

public class XContentTypeOptionsModule : IHttpModule
{
    #region IHttpModule Members
    public void Dispose()
    {
    }
    public void Init(HttpApplication context)
    {
        context.PreSendRequestHeaders += new EventHandler(context_PreSendRequestHeaders);
    }
    #endregion
    void context_PreSendRequestHeaders(object sender, EventArgs e)
    {
        HttpApplication application = sender as HttpApplication;
        if (application == null)
            return;
        if (application.Response.Headers["X-Content-Type-Options "] != null)
            return;
        application.Response.Headers.Add("X-Content-Type-Options ", "nosniff");
    }
}

```

Example

You can enable the required header only for specific pages by adding it to individual responses:

```

this.Response.Headers["X-Content-Type-Options"] = "nosniff";

```

Remove standard server headers on Windows Azure Web Sites to avoid fingerprinting

TITLE	DETAILS
Component	Web Application
SDL Phase	Build

TITLE	DETAILS
Applicable Technologies	Generic
Attributes	EnvironmentType - Azure
References	Removing standard server headers on Windows Azure Web Sites
Steps	Headers such as Server, X-Powered-By, X-AspNet-Version reveal information about the server and the underlying technologies. It is recommended to suppress these headers thereby preventing fingerprinting the application

Configure a Windows Firewall for Database Engine Access

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	SQL Azure, OnPrem
Attributes	N/A, SQL Version - V12
References	How to configure an Azure SQL database firewall, Configure a Windows Firewall for Database Engine Access
Steps	Firewall systems help prevent unauthorized access to computer resources. To access an instance of the SQL Server Database Engine through a firewall, you must configure the firewall on the computer running SQL Server to allow access

Ensure that only trusted origins are allowed if CORS is enabled on ASP.NET Web API

TITLE	DETAILS
Component	Web API
SDL Phase	Build
Applicable Technologies	MVC 5
Attributes	N/A
References	Enabling Cross-Origin Requests in ASP.NET Web API 2, ASP.NET Web API - CORS Support in ASP.NET Web API 2

TITLE	DETAILS
Steps	Browser security prevents a web page from making AJAX requests to another domain. This restriction is called the same-origin policy, and prevents a malicious site from reading sensitive data from another site. However, sometimes it might be required to expose APIs securely which other sites can consume. Cross Origin Resource Sharing (CORS) is a W3C standard that allows a server to relax the same-origin policy. Using CORS, a server can explicitly allow some cross-origin requests while rejecting others. CORS is safer and more flexible than earlier techniques such as JSONP.

Example

In the App_Start/WebApiConfig.cs, add the following code to the WebApiConfig.Register method

```
using System.Web.Http;
namespace WebService
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            // New code
            config.EnableCors();

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

Example

EnableCors attribute can be applied to action methods in a controller as follows:

```

public class ResourcesController : ApiController
{
    [EnableCors("http://localhost:55912", // Origin
               null,                      // Request headers
               "GET",                     // HTTP methods
               "bar",                     // Response headers
               SupportsCredentials=true // Allow credentials
    )]
    public HttpResponseMessage Get(int id)
    {
        var resp = Request.CreateResponse(HttpStatusCode.NoContent);
        resp.Headers.Add("bar", "a bar value");
        return resp;
    }
    [EnableCors("http://localhost:55912", // Origin
               "Accept, Origin, Content-Type", // Request headers
               "PUT",                         // HTTP methods
               PreflightMaxAge=600           // Preflight cache duration
    )]
    public HttpResponseMessage Put(Resource data)
    {
        return Request.CreateResponse(HttpStatusCode.OK, data);
    }
    [EnableCors("http://localhost:55912", // Origin
               "Accept, Origin, Content-Type", // Request headers
               "POST",                       // HTTP methods
               PreflightMaxAge=600           // Preflight cache duration
    )]
    public HttpResponseMessage Post(Resource data)
    {
        return Request.CreateResponse(HttpStatusCode.OK, data);
    }
}

```

Please note that it is critical to ensure that the list of origins in EnableCors attribute is set to a finite and trusted set of origins. Failing to configure this inappropriately (e.g., setting the value as '*') will allow malicious sites to trigger cross origin requests to the API without any restrictions, >thereby making the API vulnerable to CSRF attacks. EnableCors can be decorated at controller level.

Example

To disable CORS on a particular method in a class, the DisableCors attribute can be used as shown below:

```

[EnableCors("http://example.com", "Accept, Origin, Content-Type", "POST")]
public class ResourcesController : ApiController
{
    public HttpResponseMessage Put(Resource data)
    {
        return Request.CreateResponse(HttpStatusCode.OK, data);
    }
    public HttpResponseMessage Post(Resource data)
    {
        return Request.CreateResponse(HttpStatusCode.OK, data);
    }
    // CORS not allowed because of the [DisableCors] attribute
    [DisableCors]
    public HttpResponseMessage Delete(int id)
    {
        return Request.CreateResponse(HttpStatusCode.NoContent);
    }
}

```

TITLE	DETAILS
Component	Web API
SDL Phase	Build
Applicable Technologies	MVC 6
Attributes	N/A
References	Enabling Cross-Origin Requests (CORS) in ASP.NET Core 1.0
Steps	In ASP.NET Core 1.0, CORS can be enabled either using middleware or using MVC. When using MVC to enable CORS the same CORS services are used, but the CORS middleware is not.

Approach-1 Enabling CORS with middleware: To enable CORS for the entire application add the CORS middleware to the request pipeline using the UseCors extension method. A cross-origin policy can be specified when adding the CORS middleware using the CorsPolicyBuilder class. There are two ways to do this:

Example

The first is to call UseCors with a lambda. The lambda takes a CorsPolicyBuilder object:

```
public void Configure(IApplicationBuilder app)
{
    app.UseCors(builder =>
        builder.WithOrigins("http://example.com")
            .WithMethods("GET", "POST", "HEAD")
            .WithHeaders("accept", "content-type", "origin", "x-custom-header"));
}
```

Example

The second is to define one or more named CORS policies, and then select the policy by name at run time.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors(options =>
    {
        options.AddPolicy("AllowSpecificOrigin",
            builder => builder.WithOrigins("http://example.com"));
    });
}
public void Configure(IApplicationBuilder app)
{
    app.UseCors("AllowSpecificOrigin");
    app.Run(async (context) =>
    {
        await context.Response.WriteAsync("Hello World!");
    });
}
```

Approach-2 Enabling CORS in MVC: Developers can alternatively use MVC to apply specific CORS per action, per controller, or globally for all controllers.

Example

Per action: To specify a CORS policy for a specific action add the [EnableCors] attribute to the action. Specify the policy name.

```
public class HomeController : Controller
{
    [EnableCors("AllowSpecificOrigin")]
    public IActionResult Index()
    {
        return View();
    }
}
```

Example

Per controller:

```
[EnableCors("AllowSpecificOrigin")]
public class HomeController : Controller
{
}
```

Example

Globally:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
    services.Configure<MvcOptions>(options =>
    {
        options.Filters.Add(new CorsAuthorizationFilterFactory("AllowSpecificOrigin"));
    });
}
```

Please note that it is critical to ensure that the list of origins in EnableCors attribute is set to a finite and trusted set of origins. Failing to configure this inappropriately (e.g., setting the value as '*') will allow malicious sites to trigger cross origin requests to the API without any restrictions, >thereby making the API vulnerable to CSRF attacks.

Example

To disable CORS for a controller or action, use the [DisableCors] attribute.

```
[DisableCors]
public IActionResult About()
{
    return View();
}
```

Encrypt sections of Web API's configuration files that contain sensitive data

TITLE	DETAILS
Component	Web API
SDL Phase	Deployment
Applicable Technologies	Generic

TITLE	DETAILS
Attributes	N/A
References	How To: Encrypt Configuration Sections in ASP.NET 2.0 Using DPAPI, Specifying a Protected Configuration Provider, Using Azure Key Vault to protect application secrets
Steps	Configuration files such as the Web.config, appsettings.json are often used to hold sensitive information, including user names, passwords, database connection strings, and encryption keys. If you do not protect this information, your application is vulnerable to attackers or malicious users obtaining sensitive information such as account user names and passwords, database names and server names. Based on the deployment type (azure/on-prem), encrypt the sensitive sections of config files using DPAPI or services like Azure Key Vault.

Ensure that all admin interfaces are secured with strong credentials

TITLE	DETAILS
Component	IoT Device
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Any administrative interfaces that the device or field gateway exposes should be secured using strong credentials. Also, any other exposed interfaces like WiFi, SSH, File shares, FTP should be secured with strong credentials. Default weak passwords should not be used.

Ensure that unknown code cannot execute on devices

TITLE	DETAILS
Component	IoT Device
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Enabling Secure Boot and bit-locker Device Encryption on Windows 10 IoT Core

TITLE	DETAILS
Steps	UEFI Secure Boot restricts the system to only allow execution of binaries signed by a specified authority. This feature prevents unknown code from being executed on the platform and potentially weakening the security posture of it. Enable UEFI Secure Boot and restrict the list of certificate authorities that are trusted for signing code. Sign all code that is deployed on the device using one of the trusted authorities.

Encrypt OS and additional partitions of IoT Device with bit-locker

TITLE	DETAILS
Component	IoT Device
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Windows 10 IoT Core implements a lightweight version of bit-locker Device Encryption, which has a strong dependency on the presence of a TPM on the platform, including the necessary preOS protocol in UEFI that conducts the necessary measurements. These preOS measurements ensure that the OS later has a definitive record of how the OS was launched. Encrypt OS partitions using bit-locker and any additional partitions also in case they store any sensitive data.

Ensure that only the minimum services/features are enabled on devices

TITLE	DETAILS
Component	IoT Device
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Do not enable or turn off any features or services in the OS that is not required for the functioning of the solution. For e.g. if the device does not require a UI to be deployed, install Windows IoT Core in headless mode.

Encrypt OS and additional partitions of IoT Field Gateway with bit-locker

locker

TITLE	DETAILS
Component	IoT Field Gateway
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Windows 10 IoT Core implements a lightweight version of bit-locker Device Encryption, which has a strong dependency on the presence of a TPM on the platform, including the necessary preOS protocol in UEFI that conducts the necessary measurements. These preOS measurements ensure that the OS later has a definitive record of how the OS was launched. Encrypt OS partitions using bit-locker and any additional partitions also in case they store any sensitive data.

Ensure that the default login credentials of the field gateway are changed during installation

TITLE	DETAILS
Component	IoT Field Gateway
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure that the default login credentials of the field gateway are changed during installation

Ensure that the Cloud Gateway implements a process to keep the connected devices firmware up to date

TITLE	DETAILS
Component	IoT Cloud Gateway
SDL Phase	Build
Applicable Technologies	Generic

TITLE	DETAILS
Attributes	Gateway choice - Azure IoT Hub
References	IoT Hub Device Management Overview, How to update Device Firmware
Steps	LWM2M is a protocol from the Open Mobile Alliance for IoT Device Management. Azure IoT device management allows to interact with physical devices using device jobs. Ensure that the Cloud Gateway implements a process to routinely keep the device and other configuration data up to date using Azure IoT Hub Device Management.

Ensure that devices have end-point security controls configured as per organizational policies

TITLE	DETAILS
Component	Machine Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure that devices have end-point security controls such as bit-locker for disk-level encryption, anti-virus with updated signatures, host based firewall, OS upgrades, group policies etc. are configured as per organizational security policies.

Ensure secure management of Azure storage access keys

TITLE	DETAILS
Component	Azure Storage
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	Azure Storage security guide - Managing Your Storage Account Keys

TITLE	DETAILS
Steps	Key Storage: It is recommended to store the Azure Storage access keys in Azure Key Vault as a secret and have the applications retrieve the key from key vault. This is recommended due to the following reasons: • The application will never have the storage key hardcoded in a configuration file, which removes that avenue of somebody getting access to the keys without specific permission • Access to the keys can be controlled using Azure Active Directory. This means an account owner can grant access to the handful of applications that need to retrieve the keys from Azure Key Vault. Other applications will not be able to access the keys without granting them permission specifically • Key Regeneration: It is recommended to have a process in place to regenerate Azure storage access keys for security reasons. Details on why and how to plan for key regeneration are documented in the Azure Storage Security Guide reference article

Ensure that only trusted origins are allowed if CORS is enabled on Azure storage

TITLE	DETAILS
Component	Azure Storage
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	CORS Support for the Azure Storage Services
Steps	Azure Storage allows you to enable CORS – Cross Origin Resource Sharing. For each storage account, you can specify domains that can access the resources in that storage account. By default, CORS is disabled on all services. You can enable CORS by using the REST API or the storage client library to call one of the methods to set the service policies.

Enable WCF's service throttling feature

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	.NET Framework 3

TITLE	DETAILS
Attributes	N/A
References	MSDN, Fortify Kingdom
Steps	Not placing a limit on the use of system resources could result in resource exhaustion and ultimately a denial of service. • EXPLANATION: Windows Communication Foundation (WCF) offers the ability to throttle service requests. Allowing too many client requests can flood a system and exhaust its resources. On the other hand, allowing only a small number of requests to a service can prevent legitimate users from using the service. Each service should be individually tuned to and configured to allow the appropriate amount of resources. • RECOMMENDATIONS Enable WCF's service throttling feature and set limits appropriate for your application.

Example

The following is an example configuration with throttling enabled:

```
<system.serviceModel>
  <behaviors>
    <serviceBehaviors>
      <behavior name="Throttled">
        <serviceThrottling maxConcurrentCalls="[YOUR SERVICE VALUE]" maxConcurrentSessions="[YOUR SERVICE VALUE]"
maxConcurrentInstances="[YOUR SERVICE VALUE]" />
      </behavior>
    </serviceBehaviors>
  </behaviors>
  ...
</system.serviceModel>
```

WCF-Information disclosure through metadata

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	.NET Framework 3
Attributes	N/A
References	MSDN, Fortify Kingdom
Steps	Metadata can help attackers learn about the system and plan a form of attack. WCF services can be configured to expose metadata. Metadata gives detailed service description information and should not be broadcast in production environments. The <code>HttpGetEnabled</code> / <code>HttpsGetEnabled</code> properties of the <code>ServiceMetadata</code> class defines whether a service will expose the metadata

Example

The code below instructs WCF to broadcast a service's metadata

```
ServiceMetadataBehavior smb = new ServiceMetadataBehavior();  
smb.HttpGetEnabled = true;  
smb.HttpGetUrl = new Uri(EndPointAddress);  
Host.Description.Behaviors.Add(smb);
```

Do not broadcast service metadata in a production environment. Set the `HttpGetEnabled` / `HttpsGetEnabled` properties of the `ServiceMetadata` class to false.

Example

The code below instructs WCF to not broadcast a service's metadata.

```
ServiceMetadataBehavior smb = new ServiceMetadataBehavior();  
smb.HttpGetEnabled = false;  
smb.HttpGetUrl = new Uri(EndPointAddress);  
Host.Description.Behaviors.Add(smb);
```

Security Frame: Cryptography | Mitigations

8/22/2017 • 14 min to read • [Edit Online](#)

PRODUCT/SERVICE	ARTICLE
Web Application	• Use only approved symmetric block ciphers and key lengths • Use approved block cipher modes and initialization vectors for symmetric ciphers • Use approved asymmetric algorithms, key lengths, and padding • Use approved random number generators • Do not use symmetric stream ciphers • Use approved MAC/HMAC/keyed hash algorithms • Use only approved cryptographic hash functions
Database	• Use strong encryption algorithms to encrypt data in the database • SSIS packages should be encrypted and digitally signed • Add digital signature to critical database securables • Use SQL server EKM to protect encryption keys • Use AlwaysEncrypted feature if encryption keys should not be revealed to Database engine
IoT Device	• Store Cryptographic Keys securely on IoT Device
IoT Cloud Gateway	• Generate a random symmetric key of sufficient length for authentication to IoT Hub
Dynamics CRM Mobile Client	• Ensure a device management policy is in place that requires a use PIN and allows remote wiping
Dynamics CRM Outlook Client	• Ensure a device management policy is in place that requires a PIN/password/auto lock and encrypts all data (e.g. BitLocker)
Identity Server	• Ensure that signing keys are rolled over when using Identity Server • Ensure that cryptographically strong client ID, client secret are used in Identity Server

Use only approved symmetric block ciphers and key lengths

TITLE	DETAILS
Component	Web Application
IriusRisk	
Ex. 1002 - Page 142	

TITLE	DETAILS
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Products must use only those symmetric block ciphers and associated key lengths which have been explicitly approved by the Crypto Advisor in your organization. Approved symmetric algorithms at Microsoft include the following block ciphers: • For new code AES-128, AES-192, and AES-256 are acceptable • For backward compatibility with existing code, three-key 3DES is acceptable • For products using symmetric block ciphers: ◦ Advanced Encryption Standard (AES) is required for new code ◦ Three-key triple Data Encryption Standard (3DES) is permissible in existing code for backward compatibility ◦ All other block ciphers, including RC2, DES, 2 Key 3DES, DESX, and Skipjack, may only be used for decrypting old data, and must be replaced if used for encryption • For symmetric block encryption algorithms, a minimum key length of 128 bits is required. The only block encryption algorithm recommended for new code is AES (AES-128, AES-192 and AES-256 are all acceptable) • Three-key 3DES is currently acceptable if already in use in existing code; transition to AES is recommended. DES, DESX, RC2, and Skipjack are no longer considered secure. These algorithms may only be used for decrypting existing data for the sake of backward-compatibility, and data should be re-encrypted using a recommended block cipher Please note that all symmetric block ciphers must be used with an approved cipher mode, which requires use of an appropriate initialization vector (IV). An appropriate IV, is typically a random number and never a constant value The use of legacy or otherwise unapproved crypto algorithms and smaller key lengths for reading existing data (as opposed to writing new data) may be permitted after your organization's Crypto Board review. However, you must file for an exception against this requirement. Additionally, in enterprise deployments, products should consider warning administrators when weak crypto is used to read data. Such warnings should be explanatory and actionable. In some cases, it may be appropriate to have Group Policy control the use of weak crypto Allowed .NET algorithms for managed crypto agility (in order of preference) • AesCng (FIPS compliant)

TITLE	DETAILS
	- AuthenticatedAesCng (FIPS compliant) - AESCryptoServiceProvider (FIPS compliant) - AESManaged (non-FIPS-compliant) Please note that none of these algorithms can be specified via the <code>SymmetricAlgorithm.Create</code> or <code>CryptoConfig.CreateFromName</code> methods without making changes to the machine.config file. Also, note that AES in versions of .NET prior to .NET 3.5 is named <code>RijndaelManaged</code>, and <code>AesCng</code> and <code>AuthenticatedAesCng</code> are > available through CodePlex and require CNG in the underlying OS

Use approved block cipher modes and initialization vectors for symmetric ciphers

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	All symmetric block ciphers must be used with an approved symmetric cipher mode. The only approved modes are CBC and CTS. In particular, the electronic code book (ECB) mode of operation should be avoided; use of ECB requires your organization's Crypto Board review. All usage of OFB, CFB, CTR, CCM, and GCM or any other encryption mode must be reviewed by your organization's Crypto Board. Reusing the same initialization vector (IV) with block ciphers in "streaming ciphers modes," such as CTR, may cause encrypted data to be revealed. All symmetric block ciphers must also be used with an appropriate initialization vector (IV). An appropriate IV is a cryptographically strong, random number and never a constant value.

Use approved asymmetric algorithms, key lengths, and padding

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A

TITLE	DETAILS
References	N/A
Steps	The use of banned cryptographic algorithms introduces significant risk to product security and must be avoided. Products must use only those cryptographic algorithms and associated key lengths and padding that have been explicitly approved by your organization's Crypto Board. • RSA- may be used for encryption, key exchange and signature. RSA encryption must use only the OAEP or RSA-KEM padding modes. Existing code may use PKCS #1 v1.5 padding mode for compatibility only. Use of null padding is explicitly banned. Keys ≥ 2048 bits is required for new code. Existing code may support keys < 2048 bits only for backwards compatibility after a review by your organization's Crypto Board. Keys < 1024 bits may only be used for decrypting/verifying old data, and must be replaced if used for encryption or signing operations • ECDSA- may be used for signature only. ECDSA with ≥ 256-bit keys is required for new code. ECDSA-based signatures must use one of the three NIST approved curves (P-256, P-384, or P521). Curves that have been thoroughly analyzed may be used only after a review with your organization's Crypto Board. • ECDH- may be used for key exchange only. ECDH with ≥ 256-bit keys is required for new code. ECDH-based key exchange must use one of the three NIST approved curves (P-256, P-384, or P521). Curves that have been thoroughly analyzed may be used only after a review with your organization's Crypto Board. • DSA- may be acceptable after review and approval from your organization's Crypto Board. Contact your security advisor to schedule your organization's Crypto Board review. If your use of DSA is approved, note that you will need to prohibit use of keys less than 2048 bits in length. CNG supports 2048-bit and greater key lengths as of Windows 8. • Diffie-Hellman- may be used for session key management only. Key length ≥ 2048 bits is required for new code. Existing code may support key lengths < 2048 bits only for backwards compatibility after a review by your organization's Crypto Board. Keys < 1024 bits may not be used.

Use approved random number generators

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A

TITLE	DETAILS
References	N/A
Steps	Products must use approved random number generators. Pseudorandom functions such as the C runtime function rand, the .NET Framework class System.Random, or system functions such as GetTickCount must, therefore, never be used in such code. Use of the dual elliptic curve random number generator (DUAL_EC_DRBG) algorithm is prohibited • CNG- BCryptGenRandom(use of the BCrypt_USE_SYSTEM_PREFERRED_RNG flag recommended unless the caller might run at any IRQL greater than 0 [that is, PASSIVE_LEVEL]) • CAPI- cryptGenRandom • Win32/64- RtlGenRandom (new implementations should use BCryptGenRandom or CryptGenRandom) * rand_s * SystemPrng (for kernel mode) • .NET- RNGCryptoServiceProvider or RNGCng • Windows Store Apps- Windows.Security.Cryptography.CryptographicBuffer.GenerateRandom or .GenerateRandomNumber • Apple OS X (10.7+)/iOS(2.0+)- int SecRandomCopyBytes (SecRandomRef random, size_t count, uint8_t bytes) • <i>Apple OS X (< 10.7)</i>-* Use /dev/random to retrieve random numbers • Java(including Google Android Java code)- java.security.SecureRandom class. Note that for Android 4.3 (Jelly Bean), developers must follow the Android recommended workaround and update their applications to explicitly initialize the PRNG with entropy from /dev/urandom or /dev/random

Do not use symmetric stream ciphers

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Symmetric stream ciphers, such as RC4, must not be used. Instead of symmetric stream ciphers, products should use a block cipher, specifically AES with a key length of at least 128 bits.

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Products must use only approved message authentication code (MAC) or hash-based message authentication code (HMAC) algorithms. A message authentication code (MAC) is a piece of information attached to a message that allows its recipient to verify both the authenticity of the sender and the integrity of the message using a secret key. The use of either a hash-based MAC (HMAC) or block-cipher-based MAC is permissible as long as all underlying hash or symmetric encryption algorithms are also approved for use; currently this includes the HMAC-SHA2 functions (HMAC-SHA256, HMAC-SHA384 and HMAC-SHA512) and the CMAC/OMAC1 and OMAC2 block cipher-based MACs (these are based on AES). Use of HMAC-SHA1 may be permissible for platform compatibility, but you will be required to file an exception to this procedure and undergo your organization's Crypto review. Truncation of HMACs to less than 128 bits is not permitted. Using customer methods to hash a key and data is not approved, and must undergo your organization's Crypto Board review prior to use.

Use only approved cryptographic hash functions

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A

TITLE	DETAILS
Steps	Products must use the SHA-2 family of hash algorithms (SHA256, SHA384, and SHA512). If a shorter hash is needed, such as a 128-bit output length in order to fit a data structure designed with the shorter MD5 hash in mind, product teams may truncate one of the SHA2 hashes (typically SHA256). Note that SHA384 is a truncated version of SHA512. Truncation of cryptographic hashes for security purposes to less than 128 bits is not permitted. New code must not use the MD2, MD4, MD5, SHA-0, SHA-1, or RIPEMD hash algorithms. Hash collisions are computationally feasible for these algorithms, which effectively breaks them. Allowed .NET hash algorithms for managed crypto agility (in order of preference): • SHA512Cng (FIPS compliant) • SHA384Cng (FIPS compliant) • SHA256Cng (FIPS compliant) • SHA512Managed (non-FIPS-compliant) (use SHA512 as algorithm name in calls to HashAlgorithm.Create or CryptoConfig.CreateFromName) • SHA384Managed (non-FIPS-compliant) (use SHA384 as algorithm name in calls to HashAlgorithm.Create or CryptoConfig.CreateFromName) • SHA256Managed (non-FIPS-compliant) (use SHA256 as algorithm name in calls to HashAlgorithm.Create or CryptoConfig.CreateFromName) • SHA512CryptoServiceProvider (FIPS compliant) • SHA256CryptoServiceProvider (FIPS compliant) • SHA384CryptoServiceProvider (FIPS compliant)

Use strong encryption algorithms to encrypt data in the database

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Choosing an encryption algorithm
Steps	Encryption algorithms define data transformations that cannot be easily reversed by unauthorized users. SQL Server allows administrators and developers to choose from among several algorithms, including DES, Triple DES, TRIPLE_DES_3KEY, RC2, RC4, 128-bit RC4, DESX, 128-bit AES, 192-bit AES, and 256-bit AES

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Identify the Source of Packages with Digital Signatures, Threat and Vulnerability Mitigation (Integration Services)
Steps	The source of a package is the individual or organization that created the package. Running a package from an unknown or untrusted source might be risky. To prevent unauthorized tampering of SSIS packages, digital signatures should be used. Also, to ensure the confidentiality of the packages during storage/transit, SSIS packages have to be encrypted

Add digital signature to critical database securables

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	ADD SIGNATURE (Transact-SQL)
Steps	In cases where the integrity of a critical database securable has to be verified, digital signatures should be used. Database securables such as a stored procedure, function, assembly, or trigger can be digitally signed. Below is an example of when this can be useful: Let us say an ISV (Independent Software Vendor) has provided support to a software delivered to one of their customers. Before providing support, the ISV would want to ensure that a database securable in the software was not tampered either by mistake or by a malicious attempt. If the securable is digitally signed, the ISV can verify its digital signature and validate its integrity.

Use SQL server EKM to protect encryption keys

TITLE	DETAILS
Component	Database
SDL Phase	Build

TITLE	DETAILS
Applicable Technologies	Generic
Attributes	N/A
References	SQL Server Extensible Key Management (EKM), Extensible Key Management Using Azure Key Vault (SQL Server)
Steps	SQL Server Extensible Key Management enables the encryption keys that protect the database files to be stored in an off-box device such as a smartcard, USB device, or EKM/HSM module. This also enables data protection from database administrators (except members of the sysadmin group). Data can be encrypted by using encryption keys that only the database user has access to on the external EKM/HSM module.

Use AlwaysEncrypted feature if encryption keys should not be revealed to Database engine

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	SQL Azure, OnPrem
Attributes	SQL Version - V12, MsSQL2016
References	Always Encrypted (Database Engine)
Steps	Always Encrypted is a feature designed to protect sensitive data, such as credit card numbers or national identification numbers (e.g. U.S. social security numbers), stored in Azure SQL Database or SQL Server databases. Always Encrypted allows clients to encrypt sensitive data inside client applications and never reveal the encryption keys to the Database Engine (SQL Database or SQL Server). As a result, Always Encrypted provides a separation between those who own the data (and can view it) and those who manage the data (but should have no access)

Store Cryptographic Keys securely on IoT Device

TITLE	DETAILS
Component	IoT Device
SDL Phase	Build
Applicable Technologies	Generic

TITLE	DETAILS
Attributes	Device OS - Windows IoT Core, Device Connectivity - Azure IoT device SDKs
References	TPM on Windows IoT Core, Set up TPM on Windows IoT Core, Azure IoT Device SDK TPM
Steps	Symmetric or Certificate Private keys securely in a hardware protected storage like TPM or Smart Card chips. Windows 10 IoT Core supports the user of a TPM and there are several compatible TPMs that can be used: https://developer.microsoft.com/windows/iot/win10/tpm. It is recommended to use a Firmware or Discrete TPM. A Software TPM should only be used for development and testing purposes. Once a TPM is available and the keys are provisioned in it, the code that generates the token should be written without hard coding any sensitive information in it.

Example

```
TpmDevice myDevice = new TpmDevice(0);
// Use logical device 0 on the TPM
string hubUri = myDevice.GetHostName();
string deviceId = myDevice.GetDeviceId();
string sasToken = myDevice.GetSASToken();

var deviceClient = DeviceClient.Create( hubUri, AuthenticationMethodFactory.
CreateAuthenticationWithToken(deviceId, sasToken), TransportType.Amqp);
```

As can be seen, the device primary key is not present in the code. Instead, it is stored in the TPM at slot 0. TPM device generates a short-lived SAS token that is then used to connect to the IoT Hub.

Generate a random symmetric key of sufficient length for authentication to IoT Hub

TITLE	DETAILS
Component	IoT Cloud Gateway
SDL Phase	Build
Applicable Technologies	Generic
Attributes	Gateway choice - Azure IoT Hub
References	N/A
Steps	IoT Hub contains a device Identity Registry and while provisioning a device, automatically generates a random Symmetric key. It is recommended to use this feature of the Azure IoT Hub Identity Registry to generate the key used for authentication. IoT Hub also allows for a key to be specified while creating the device. If a key is generated outside of IoT Hub during device provisioning, it is recommended to create a random symmetric key or at least 256 bits.

Ensure a device management policy is in place that requires a use PIN and allows remote wiping

TITLE	DETAILS
Component	Dynamics CRM Mobile Client
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure a device management policy is in place that requires a use PIN and allows remote wiping

Ensure a device management policy is in place that requires a PIN/password/auto lock and encrypts all data (e.g. Bitlocker)

TITLE	DETAILS
Component	Dynamics CRM Outlook Client
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure a device management policy is in place that requires a PIN/password/auto lock and encrypts all data (e.g. Bitlocker)

Ensure that signing keys are rolled over when using Identity Server

TITLE	DETAILS
Component	Identity Server
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	Identity Server - Keys, Signatures and Cryptography

TITLE	DETAILS
Steps	Ensure that signing keys are rolled over when using Identity Server. The link in the references section explains how this should be planned without causing outages to applications relying on Identity Server.

Ensure that cryptographically strong client ID, client secret are used in Identity Server

TITLE	DETAILS
Component	Identity Server
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure that cryptographically strong client ID, client secret are used in Identity Server. The following guidelines should be used while generating a client ID and secret: • Generate a random GUID as the client ID • Generate a cryptographically random 256-bit key as the secret

Security Frame: Exception Management | Mitigations

1/24/2018 • 7 min to read • [Edit Online](#)

PRODUCT/SERVICE	ARTICLE
WCF	• WCF- Do not include <code>serviceDebug</code> node in configuration file • WCF- Do not include <code>serviceMetadata</code> node in configuration file
Web API	• Ensure that proper exception handling is done in ASP.NET Web API
Web Application	• Do not expose security details in error messages • Implement Default error handling page • Set Deployment Method to Retail in IIS • Exceptions should fail safely

WCF- Do not include serviceDebug node in configuration file

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	Generic, .NET Framework 3
Attributes	N/A
References	MSDN, Fortify Kingdom
Steps	Windows Communication Framework (WCF) services can be configured to expose debugging information. Debug information should not be used in production environments. The <code><serviceDebug></code> tag defines whether the debug information feature is enabled for a WCF service. If the attribute <code>includeExceptionDetailInFaults</code> is set to true, exception information from the application will be returned to clients. Attackers can leverage the additional information they gain from debugging output to mount attacks targeted on the framework, database, or other resources used by the application.

Example

The following configuration file includes the `<serviceDebug>` tag:

```

<configuration>
<system.serviceModel>
<behaviors>
<serviceBehaviors>
<behavior name=""MyServiceBehavior"">
<serviceDebug includeExceptionDetailInFaults=""True"" httpHelpPageEnabled=""True""/>
...

```

Disable debugging information in the service. This can be accomplished by removing the `<serviceDebug>` tag from your application's configuration file.

WCF- Do not include serviceMetadata node in configuration file

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	Generic
Attributes	Generic, NET Framework 3
References	MSDN, Fortify Kingdom
Steps	Publicly exposing information about a service can provide attackers with valuable insight into how they might exploit the service. The <code><serviceMetadata></code> tag enables the metadata publishing feature. Service metadata could contain sensitive information that should not be publicly accessible. At a minimum, only allow trusted users to access the metadata and ensure that unnecessary information is not exposed. Better yet, entirely disable the ability to publish metadata. A safe WCF configuration will not contain the <code><serviceMetadata></code> tag.

Ensure that proper exception handling is done in ASP.NET Web API

TITLE	DETAILS
Component	Web API
SDL Phase	Build
Applicable Technologies	MVC 5, MVC 6
Attributes	N/A
References	Exception Handling in ASP.NET Web API, Model Validation in ASP.NET Web API
Steps	By default, most uncaught exceptions in ASP.NET Web API are translated into an HTTP response with status code <code>500, Internal Server Error</code>

Example

To control the status code returned by the API, `HttpResponseException` can be used as shown below:

```
public Product GetProduct(int id)
{
    Product item = repository.Get(id);
    if (item == null)
    {
        throw new HttpResponseException(HttpStatusCode.NotFound);
    }
    return item;
}
```

Example

For further control on the exception response, the `HttpResponseMessage` class can be used as shown below:

```
public Product GetProduct(int id)
{
    Product item = repository.Get(id);
    if (item == null)
    {
        var resp = new HttpResponseMessage(HttpStatusCode.NotFound)
        {
            Content = new StringContent(string.Format("No product with ID = {0}", id)),
            ReasonPhrase = "Product ID Not Found"
        };
        throw new HttpResponseException(resp);
    }
    return item;
}
```

To catch unhandled exceptions that are not of the type `HttpResponseException`, Exception Filters can be used. Exception filters implement the `System.Web.Http.Filters.IExceptionHandler` interface. The simplest way to write an exception filter is to derive from the `System.Web.Http.Filters.ExceptionFilterAttribute` class and override the `OnException` method.

Example

Here is a filter that converts `NotImplementedException` exceptions into HTTP status code `501, Not Implemented`:

```
namespace ProductStore.Filters
{
    using System;
    using System.Net;
    using System.Net.Http;
    using System.Web.Http.Filters;

    public class NotImplExceptionHandlerAttribute : ExceptionFilterAttribute
    {
        public override void OnException(HttpActionExecutedContext context)
        {
            if (context.Exception is NotImplementedException)
            {
                context.Response = new HttpResponseMessage(HttpStatusCode.NotImplemented);
            }
        }
    }
}
```

There are several ways to register a Web API exception filter:

- By action
- By controller
- Globally

Example

To apply the filter to a specific action, add the filter as an attribute to the action:

```
public class ProductsController : ApiController
{
    [NotImplExceptionFilter]
    public Contact GetContact(int id)
    {
        throw new NotImplementedException("This method is not implemented");
    }
}
```

Example

To apply the filter to all of the actions on a `controller`, add the filter as an attribute to the `controller` class:

```
[NotImplExceptionFilter]
public class ProductsController : ApiController
{
    // ...
}
```

Example

To apply the filter globally to all Web API controllers, add an instance of the filter to the `GlobalConfiguration.Configuration.Filters` collection. Exception filters in this collection apply to any Web API controller action.

```
GlobalConfiguration.Configuration.Filters.Add(
    new ProductStore.NotImplExceptionFilterAttribute());
```

Example

For model validation, the model state can be passed to `CreateErrorResponse` method as shown below:

```
public HttpResponseMessage PostProduct(Product item)
{
    if (!ModelState.IsValid)
    {
        return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ModelState);
    }
    // Implementation not shown...
}
```

Check the links in the references section for additional details about exceptional handling and model validation in ASP.Net Web API

Do not expose security details in error messages

TITLE	DETAILS
Component	Web Application
IriusRisk	
Ex. 1002 - Page 157	

TITLE	DETAILS
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Generic error messages are provided directly to the user without including sensitive application data. Examples of sensitive data include: • Server names • Connection strings • Usernames • Passwords • SQL procedures • Details of dynamic SQL failures • Stack trace and lines of code • Variables stored in memory • Drive and folder locations • Application install points • Host configuration settings • Other internal application details Trapping all errors within an application and providing generic error messages, as well as enabling custom errors within IIS will help prevent information disclosure. SQL Server database and .NET Exception handling, among other error handling architectures, are especially verbose and extremely useful to a malicious user profiling your application. Do not directly display the contents of a class derived from the .NET Exception class, and ensure that you have proper exception handling so that an unexpected exception isn't inadvertently raised directly to the user. • Provide generic error messages directly to the user that abstract away specific details found directly in the exception/error message • Do not display the contents of a .NET exception class directly to the user • Trap all error messages and if appropriate inform the user via a generic error message sent to the application client • Do not expose the contents of the Exception class directly to the user, especially the return value from <code>.ToString()</code>, or the values of the Message or StackTrace properties. Securely log this information and display a more innocuous message to the user

Implement Default error handling page

TITLE	DETAILS
Component	Web Application IriusRisk

TITLE	DETAILS
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Edit ASP.NET Error Pages Settings Dialog Box
Steps	When an ASP.NET application fails and causes an HTTP/1.x 500 Internal Server Error, or a feature configuration (such as Request Filtering) prevents a page from being displayed, an error message will be generated. Administrators can choose whether or not the application should display a friendly message to the client, detailed error message to the client, or detailed error message to localhost only. The tag in the web.config has three modes: • On: Specifies that custom errors are enabled. If no defaultRedirect attribute is specified, users see a generic error. The custom errors are shown to the remote clients and to the local host • Off: Specifies that custom errors are disabled. The detailed ASP.NET errors are shown to the remote clients and to the local host • RemoteOnly: Specifies that custom errors are shown only to the remote clients, and that ASP.NET errors are shown to the local host. This is the default value Open the <code>web.config</code> file for the application/site and ensure that the tag has either <code><customErrors mode="RemoteOnly" /></code> or <code><customErrors mode="On" /></code> defined.

Set Deployment Method to Retail in IIS

TITLE	DETAILS
Component	Web Application
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	deployment Element (ASP.NET Settings Schema)

TITLE	DETAILS
Steps	The <code><deployment retail></code> switch is intended for use by production IIS servers. This switch is used to help applications run with the best possible performance and least possible security information leakages by disabling the application's ability to generate trace output on a page, disabling the ability to display detailed error messages to end users, and disabling the debug switch. Often times, switches and options that are developer-focused, such as failed request tracing and debugging, are enabled during active development. It is recommended that the deployment method on any production server be set to retail. open the machine.config file and ensure that <code><deployment retail="true" /></code> remains set to true.

Exceptions should fail safely

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Fail securely
Steps	Application should fail safely. Any method that returns a Boolean value, based on which certain decision is made, should have exception block carefully created. There are lot of logical errors due to which security issues creep in, when the exception block is written carelessly.

Example

```

public static bool ValidateDomain(string pathToValidate, Uri currentUrl)
{
    try
    {
        if (!string.IsNullOrEmpty(pathToValidate))
        {
            var domain = RetrieveDomain(currentUrl);
            var replyPath = new Uri(pathToValidate);
            var replyDomain = RetrieveDomain(replyPath);

            if (string.Compare(domain, replyDomain, StringComparison.OrdinalIgnoreCase) != 0)
            {
                //// Adding additional check to enable CMS urls if they are not hosted on same domain.
                if (!string.IsNullOrEmpty(Utilities.CmsBase))
                {
                    var cmsDomain = RetrieveDomain(new Uri(Utilities.Base.Trim()));
                    if (string.Compare(cmsDomain, replyDomain, StringComparison.OrdinalIgnoreCase) != 0)
                    {
                        return false;
                    }
                    else
                    {
                        return true;
                    }
                }

                return false;
            }
        }

        return true;
    }
    catch (UriFormatException ex)
    {
        LogHelper.LogException("Utilities:ValidateDomain", ex);
        return true;
    }
}

```

The above method will always return True, if some exception happens. If the end user provides a malformed URL, that the browser respects, but the `Uri()` constructor doesn't, this will throw an exception, and the victim will be taken to the valid but malformed URL.

Security Frame: Input Validation | Mitigations

1/24/2018 • 31 min to read • [Edit Online](#)

PRODUCT/SERVICE	ARTICLE
Web Application	• Disable XSLT scripting for all transforms using untrusted style sheets • Ensure that each page that could contain user controllable content opts out of automatic MIME sniffing • Harden or Disable XML Entity Resolution • Applications utilizing http.sys perform URL canonicalization verification • Ensure appropriate controls are in place when accepting files from users • Ensure that type-safe parameters are used in Web Application for data access • Use separate model binding classes or binding filter lists to prevent MVC mass assignment vulnerability • Encode untrusted web output prior to rendering • Perform input validation and filtering on all string type Model properties • Sanitization should be applied on form fields that accept all characters, e.g. rich text editor • Do not assign DOM elements to sinks that do not have inbuilt encoding • Validate all redirects within the application are closed or done safely • Implement input validation on all string type parameters accepted by Controller methods • Set upper limit timeout for regular expression processing to prevent DoS due to bad regular expressions • Avoid using Html.Raw in Razor views
Database	• Do not use dynamic queries in stored procedures
Web API	• Ensure that model validation is done on Web API methods • Implement input validation on all string type parameters accepted by Web API methods • Ensure that type-safe parameters are used in Web API for data access
Azure Document DB	• Use parametrized SQL queries for Azure Cosmos DB
WCF	• WCF Input validation through Schema binding • WCF- Input validation through Parameter Inspectors

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	XSLT Security, XsltSettings.EnableScript Property
Steps	XSLT supports scripting inside style sheets using the <code><msxml:script></code> element. This allows custom functions to be used in an XSLT transformation. The script is executed under the context of the process performing the transform. XSLT script must be disabled when in an untrusted environment to prevent execution of untrusted code. <i>If using .NET:</i> XSLT scripting is disabled by default; however, you must ensure that it has not been explicitly enabled through the <code>XsltSettings.EnableScript</code> property.

Example

```
XsltSettings settings = new XsltSettings();
settings.EnableScript = true; // WRONG: THIS SHOULD BE SET TO false
```

Example

If you are using using MSXML 6.0, XSLT scripting is disabled by default; however, you must ensure that it has not been explicitly enabled through the XML DOM object property AllowXsltScript.

```
doc.setProperty("AllowXsltScript", true); // WRONG: THIS SHOULD BE SET TO false
```

Example

If you are using MSXML 5 or below, XSLT scripting is enabled by default and you must explicitly disable it. Set the XML DOM object property AllowXsltScript to false.

```
doc.setProperty("AllowXsltScript", false); // CORRECT. Setting to false disables XSLT scripting.
```

Ensure that each page that could contain user controllable content opts out of automatic MIME sniffing

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A

TITLE	DETAILS
References	IE8 Security Part V - Comprehensive Protection
Steps	For each page that could contain user controllable content, you must use the HTTP Header <code>X-Content-Type-Options:nosniff</code>. To comply with this requirement, you can either set the required header page by page for only those pages that might contain user-controllable content, or you can set it globally for all pages in the application. Each type of file delivered from a web server has an associated MIME type (also called a <i>content-type</i>) that describes the nature of the content (that is, image, text, application, etc.) The X-Content-Type-Options header is an HTTP header that allows developers to specify that their content should not be MIME-sniffed. This header is designed to mitigate MIME-Sniffing attacks. Support for this header was added in Internet Explorer 8 (IE8) Only users of Internet Explorer 8 (IE8) will benefit from X-Content-Type-Options. Previous versions of Internet Explorer do not currently respect the X-Content-Type-Options header Internet Explorer 8 (and later) are the only major browsers to implement a MIME-sniffing opt-out feature. If and when other major browsers (Firefox, Safari, Chrome) implement similar features, this recommendation will be updated to include syntax for those browsers as well

Example

To enable the required header globally for all pages in the application, you can do one of the following:

- Add the header in the web.config file if the application is hosted by Internet Information Services (IIS) 7

```
<system.webServer>
  <httpProtocol>
    <customHeaders>
      <add name="X-Content-Type-Options" value="nosniff"/>
    </customHeaders>
  </httpProtocol>
</system.webServer>
```

- Add the header through the global Application_BeginRequest

```
void Application_BeginRequest(object sender, EventArgs e)
{
    this.Response.Headers["X-Content-Type-Options"] = "nosniff";
}
```

- Implement custom HTTP module

```

public class XContentTypeOptionsModule : IHttpModule
{
    #region IHttpModule Members
    public void Dispose()
    {

    }
    public void Init(HttpApplication context)
    {
        context.PreSendRequestHeaders += new EventHandler(context_PreSendRequestHeaders);
    }
    #endregion
    void context_PreSendRequestHeaders(object sender, EventArgs e)
    {
        HttpApplication application = sender as HttpApplication;
        if (application == null)
            return;
        if (application.Response.Headers["X-Content-Type-Options "] != null)
            return;
        application.Response.Headers.Add("X-Content-Type-Options ", "nosniff");
    }
}

```

- You can enable the required header only for specific pages by adding it to individual responses:

```

this.Response.Headers["X-Content-Type-Options"] = "nosniff";

```

Harden or Disable XML Entity Resolution

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	XML Entity Expansion, XML Denial of Service Attacks and Defenses, MSXML Security Overview, Best Practices for Securing MSXML Code, NSXMLParserDelegate Protocol Reference, Resolving External References

TITLE	DETAILS
Steps	Although it is not widely used, there is a feature of XML that allows the XML parser to expand macro entities with values defined either within the document itself or from external sources. For example, the document might define an entity "companyname" with the value "Microsoft," so that every time the text "&companyname;" appears in the document, it is automatically replaced with the text Microsoft. Or, the document might define an entity "MSFTStock" that references an external web service to fetch the current value of Microsoft stock. Then any time "&MSFTStock;" appears in the document, it is automatically replaced with the current stock price. However, this functionality can be abused to create denial of service (DoS) conditions. An attacker can nest multiple entities to create an exponential expansion XML bomb that consumes all available memory on the system. Alternatively, he can create an external reference that streams back an infinite amount of data or that simply hangs the thread. As a result, all teams must disable internal and/or external XML entity resolution entirely if their application does not use it, or manually limit the amount of memory and time that the application can consume for entity resolution if this functionality is absolutely necessary. If entity resolution is not required by your application, then disable it.

Example

For .NET Framework code, you can use the following approaches:

```

XmlTextReader reader = new XmlTextReader(stream);
reader.ProhibitDtd = true;

XmlReaderSettings settings = new XmlReaderSettings();
settings.ProhibitDtd = true;
XmlReader reader = XmlReader.Create(stream, settings);

// for .NET 4
XmlReaderSettings settings = new XmlReaderSettings();
settings.DtdProcessing = DtdProcessing.Prohibit;
XmlReader reader = XmlReader.Create(stream, settings);

```

Note that the default value of `ProhibitDtd` in `XmlReaderSettings` is true, but in `XmlTextReader` it is false. If you are using `XmlReaderSettings`, you do not need to set `ProhibitDtd` to true explicitly, but it is recommended for safety sake that you do. Also note that the `XmlDocument` class allows entity resolution by default.

Example

To disable entity resolution for `XmlDocuments`, use the `XmlDocument.Load(XmlReader)` overload of the `Load` method and set the appropriate properties in the `XmlReader` argument to disable resolution, as illustrated in the following code:

```

XmlReaderSettings settings = new XmlReaderSettings();
settings.ProhibitDtd = true;
XmlReader reader = XmlReader.Create(stream, settings);
XmlDocument doc = new XmlDocument();
doc.Load(reader);

```

Example

If disabling entity resolution is not possible for your application, set the `XmlReaderSettings.MaxCharactersFromEntities` property to a reasonable value according to your application's needs. This will limit the impact of potential exponential expansion DoS attacks. The following code provides an example of this approach:

```
XmlReaderSettings settings = new XmlReaderSettings();
settings.ProhibitDtd = false;
settings.MaxCharactersFromEntities = 1000;
XmlReader reader = XmlReader.Create(stream, settings);
```

Example

If you need to resolve inline entities but do not need to resolve external entities, set the `XmlReaderSettings.XmlResolver` property to null. For example:

```
XmlReaderSettings settings = new XmlReaderSettings();
settings.ProhibitDtd = false;
settings.MaxCharactersFromEntities = 1000;
settings.XmlResolver = null;
XmlReader reader = XmlReader.Create(stream, settings);
```

Note that in MSXML6, `ProhibitDTD` is set to true (disabling DTD processing) by default. For Apple OSX/iOS code, there are two XML parsers you can use: `NSXMLParser` and `libXML2`.

Applications utilizing http.sys perform URL canonicalization verification

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A

TITLE	DETAILS
Steps	Any application that uses http.sys should follow these guidelines: • Limit the URL length to no more than 16,384 characters (ASCII or Unicode). This is the absolute maximum URL length based on the default Internet Information Services (IIS) 6 setting. Websites should strive for a length shorter than this if possible • Use the standard .NET Framework file I/O classes (such as FileStream) as these will take advantage of the canonicalization rules in the .NET FX • Explicitly build an allow-list of known filenames • Explicitly reject known filetypes you will not serve UrlScan rejects: exe, bat, cmd, com, htw, ida, idq, htr, idc, shtm[l], stm, printer, ini, pol, dat files • Catch the following exceptions: ◦ System.ArgumentException (for device names) ◦ System.NotSupportedException (for data streams) ◦ System.IO.FileNotFoundException (for invalid escaped filenames) ◦ System.IO.DirectoryNotFoundException (for invalid escaped dirs) • <i>Do not</i> call out to Win32 file I/O APIs. On an invalid URL gracefully return a 400 error to the user, and log the real error.

Ensure appropriate controls are in place when accepting files from users

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Unrestricted File Upload, File Signature Table


```

        new byte[] { 0xFF, 0x08, 0xFF, 0xE0 },
        new byte[] { 0xFF, 0x08, 0xFF, 0xE2 },
        new byte[] { 0xFF, 0x08, 0xFF, 0xE3 }
    },
    { ".XLS", new List<byte[]>
    {
        new byte[] { 0xD0, 0xCF, 0x11, 0xE0, 0xA1, 0xB1, 0x1A, 0xE1 },
        new byte[] { 0x09, 0x08, 0x10, 0x00, 0x00, 0x06, 0x05, 0x00 },
        new byte[] { 0xFD, 0xFF, 0xFF, 0xFF }
    }
    },
    { ".XLSX", new List<byte[]> { new byte[] { 0x50, 0x4B, 0x03, 0x04 } } },
    { ".GIF", new List<byte[]> { new byte[] { 0x47, 0x49, 0x46, 0x38 } } }
};

public static bool IsValidFileExtension(string fileName, byte[] fileData, byte[] allowedChars)
{
    if (string.IsNullOrEmpty(fileName) || fileData == null || fileData.Length == 0)
    {
        return false;
    }

    bool flag = false;
    string ext = Path.GetExtension(fileName);
    if (string.IsNullOrEmpty(ext))
    {
        return false;
    }

    ext = ext.ToUpperInvariant();

    if (ext.Equals(".TXT") || ext.Equals(".CSV") || ext.Equals(".PRN"))
    {
        foreach (byte b in fileData)
        {
            if (b > 0x7F)
            {
                if (allowedChars != null)
                {
                    if (!allowedChars.Contains(b))
                    {
                        return false;
                    }
                }
                else
                {
                    return false;
                }
            }
        }

        return true;
    }

    if (!fileSignature.ContainsKey(ext))
    {
        return true;
    }

    List<byte[]> sig = fileSignature[ext];
    foreach (byte[] b in sig)
    {
        var curFileSig = new byte[b.Length];
        Array.Copy(fileData, curFileSig, b.Length);
        if (curFileSig.SequenceEqual(b))
        {
            return true;
        }
    }
}

```

```

        flag = true;
        break;
    }
}

return flag;
}

```

Ensure that type-safe parameters are used in Web Application for data access

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	If you use the Parameters collection, SQL treats the input is as a literal value rather than as executable code. The Parameters collection can be used to enforce type and length constraints on input data. Values outside of the range trigger an exception. If type-safe SQL parameters are not used, attackers might be able to execute injection attacks that are embedded in the unfiltered input. Use type safe parameters when constructing SQL queries to avoid possible SQL injection attacks that can occur with unfiltered input. You can use type safe parameters with stored procedures and with dynamic SQL statements. Parameters are treated as literal values by the database and not as executable code. Parameters are also checked for type and length.

Example

The following code shows how to use type safe parameters with the SqlParameterCollection when calling a stored procedure.

```

using System.Data;
using System.Data.SqlClient;

using (SqlConnection connection = new SqlConnection(connectionString))
{
    DataSet userDataset = new DataSet();
    SqlDataAdapter myCommand = new SqlDataAdapter(LoginStoredProcedure", connection);
    myCommand.SelectCommand.CommandType = CommandType.StoredProcedure;
    myCommand.SelectCommand.Parameters.Add("@au_id", SqlDbType.VarChar, 11);
    myCommand.SelectCommand.Parameters["@au_id"].Value = SSN.Text;
    myCommand.Fill(userDataset);
}

```

the type or length defined by the parameter, the SqlParameter class throws an exception.

Use separate model binding classes or binding filter lists to prevent MVC mass assignment vulnerability

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	MVC5, MVC6
Attributes	N/A
References	Metadata Attributes, Public Key Security Vulnerability And Mitigation, Complete Guide to Mass Assignment in ASP.NET MVC, Getting Started with EF using MVC
Steps	• When should I look for over-posting vulnerabilities? - Over-posting vulnerabilities can occur any place you bind model classes from user input. Frameworks like MVC can represent user data in custom .NET classes, including Plain Old CLR Objects (POCOs). MVC automatically populates these model classes with data from the request, providing a convenient representation for dealing with user input. When these classes include properties that should not be set by the user, the application can be vulnerable to over-posting attacks, which allow user control of data that the application never intended. Like MVC model binding, database access technologies such as object/relational mappers like Entity Framework often also support using POCO objects to represent database data. These data model classes provide the same convenience in dealing with database data as MVC does in dealing with user input. Because both MVC and the database support similar models, like POCO objects, it seems easy to reuse the same classes for both purposes. This practice fails to preserve separation of concerns, and it is one common area where unintended properties are exposed to model binding, enabling over-posting attacks. • Why shouldn't I use my unfiltered database model classes as parameters to my MVC actions? - Because MVC model binding will bind anything in that class. Even if the data does not appear in your view, a malicious user can send an HTTP request with this data included, and MVC will gladly bind it because your action says that database class is the shape of data it should accept for user input. • Why should I care about the shape used for model binding? - Using ASP.NET MVC model binding with overly broad models exposes an application to over-posting attacks. Over-posting could enable attackers to change application data beyond what the developer intended, such as overriding the price for an item or the security privileges for an account. Applications should use action-specific binding models (or specific allowed property filter lists) to provide solid

TITLE	DETAILS
	contract for what untrusted input to allow via model binding. • Is having separate binding models just duplicating code? - No, it is a matter of separation of concerns. If you reuse database models in action methods, you are saying any property (or sub-property) in that class can be set by the user in an HTTP request. If that is not what you want MVC to do, you need a filter list or a separate class shape to show MVC what data can come from user input instead. • If I have separate binding models for user input, do I have to duplicate all my data annotation attributes? - Not necessarily. You can use <code>MetadataTypeAttribute</code> on the database model class to link to the metadata on a model binding class. Just note that the type referenced by the <code>MetadataTypeAttribute</code> must be a subset of the referencing type (it can have fewer properties, but not more). • Moving data back and forth between user input models and database models is tedious. Can I just copy over all properties using reflection? - Yes. The only properties that appear in the binding models are the ones you have determined to be safe for user input. There is no security reason that prevents using reflection to copy over all properties that exist in common between these two models. • What about <code>[Bind(Exclude = "â€¦")]</code>. Can I use that instead of having separate binding models? - This approach is not recommended. Using <code>[Bind(Exclude = "â€¦")]</code> means that any new property is bindable by default. When a new property is added, there is an extra step to remember to keep things secure, rather than having the design be secure by default. Depending on the developer checking this list every time a property is added is risky. • Is <code>[Bind(Include = "â€¦")]</code> useful for Edit operations? - No. <code>[Bind(Include = "â€¦")]</code> is only suitable for INSERT-style operations (adding new data). For UPDATE-style operations (revising existing data), use another approach, like having separate binding models or passing an explicit list of allowed properties to <code>UpdateModel</code> or <code>TryUpdateModel</code>. Adding a <code>[Bind(Include = "â€¦")]</code> attribute on an Edit operation means that MVC will create an object instance and set only the listed properties, leaving all others at their default values. When the data is persisted, it will entirely replace the existing entity, resetting the values for any omitted properties to their defaults. For example, if <code>IsAdmin</code> was omitted from a <code>[Bind(Include = "â€¦")]</code> attribute on an Edit operation, any user whose name was edited via this action would be reset to <code>IsAdmin = false</code> (any edited user would lose administrator status). If you want to prevent updates to certain properties, use one of the other approaches above. Note that some versions of MVC tooling generate controller classes with <code>[Bind(Include = "â€¦")]</code> on Edit actions and imply that removing a property from that list will prevent over-posting attacks. However, as described above, that approach does not work as intended and instead will reset any data in the omitted properties to their default values. • For Create operations, are there any caveats to using <code>[Bind(Include = "â€¦")]</code> rather than separate models? - This is a risk. EX. 1002

TITLE	binding models? - Yes. First this approach does not work for Edit scenarios, requiring maintaining two separate approaches for mitigating all over-posting vulnerabilities. Second, separate binding models enforce separation of concerns between the shape used for user input and the shape used for persistence, something [Bind(Include = "â€¦")] does not do. Third, note that [Bind(Include = "â€¦")] can only handle top-level properties; you cannot allow only portions of sub-properties (such as "Details.Name") in the attribute. Finally, and perhaps most importantly, using [Bind(Include = "â€¦")] adds an extra step that must be remembered any time the class is used for model binding. If a new action method binds to the data class directly and forgets to include a [Bind(Include = "â€¦")] attribute, it can be vulnerable to over-posting attacks, so the [Bind(Include = "â€¦")] approach is somewhat less secure by default. If you use [Bind(Include = "â€¦")], take care always to remember to specify it every time your data classes appear as action method parameters.
	DETAILS separate approaches for mitigating all over-posting vulnerabilities. Second, separate binding models enforce separation of concerns between the shape used for user input and the shape used for persistence, something [Bind(Include = "â€¦")] does not do. Third, note that [Bind(Include = "â€¦")] can only handle top-level properties; you cannot allow only portions of sub-properties (such as "Details.Name") in the attribute. Finally, and perhaps most importantly, using [Bind(Include = "â€¦")] adds an extra step that must be remembered any time the class is used for model binding. If a new action method binds to the data class directly and forgets to include a [Bind(Include = "â€¦")] attribute, it can be vulnerable to over-posting attacks, so the [Bind(Include = "â€¦")] approach is somewhat less secure by default. If you use [Bind(Include = "â€¦")], take care always to remember to specify it every time your data classes appear as action method parameters. • For Create operations, what about putting the [Bind(Include = "â€¦")] attribute on the model class itself? Does not this approach avoid the need to remember putting the attribute on every action method? - This approach works in some cases. Using [Bind(Include = "â€¦")] on the model type itself (rather than on action parameters using this class), does avoid the need to remember to include the [Bind(Include = "â€¦")] attribute on every action method. Using the attribute directly on the class effectively creates a separate surface area of this class for model binding purposes. However, this approach only allows for one model binding shape per model class. If one action method needs to allow model binding of a field (for example, an administrator-only action that updates user roles) and other actions need to prevent model binding of this field, this approach will not work. Each class can only have one model binding shape; if different actions need different model binding shapes, they need to represent these separate shapes using either separate model binding classes or separate [Bind(Include = "â€¦")] attributes on the action methods. • What are binding models? Are they the same thing as view models? - These are two related concepts. The term binding model refers to a model class used in an action is parameter list (the shape passed from MVC model binding to the action method). The term view model refers to a model class passed from an action method to a view. Using a view-specific model is a common approach for passing data from an action method to a view. Often, this shape is also suitable for model binding, and the term view model can be used to refer the same model used in both places. To be precise, this procedure talks specifically about binding models, focusing on the shape passed to the action, which is what matters for mass assignment purposes.

Encode untrusted web output prior to rendering

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic, Web Forms, MVC5, MVC6
Attributes	N/A
References	How to prevent Cross-site scripting in ASP.NET, Cross-site Scripting, XSS (Cross Site Scripting) Prevention Cheat Sheet
Steps	Cross-site scripting (commonly abbreviated as XSS) is an attack vector for online services or any application/component that consumes input from the web. XSS vulnerabilities may allow an attacker to execute script on another user's machine through a vulnerable web application. Malicious scripts can be used to steal cookies and otherwise tamper with a victim's machine through JavaScript. XSS is prevented by validating user input, ensuring it is well formed and encoding before it is rendered in a web page. Input validation and output encoding can be done by using Web Protection Library. For Managed code (C#, VB.net, etc.), use one or more appropriate encoding methods from the Web Protection (Anti-XSS) Library, depending on the context where the user input gets manifested:

Example

```
* Encoder.HtmlEncode
* Encoder.HtmlAttributeEncode
* Encoder.JavaScriptEncode
* Encoder.UriEncode
* Encoder.VisualBasicScriptEncode
* Encoder.XmlEncode
* Encoder.XmlAttributeEncode
* Encoder.CssEncode
* Encoder.LdapEncode
```

Perform input validation and filtering on all string type Model properties

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic, MVC5, MVC6
Attributes	N/A

TITLE	DETAILS
References	Adding Validation, Validating Model Data in an MVC Application, Guiding Principles For Your ASP.NET MVC Applications
Steps	All the input parameters must be validated before they are used in the application to ensure that the application is safeguarded against malicious user inputs. Validate the input values using regular expression validations on server side with a whitelist validation strategy. Unsanitized user inputs / parameters passed to the methods can cause code injection vulnerabilities. For web applications, entry points can also include form fields, QueryStrings, cookies, HTTP headers, and web service parameters. The following input validation checks must be performed upon model binding: • The model properties should be annotated with RegularExpression annotation, for accepting allowed characters and maximum permissible length • The controller methods should perform ModelState validity

Sanitization should be applied on form fields that accept all characters, e.g, rich text editor

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Encode Unsafe Input, HTML Sanitizer

TITLE	DETAILS
Steps	Identify all static markup tags that you want to use. A common practice is to restrict formatting to safe HTML elements, such as <code></code> (bold) and <code><i></code> (italic). Before writing the data, HTML-encode it. This makes any malicious script safe by causing it to be handled as text, not as executable code. 1. Disable ASP.NET request validation by the adding the <code>ValidateRequest="false"</code> attribute to the <code>@ Page</code> directive 2. Encode the string input with the <code>HtmlEncode</code> method 3. Use a <code>StringBuilder</code> and call its <code>Replace</code> method to selectively remove the encoding on the HTML elements that you want to permit The page-in the references disables ASP.NET request validation by setting <code>ValidateRequest="false"</code>. It HTML-encodes the input and selectively allows the <code></code> and <code><i></code>. Alternatively, a .NET library for HTML sanitization may also be used. HtmlSanitizer is a .NET library for cleaning HTML fragments and documents from constructs that can lead to XSS attacks. It uses AngleSharp to parse, manipulate, and render HTML and CSS. HtmlSanitizer can be installed as a NuGet package, and the user input can be passed through relevant HTML or CSS sanitization methods, as applicable, on the server side. Please note that Sanitization as a security control should be considered only as a last option. Input validation and Output Encoding are considered better security controls.

Do not assign DOM elements to sinks that do not have inbuilt encoding

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Many javascript functions don't do encoding by default. When assigning untrusted input to DOM elements via such functions, may result in cross site script (XSS) executions.

Example

Following are insecure examples:

```
document.getElementById("div1").innerHTML = value;
$("#userName").html(res.Name);
return $('<div/>').html(value)
$('body').append(resHTML);
```

Don't use `innerHTML` ; instead use `innerText` . Similarly, instead of `$("#elm").html()` , use `$("#elm").text()`

Validate all redirects within the application are closed or done safely

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	The OAuth 2.0 Authorization Framework - Open Redirectors
Steps	Application design requiring redirection to a user-supplied location must constrain the possible redirection targets to a predefined "safe" list of sites or domains. All redirects in the application must be closed/safe. To do this: • Identify all redirects • Implement an appropriate mitigation for each redirect. Appropriate mitigations include redirect whitelist or user confirmation. If a web site or service with an open redirect vulnerability uses Facebook/OAuth/OpenID identity providers, an attacker can steal a user's logon token and impersonate that user. This is an inherent risk when using OAuth, which is documented in RFC 6749 "The OAuth 2.0 Authorization Framework", Section 10.15 "Open Redirects" Similarly, users' credentials can be compromised by spear phishing attacks using open redirects

Implement input validation on all string type parameters accepted by Controller methods

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic, MVC5, MVC6
Attributes	N/A

TITLE	DETAILS
References	Validating Model Data in an MVC Application, Guiding Principles For Your ASP.NET MVC Applications
Steps	For methods that just accept primitive data type, and not models as argument,input validation using Regular Expression should be done. Here Regex.IsMatch should be used with a valid regex pattern. If the input doesn't match the specified Regular Expression, control should not proceed further, and an adequate warning regarding validation failure should be displayed.

Set upper limit timeout for regular expression processing to prevent DoS due to bad regular expressions

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic, Web Forms, MVC5, MVC6
Attributes	N/A
References	DefaultRegexMatchTimeout Property
Steps	To ensure denial of service attacks against badly created regular expressions, that cause a lot of backtracking, set the global default timeout. If the processing time takes longer than the defined upper limit, it would throw a Timeout exception. If nothing is configured, the timeout would be infinite.

Example

For example, the following configuration will throw a RegexMatchTimeoutException, if the processing takes more than 5 seconds:

```
<httpRuntime targetFramework="4.5" defaultRegexMatchTimeout="00:00:05" />
```

Avoid using Html.Raw in Razor views

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	MVC5, MVC6
Attributes	N/A

TITLE	DETAILS
References	N/A
Step	ASP.Net WebPages (Razor) perform automatic HTML encoding. All strings printed by embedded code nuggets (@ blocks) are automatically HTML-encoded. However, when <code>HtmlHelper.Raw</code> Method is invoked, it returns markup that is not HTML encoded. If <code>Html.Raw()</code> helper method is used, it bypasses the automatic encoding protection that Razor provides.

Example

Following is an insecure example:

```
<div class="form-group">
    @Html.Raw(Model.AccountConfirmText)
</div>
<div class="form-group">
    @Html.Raw(Model.PaymentConfirmText)
</div>
</div>
```

Do not use `Html.Raw()` unless you need to display markup. This method does not perform output encoding implicitly. Use other ASP.NET helpers e.g., `@Html.DisplayFor()`

Do not use dynamic queries in stored procedures

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	A SQL injection attack exploits vulnerabilities in input validation to run arbitrary commands in the database. It can occur when your application uses input to construct dynamic SQL statements to access the database. It can also occur if your code uses stored procedures that are passed strings that contain raw user input. Using the SQL injection attack, the attacker can execute arbitrary commands in the database. All SQL statements (including the SQL statements in stored procedures) must be parameterized. Parameterized SQL statements will accept characters that have special meaning to SQL (like single quote) without problems because they are strongly typed.

Example

Following is an example of insecure dynamic Stored Procedure:

```

CREATE PROCEDURE [dbo].[uspGetProductsByCriteria]
(
    @productName nvarchar(200) = NULL,
    @startPrice float = NULL,
    @endPrice float = NULL
)
AS
BEGIN
    DECLARE @sql nvarchar(max)
    SELECT @sql = ' SELECT ProductID, ProductName, Description, UnitPrice, ImagePath' +
        ' FROM dbo.Products WHERE 1 = 1 '
    PRINT @sql
    IF @productName IS NOT NULL
        SELECT @sql = @sql + ' AND ProductName LIKE ''%' + @productName + '%''
    IF @startPrice IS NOT NULL
        SELECT @sql = @sql + ' AND UnitPrice > ''' + CONVERT(VARCHAR(10),@startPrice) + ''''
    IF @endPrice IS NOT NULL
        SELECT @sql = @sql + ' AND UnitPrice < ''' + CONVERT(VARCHAR(10),@endPrice) + ''''

    PRINT @sql
    EXEC(@sql)
END

```

Example

Following is the same stored procedure implemented securely:

```

CREATE PROCEDURE [dbo].[uspGetProductsByCriteriaSecure]
(
    @productName nvarchar(200) = NULL,
    @startPrice float = NULL,
    @endPrice float = NULL
)
AS
BEGIN
    SELECT ProductID, ProductName, Description, UnitPrice, ImagePath
    FROM dbo.Products where
        (@productName IS NULL or ProductName like '%' + @productName + '%')
        AND
        (@startPrice IS NULL or UnitPrice > @startPrice)
        AND
        (@endPrice IS NULL or UnitPrice < @endPrice)
END

```

Ensure that model validation is done on Web API methods

TITLE	DETAILS
Component	Web API
SDL Phase	Build
Applicable Technologies	MVC5, MVC6
Attributes	N/A
References	Model Validation in ASP.NET Web API

TITLE	DETAILS
Steps	When a client sends data to a web API, it is mandatory to validate the data before doing any processing. For ASP.NET Web APIs which accept models as input, use data annotations on models to set validation rules on the properties of the model.

Example

The following code demonstrates the same:

```
using System.ComponentModel.DataAnnotations;

namespace MyApi.Models
{
    public class Product
    {
        public int Id { get; set; }
        [Required]
        [RegularExpression(@"^[a-zA-Z0-9]*$", ErrorMessage="Only alphanumeric characters are allowed.")]
        public string Name { get; set; }
        public decimal Price { get; set; }
        [Range(0, 999)]
        public double Weight { get; set; }
    }
}
```

Example

In the action method of the API controllers, validity of the model has to be explicitly checked as shown below:

```
namespace MyApi.Controllers
{
    public class ProductsController : ApiController
    {
        public HttpResponseMessage Post(Product product)
        {
            if (ModelState.IsValid)
            {
                // Do something with the product (not shown).

                return new HttpResponseMessage(HttpStatusCode.OK);
            }
            else
            {
                return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ModelState);
            }
        }
    }
}
```

Implement input validation on all string type parameters accepted by Web API methods

TITLE	DETAILS
Component	Web API

TITLE	DETAILS
SDL Phase	Build
Applicable Technologies	Generic, MVC 5, MVC 6
Attributes	N/A
References	Validating Model Data in an MVC Application, Guiding Principles For Your ASP.NET MVC Applications
Steps	For methods that just accept primitive data type, and not models as argument, input validation using Regular Expression should be done. Here Regex.IsMatch should be used with a valid regex pattern. If the input doesn't match the specified Regular Expression, control should not proceed further, and an adequate warning regarding validation failure should be displayed.

Ensure that type-safe parameters are used in Web API for data access

TITLE	DETAILS
Component	Web API
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	If you use the Parameters collection, SQL treats the input is as a literal value rather than as executable code. The Parameters collection can be used to enforce type and length constraints on input data. Values outside of the range trigger an exception. If type-safe SQL parameters are not used, attackers might be able to execute injection attacks that are embedded in the unfiltered input. Use type safe parameters when constructing SQL queries to avoid possible SQL injection attacks that can occur with unfiltered input. You can use type safe parameters with stored procedures and with dynamic SQL statements. Parameters are treated as literal values by the database and not as executable code. Parameters are also checked for type and length.

Example

The following code shows how to use type safe parameters with the SqlParameterCollection when calling a stored procedure.

```

using System.Data;
using System.Data.SqlClient;

using (SqlConnection connection = new SqlConnection(connectionString))
{
    DataSet userDataset = new DataSet();
    SqlDataAdapter myCommand = new SqlDataAdapter(LoginStoredProcedure", connection);
    myCommand.SelectCommand.CommandType = CommandType.StoredProcedure;
    myCommand.SelectCommand.Parameters.Add("@au_id", SqlDbType.VarChar, 11);
    myCommand.SelectCommand.Parameters["@au_id"].Value = SSN.Text;
    myCommand.Fill(userDataset);
}

```

In the preceding code example, the input value cannot be longer than 11 characters. If the data does not conform to the type or length defined by the parameter, the SqlParameter class throws an exception.

Use parametrized SQL queries for Cosmos DB

TITLE	DETAILS
Component	Azure Document DB
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Announcing SQL Parameterization in Azure Cosmos DB
Steps	Although Azure Cosmos DB only supports read-only queries, SQL injection is still possible if queries are constructed by concatenating with user input. It might be possible for a user to gain access to data they shouldn't be accessing within the same collection by crafting malicious SQL queries. Use parameterized SQL queries if queries are constructed based on user input.

WCF Input validation through Schema binding

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	Generic, NET Framework 3
Attributes	N/A
References	MSDN

TITLE	DETAILS
Steps	Lack of validation leads to different type injection attacks. Message validation represents one line of defense in the protection of your WCF application. With this approach, you validate messages using schemas to protect WCF service operations from attack by a malicious client. Validate all messages received by the client to protect the client from attack by a malicious service. Message validation makes it possible to validate messages when operations consume message contracts or data contracts, which cannot be done using parameter validation. Message validation allows you to create validation logic inside schemas, thereby providing more flexibility and reducing development time. Schemas can be reused across different applications inside the organization, creating standards for data representation. Additionally, message validation allows you to protect operations when they consume more complex data types involving contracts representing business logic. To perform message validation, you first build a schema that represents the operations of your service and the data types consumed by those operations. You then create a .NET class that implements a custom client message inspector and custom dispatcher message inspector to validate the messages sent/received to/from the service. Next, you implement a custom endpoint behavior to enable message validation on both the client and the service. Finally, you implement a custom configuration element on the class that allows you to expose the extended custom endpoint behavior in the configuration file of the service or the client"

WCF- Input validation through Parameter Inspectors

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	Generic, NET Framework 3
Attributes	N/A
References	MSDN

TITLE	DETAILS
Steps	Input and data validation represents one important line of defense in the protection of your WCF application. You should validate all parameters exposed in WCF service operations to protect the service from attack by a malicious client. Conversely, you should also validate all return values received by the client to protect the client from attack by a malicious service WCF provides different extensibility points that allow you to customize the WCF runtime behavior by creating custom extensions. Message Inspectors and Parameter Inspectors are two extensibility mechanisms used to gain greater control over the data passing between a client and a service. You should use parameter inspectors for input validation and use message inspectors only when you need to inspect the entire message flowing in and out of a service. To perform input validation, you will build a .NET class and implement a custom parameter inspector in order to validate parameters on operations in your service. You will then implement a custom endpoint behavior to enable validation on both the client and the service. Finally, you will implement a custom configuration element on the class that allows you to expose the extended custom endpoint behavior in the configuration file of the service or the client

Security Frame: Sensitive Data | Mitigations

1/24/2018 • 16 min to read • [Edit Online](#)

PRODUCT/SERVICE	ARTICLE
Machine Trust Boundary	• Ensure that binaries are obfuscated if they contain sensitive information • Consider using Encrypted File System (EFS) is used to protect confidential user-specific data • Ensure that sensitive data stored by the application on the file system is encrypted
Web Application	• Ensure that sensitive content is not cached on the browser • Encrypt sections of Web App's configuration files that contain sensitive data • Explicitly disable the autocomplete HTML attribute in sensitive forms and inputs • Ensure that sensitive data displayed on the user screen is masked
Database	• Implement dynamic data masking to limit sensitive data exposure non privileged users • Ensure that passwords are stored in salted hash format • Ensure that sensitive data in database columns is encrypted • Ensure that database-level encryption (TDE) is enabled • Ensure that database backups are encrypted
Web API	• Ensure that sensitive data relevant to Web API is not stored in browser's storage
Azure Document DB	• Encrypt sensitive data stored in Azure Cosmos DB
Azure IaaS VM Trust Boundary	• Use Azure Disk Encryption to encrypt disks used by Virtual Machines
Service Fabric Trust Boundary	• Encrypt secrets in Service Fabric applications
Dynamics CRM	• Perform security modeling and use Business Units/Teams where required • Minimize access to share feature on critical entities • Train users on the risks associated with the Dynamics CRM Share feature and good security practices • Include a development standards rule proscribing showing config details in exception management

PRODUCT/SERVICE	ARTICLE
Azure Storage	• Use Azure Storage Service Encryption (SSE) for Data at Rest (Preview) • Use Client-Side Encryption to store sensitive data in Azure Storage
Mobile Client	• Encrypt sensitive or PII data written to phones local storage • Obfuscate generated binaries before distributing to end users
WCF	• Set clientCredentialType to Certificate or Windows • WCF-Security Mode is not enabled

Ensure that binaries are obfuscated if they contain sensitive information

TITLE	DETAILS
Component	Machine Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure that binaries are obfuscated if they contain sensitive information such as trade secrets, sensitive business logic that should not reversed. This is to stop reverse engineering of assemblies. Tools like <code>CryptoObfuscator</code> may be used for this purpose.

Consider using Encrypted File System (EFS) is used to protect confidential user-specific data

TITLE	DETAILS
Component	Machine Trust Boundary
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A

TITLE	DETAILS
Steps	Consider using Encrypted File System (EFS) is used to protect confidential user-specific data from adversaries with physical access to the computer.

Ensure that sensitive data stored by the application on the file system is encrypted

TITLE	DETAILS
Component	Machine Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Ensure that sensitive data stored by the application on the file system is encrypted (e.g., using DPAPI), if EFS cannot be enforced

Ensure that sensitive content is not cached on the browser

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic, Web Forms, MVC5, MVC6
Attributes	N/A
References	N/A
Steps	Browsers can store information for purposes of caching and history. These cached files are stored in a folder, like the Temporary Internet Files folder in the case of Internet Explorer. When these pages are referred again, the browser displays them from its cache. If sensitive information is displayed to the user (such as their address, credit card details, Social Security Number, or username), then this information could be stored in browser's cache, and therefore retrievable through examining the browser's cache or by simply pressing the browser's "Back" button. Set cache-control response header value to "no-store" for all pages.

Example

```

<configuration>
  <system.webServer>
    <httpProtocol>
      <customHeaders>
        <add name="Cache-Control" value="no-cache" />
        <add name="Pragma" value="no-cache" />
        <add name="Expires" value="-1" />
      </customHeaders>
    </httpProtocol>
  </system.webServer>
</configuration>

```

Example

This may be implemented through a filter. Following example may be used:

```

public override void OnActionExecuting(ActionExecutingContext filterContext)
{
    if (filterContext == null || (filterContext.HttpContext != null &&
filterContext.HttpContext.Response != null && filterContext.HttpContext.Response.IsRequestBeingRedirected))
    {
        //// Since this is MVC pipeline, this should never be null.
        return;
    }

    var attributes =
filterContext.ActionDescriptor.GetCustomAttributes(typeof(System.Web.Mvc.OutputCacheAttribute), false);
    if (attributes == null || **Attributes**.Count() == 0)
    {
        filterContext.HttpContext.Response.Cache.SetNoStore();
        filterContext.HttpContext.Response.Cache.SetCacheability(HttpCacheability.NoCache);
        filterContext.HttpContext.Response.Cache.SetExpires(DateTime.UtcNow.AddHours(-1));
        if (!filterContext.IsChildAction)
        {
            filterContext.HttpContext.Response.AppendHeader("Pragma", "no-cache");
        }
    }

    base.OnActionExecuting(filterContext);
}

```

Encrypt sections of Web App's configuration files that contain sensitive data

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	How To: Encrypt Configuration Sections in ASP.NET 2.0 Using DPAPI, Specifying a Protected Configuration Provider, Using Azure Key Vault to protect application secrets

TITLE	DETAILS
Steps	Configuration files such as the Web.config, appsettings.json are often used to hold sensitive information, including user names, passwords, database connection strings, and encryption keys. If you do not protect this information, your application is vulnerable to attackers or malicious users obtaining sensitive information such as account user names and passwords, database names and server names. Based on the deployment type (azure/on-prem), encrypt the sensitive sections of config files using DPAPI or services like Azure Key Vault.

Explicitly disable the autocomplete HTML attribute in sensitive forms and inputs

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	MSDN: autocomplete attribute, Using AutoComplete in HTML, HTML Sanitization Vulnerability, Autocomplete, again?!
Steps	The autocomplete attribute specifies whether a form should have autocomplete on or off. When autocomplete is on, the browser automatically complete values based on values that the user has entered before. For example, when a new name and password is entered in a form and the form is submitted, the browser asks if the password should be saved. Thereafter when the form is displayed, the name and password are filled in automatically or are completed as the name is entered. An attacker with local access could obtain the clear text password from the browser cache. By default autocomplete is enabled, and it must explicitly be disabled.

Example

```
<form action="Login.aspx" method="post " autocomplete="off" >
  Social Security Number: <input type="text" name="ssn" />
  <input type="submit" value="Submit" />
</form>
```

Ensure that sensitive data displayed on the user screen is masked

TITLE	DETAILS
Component	Web Application

TITLE	DETAILS
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Sensitive data such as passwords, credit card numbers, SSN etc. should be masked when displayed on the screen. This is to prevent unauthorized personnel from accessing the data (e.g., shoulder-surfing passwords, support personnel viewing SSN numbers of users) . Ensure that these data elements are not visible in plain text and are appropriately masked. This has to be taken care while accepting them as input (e.g., input type="password") as well as displaying back on the screen (e.g., display only the last 4 digits of the credit card number).

Implement dynamic data masking to limit sensitive data exposure non privileged users

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	Sql Azure, OnPrem
Attributes	SQL Version - V12, SQL Version - MsSQL2016
References	Dynamic Data Masking
Steps	The purpose of dynamic data masking is to limit exposure of sensitive data, preventing users who should not have access to the data from viewing it. Dynamic data masking does not aim to prevent database users from connecting directly to the database and running exhaustive queries that expose pieces of the sensitive data. Dynamic data masking is complementary to other SQL Server security features (auditing, encryption, row level security...) and it is highly recommended to use this feature in conjunction with them in addition in order to better protect the sensitive data in the database. Please note that this feature is supported only by SQL Server starting with 2016 and Azure SQL Database.

Ensure that passwords are stored in salted hash format

TITLE	DETAILS
Component	Database

TITLE	DETAILS
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Password Hashing using .NET Crypto APIs
Steps	Passwords should not be stored in custom user store databases. Password hashes should be stored with salt values instead. Make sure the salt for the user is always unique and you apply b-crypt, s-crypt or PBKDF2 before storing the password, with a minimum work factor iteration count of 150,000 loops to eliminate the possibility of brute forcing.

Ensure that sensitive data in database columns is encrypted

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	Generic
Attributes	SQL Version - All
References	Encrypting sensitive data in SQL server, How to: Encrypt a Column of Data in SQL Server, Encrypt by Certificate
Steps	Sensitive data such as credit card numbers has to be encrypted in the database. Data can be encrypted using column-level encryption or by an application function using the encryption functions.

Ensure that database-level encryption (TDE) is enabled

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Understanding SQL Server Transparent Data Encryption (TDE)

TITLE	DETAILS
Steps	Transparent Data Encryption (TDE) feature in SQL server helps in encrypting sensitive data in a database and protect the keys that are used to encrypt the data with a certificate. This prevents anyone without the keys from using the data. TDE protects data "at rest", meaning the data and log files. It provides the ability to comply with many laws, regulations, and guidelines established in various industries.

Ensure that database backups are encrypted

TITLE	DETAILS
Component	Database
SDL Phase	Build
Applicable Technologies	SQL Azure, OnPrem
Attributes	SQL Version - V12, SQL Version - MsSQL2014
References	SQL database backup encryption
Steps	SQL Server has the ability to encrypt the data while creating a backup. By specifying the encryption algorithm and the encryptor (a Certificate or Asymmetric Key) when creating a backup, one can create an encrypted backup file.

Ensure that sensitive data relevant to Web API is not stored in browser's storage

TITLE	DETAILS
Component	Web API
SDL Phase	Build
Applicable Technologies	MVC 5, MVC 6
Attributes	Identity Provider - ADFS, Identity Provider - Azure AD
References	N/A

TITLE	DETAILS
Steps	In certain implementations, sensitive artifacts relevant to Web API's authentication are stored in browser's local storage. E.g., Azure AD authentication artifacts like <code>adal.idtoken</code>, <code>adal.nonce.idtoken</code>, <code>adal.access.token.key</code>, <code>adal.token.keys</code>, <code>adal.state.login</code>, <code>adal.session.state</code>, <code>adal.expiration.key</code> etc. All these artifacts are available even after sign out or browser is closed. If an adversary gets access to these artifacts, he/she can reuse them to access the protected resources (APIs). Ensure that all sensitive artifacts related to Web API is not stored in browser's storage. In cases where client-side storage is unavoidable (e.g., Single Page Applications (SPA) that leverage Implicit OpenIdConnect/OAuth flows need to store access tokens locally), use storage choices which do not have persistence. e.g., prefer <code>SessionStorage</code> to <code>LocalStorage</code>.

Example

The below JavaScript snippet is from a custom authentication library which stores authentication artifacts in local storage. Such implementations should be avoided.

```
ns.AuthHelper.Authenticate = function () {
window.config = {
instance: 'https://login.microsoftonline.com/',
tenant: ns.Configurations.Tenant,
clientId: ns.Configurations.AADApplicationClientID,
postLogoutRedirectUri: window.location.origin,
cacheLocation: 'localStorage', // enable this for IE, as sessionStorage does not work for localhost.
};
```

Encrypt sensitive data stored in Cosmos DB

TITLE	DETAILS
Component	Azure Document DB
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Encrypt sensitive data at application level before storing in document DB or store any sensitive data in other storage solutions like Azure Storage or Azure SQL

Use Azure Disk Encryption to encrypt disks used by Virtual Machines

TITLE	DETAILS
Component	Azure IaaS VM Trust Boundary
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	Using Azure Disk Encryption to encrypt disks used by your virtual machines
Steps	Azure Disk Encryption is a new feature that is currently in preview. This feature allows you to encrypt the OS disks and Data disks used by an IaaS Virtual Machine. For Windows, the drives are encrypted using industry-standard BitLocker encryption technology. For Linux, the disks are encrypted using the DM-Crypt technology. This is integrated with Azure Key Vault to allow you to control and manage the disk encryption keys. The Azure Disk Encryption solution supports the following three customer encryption scenarios: • Enable encryption on new IaaS VMs created from customer-encrypted VHD files and customer-provided encryption keys, which are stored in Azure Key Vault. • Enable encryption on new IaaS VMs created from the Azure Marketplace. • Enable encryption on existing IaaS VMs already running in Azure.

Encrypt secrets in Service Fabric applications

TITLE	DETAILS
Component	Service Fabric Trust Boundary
SDL Phase	Build
Applicable Technologies	Generic
Attributes	Environment - Azure
References	Managing secrets in Service Fabric applications
Steps	Secrets can be any sensitive information, such as storage connection strings, passwords, or other values that should not be handled in plain text. Use Azure Key Vault to manage keys and secrets in service fabric applications.

Perform security modeling and use Business Units/Teams where required

TITLE	DETAILS
Component	Dynamics CRM
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Perform security modeling and use Business Units/Teams where required

Minimize access to share feature on critical entities

TITLE	DETAILS
Component	Dynamics CRM
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Minimize access to share feature on critical entities

Train users on the risks associated with the Dynamics CRM Share feature and good security practices

TITLE	DETAILS
Component	Dynamics CRM
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Train users on the risks associated with the Dynamics CRM Share feature and good security practices

Include a development standards rule proscribing showing configuration

details in exception management

TITLE	DETAILS
Component	Dynamics CRM
SDL Phase	Deployment
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Include a development standards rule proscribing showing config details in exception management outside development. Test for this as part of code reviews or periodic inspection.

Use Azure Storage Service Encryption (SSE) for Data at Rest (Preview)

TITLE	DETAILS
Component	Azure Storage
SDL Phase	Build
Applicable Technologies	Generic
Attributes	StorageType - Blob
References	Azure Storage Service Encryption for Data at Rest (Preview)

TITLE	DETAILS
Steps	Azure Storage Service Encryption (SSE) for Data at Rest helps you protect and safeguard your data to meet your organizational security and compliance commitments. With this feature, Azure Storage automatically encrypts your data prior to persisting to storage and decrypts prior to retrieval. The encryption, decryption and key management is totally transparent to users. SSE applies only to block blobs, page blobs, and append blobs. The other types of data, including tables, queues, and files, will not be encrypted. Encryption and Decryption Workflow: • The customer enables encryption on the storage account • When the customer writes new data (PUT Blob, PUT Block, PUT Page, etc.) to Blob storage, every write is encrypted using 256-bit AES encryption, one of the strongest block ciphers available • When the customer needs to access data (GET Blob, etc.), data is automatically decrypted before returning to the user • If encryption is disabled, new writes are no longer encrypted and existing encrypted data remains encrypted until rewritten by the user. While encryption is enabled, writes to Blob storage will be encrypted. The state of data does not change with the user toggling between enabling/disabling encryption for the storage account • All encryption keys are stored, encrypted, and managed by Microsoft Please note that at this time, the keys used for the encryption are managed by Microsoft. Microsoft generates the keys originally, and manage the secure storage of the keys as well as the regular rotation as defined by internal Microsoft policy. In the future, customers will get the ability to manage their own > encryption keys, and provide a migration path from Microsoft-managed keys to customer-managed keys.

Use Client-Side Encryption to store sensitive data in Azure Storage

TITLE	DETAILS
Component	Azure Storage
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Client-Side Encryption and Azure Key Vault for Microsoft Azure Storage, Tutorial: Encrypt and decrypt blobs in Microsoft Azure Storage using Azure Key Vault, Storing Data Securely in Azure Blob Storage with Azure Encryption Extension

TITLE	DETAILS
Steps	The Azure Storage Client Library for .NET Nuget package supports encrypting data within client applications before uploading to Azure Storage, and decrypting data while downloading to the client. The library also supports integration with Azure Key Vault for storage account key management. Here is a brief description of how client side encryption works: • The Azure Storage client SDK generates a content encryption key (CEK), which is a one-time-use symmetric key • Customer data is encrypted using this CEK • The CEK is then wrapped (encrypted) using the key encryption key (KEK). The KEK is identified by a key identifier and can be an asymmetric key pair or a symmetric key and can be managed locally or stored in Azure Key Vault. The Storage client itself never has access to the KEK. It just invokes the key wrapping algorithm that is provided by Key Vault. Customers can choose to use custom providers for key wrapping/unwrapping if they want • The encrypted data is then uploaded to the Azure Storage service. Check the links in the references section for low-level implementation details.

Encrypt sensitive or PII data written to phones local storage

TITLE	DETAILS
Component	Mobile Client
SDL Phase	Build
Applicable Technologies	Generic, Xamarin
Attributes	N/A
References	Manage settings and features on your devices with Microsoft Intune policies, Keychain Valet
Steps	If the application writes sensitive information like user's PII (email, phone number, first name, last name, preferences etc.)- on mobile's file system, then it should be encrypted before writing to the local file system. If the application is an enterprise application, then explore the possibility of publishing application using Windows Intune.

Example

Intune can be configured with following security policies to safeguard sensitive data:

Require encryption on mobile device
Require encryption on storage cards
Allow screen capture

Example

If the application is not an enterprise application, then use platform provided keystore, keychains to store encryption keys, using which cryptographic operation may be performed on the file system. Following code snippet shows how to access key from keychain using xamarin:

```
protected static string EncryptionKey
{
    get
    {
        if (String.IsNullOrEmpty(_Key))
        {
            var query = new SecRecord(SecKind.GenericPassword);
            query.Service = NSBundle.MainBundle.BundleIdentifier;
            query.Account = "UniqueID";

            NSData uniqueId = SecKeyChain.QueryAsData(query);
            if (uniqueId == null)
            {
                query.ValueData = NSData.FromString(System.Guid.NewGuid().ToString());
                var err = SecKeyChain.Add(query);
                _Key = query.ValueData.ToString();
            }
            else
            {
                _Key = uniqueId.ToString();
            }
        }

        return _Key;
    }
}
```

Obfuscate generated binaries before distributing to end users

TITLE	DETAILS
Component	Mobile Client
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Crypto Obfuscation For .Net
Steps	Generated binaries (assemblies within apk) should be obfuscated to stop reverse engineering of assemblies.Tools like CryptoObfuscator may be used for this purpose.

Set clientCredentialType to Certificate or Windows

TITLE	DETAILS
Component	WCF

TITLE	DETAILS
SDL Phase	Build
Applicable Technologies	.NET Framework 3
Attributes	N/A
References	Fortify
Steps	Using a UsernameToken with a plaintext password over an unencrypted channel exposes the password to attackers who can sniff the SOAP messages. Service Providers that use the UsernameToken might accept passwords sent in plaintext. Sending plaintext passwords over an unencrypted channel can expose the credential to attackers who can sniff the SOAP message.

Example

The following WCF service provider configuration uses the UsernameToken:

```
<security mode="Message">
  <message clientCredentialType="UserName" />
```

Set clientCredentialType to Certificate or Windows.

WCF-Security Mode is not enabled

TITLE	DETAILS
Component	WCF
SDL Phase	Build
Applicable Technologies	Generic, .NET Framework 3
Attributes	Security Mode - Transport, Security Mode - Message
References	MSDN, Fortify Kingdom, Fundamentals of WCF Security CoDe Magazine
Steps	No transport or message security has been defined. Applications that transmit messages without transport or message security cannot guarantee the integrity or confidentiality of the messages. When a WCF security binding is set to None, both transport and message security are disabled.

Example

The following configuration sets the security mode to None.

```

<system.serviceModel>
  <bindings>
    <wsHttpBinding>
      <binding name="MyBinding">
        <security mode="None"/>
      </binding>
    </bindings>
  </system.serviceModel>

```

Example

Security Mode Across all service bindings there are five possible security modes:

- None. Turns security off.
- Transport. Uses transport security for mutual authentication and message protection.
- Message. Uses message security for mutual authentication and message protection.
- Both. Allows you to supply settings for transport and message-level security (only MSMQ supports this).
- TransportWithMessageCredential. Credentials are passed with the message and message protection and server authentication are provided by the transport layer.
- TransportCredentialOnly. Client credentials are passed with the transport layer and no message protection is applied. Use transport and message security to protect the integrity and confidentiality of messages. The configuration below tells the service to use transport security with message credentials.

```

<system.serviceModel> <bindings> <wsHttpBinding> <binding name="MyBinding"> <security
mode="TransportWithMessageCredential"/> <message clientCredentialType="Windows"/> </binding> </bindings>
</system.serviceModel>

```

Security Frame: Session Management | Articles

1/24/2018 • 14 min to read • [Edit Online](#)

PRODUCT/SERVICE	ARTICLE
Azure AD	• Implement proper logout using ADAL methods when using Azure AD
IoT Device	• Use finite lifetimes for generated SaS tokens
Azure Document DB	• Use minimum token lifetimes for generated Resource tokens
ADFS	• Implement proper logout using WsFederation methods when using ADFS
Identity Server	• Implement proper logout when using Identity Server
Web Application	• Applications available over HTTPS must use secure cookies • All http based application should specify http only for cookie definition • Mitigate against Cross-Site Request Forgery (CSRF) attacks on ASP.NET web pages • Set up session for inactivity lifetime • Implement proper logout from the application
Web API	• Mitigate against Cross-Site Request Forgery (CSRF) attacks on ASP.NET Web APIs

Implement proper logout using ADAL methods when using Azure AD

TITLE	DETAILS
Component	Azure AD
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A

TITLE	DETAILS
Steps	If the application relies on access token issued by Azure AD, the logout event handler should call

Example

```
HttpContext.GetOwinContext().Authentication.SignOut(OpenIdConnectAuthenticationDefaults.AuthenticationType,
CookieAuthenticationDefaults.AuthenticationType)
```

Example

It should also destroy user's session by calling Session.Abandon() method. Following method shows secure implementation of user logout:

```
[HttpPost]
[ValidateAntiForgeryToken]
public void LogOff()
{
    string userObjectID =
ClaimsPrincipal.Current.FindFirst("http://schemas.microsoft.com/identity/claims/objectidentifier").Value;
    AuthenticationContext authContext = new AuthenticationContext(Authority + TenantId, new
NaiveSessionCache(userObjectID));
    authContext.TokenCache.Clear();
    Session.Clear();
    Session.Abandon();
    Response.SetCookie(new HttpCookie("ASP.NET_SessionId", string.Empty));
    HttpContext.GetOwinContext().Authentication.SignOut(
        OpenIdConnectAuthenticationDefaults.AuthenticationType,
        CookieAuthenticationDefaults.AuthenticationType);
}
```

Use finite lifetimes for generated SaS tokens

TITLE	DETAILS
Component	IoT Device
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	SaS tokens generated for authenticating to Azure IoT Hub should have a finite expiry period. Keep the SaS token lifetimes to a minimum to limit the amount of time they can be replayed in case the tokens are compromised.

Use minimum token lifetimes for generated Resource tokens

TITLE	DETAILS
Component	Azure Document DB
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Reduce the timespan of resource token to a minimum value required. Resource tokens have a default valid timespan of 1 hour.

Implement proper logout using WsFederation methods when using ADFS

TITLE	DETAILS
Component	ADFS
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	If the application relies on STS token issued by ADFS, the logout event handler should call <code>WSFederationAuthenticationModule.FederatedSignOut()</code> method to log out the user. Also the current session should be destroyed, and the session token value should be reset and nullified.

Example

```

[HttpPost, ValidateAntiForgeryToken]
[Authorization]
public ActionResult SignOut(string redirectUrl)
{
    if (!this.User.Identity.IsAuthenticated)
    {
        return this.View("LogOff", null);
    }

    // Removes the user profile.
    this.Session.Clear();
    this.Session.Abandon();
    HttpContext.Current.Response.Cookies.Add(new System.Web.HttpCookie("ASP.NET_SessionId",
string.Empty)
    {
        Expires = DateTime.Now.AddDays(-1D),
        Secure = true,
        HttpOnly = true
    }));

    // Signs out at the specified security token service (STS) by using the WS-Federation protocol.
    Uri signOutUrl = new Uri(FederatedAuthentication.WSFederationAuthenticationModule.Issuer);
    Uri replyUrl = new Uri(FederatedAuthentication.WSFederationAuthenticationModule.Realm);
    if (!string.IsNullOrEmpty(redirectUrl))
    {
        replyUrl = new Uri(FederatedAuthentication.WSFederationAuthenticationModule.Realm +
redirectUrl);
    }
    // Signs out of the current session and raises the appropriate events.
    var authModule = FederatedAuthentication.WSFederationAuthenticationModule;
    authModule.SignOut(false);
    // Signs out at the specified security token service (STS) by using the WS-Federation
    // protocol.
    WSFederationAuthenticationModule.FederatedSignOut(signOutUrl, replyUrl);
    return new RedirectResult(redirectUrl);
}

```

Implement proper logout when using Identity Server

TITLE	DETAILS
Component	Identity Server
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	IdentityServer3-Federated sign out
Steps	IdentityServer supports the ability to federate with external identity providers. When a user signs out of an upstream identity provider, depending upon the protocol used, it might be possible to receive a notification when the user signs out. It allows IdentityServer to notify its clients so they can also sign the user out. Check the documentation in the references section for the implementation details.

Applications available over HTTPS must use secure cookies

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	EnvironmentType - OnPrem
References	httpCookies Element (ASP.NET Settings Schema), HttpCookie.Secure Property
Steps	Cookies are normally only accessible to the domain for which they were scoped. Unfortunately, the definition of "domain" does not include the protocol so cookies that are created over HTTPS are accessible over HTTP. The "secure" attribute indicates to the browser that the cookie should only be made available over HTTPS. Ensure that all cookies set over HTTPS use the secure attribute. The requirement can be enforced in the web.config file by setting the requireSSL attribute to true. It is the preferred approach because it will enforce the secure attribute for all current and future cookies without the need to make any additional code changes.

Example

```
<configuration>
  <system.web>
    <httpCookies requireSSL="true"/>
  </system.web>
</configuration>
```

The setting is enforced even if HTTP is used to access the application. If HTTP is used to access the application, the setting breaks the application because the cookies are set with the secure attribute and the browser will not send them back to the application.

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Web Forms, MVC5
Attributes	EnvironmentType - OnPrem
References	N/A
Steps	When the web application is the Relying Party, and the IdP is ADFS server, the FedAuth token's secure attribute can be configured by setting requireSSL to True in <code>system.identityModel.services</code> section of web.config

Example

```
<system.identityModel.services>
  <federationConfiguration>
    <!-- Set requireSsl=true; domain=application domain name used by FedAuth cookies (Ex: .gdinfra.com); -->
    <cookieHandler requireSsl="true" persistentSessionLifetime="0.0:20:0" />
    ....
  </federationConfiguration>
</system.identityModel.services>
```

All http based application should specify http only for cookie definition

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	Secure Cookie Attribute
Steps	To mitigate the risk of information disclosure with a cross-site scripting (XSS) attack, a new attribute - httpOnly - was introduced to cookies and is supported by all major browsers. The attribute specifies that a cookie is not accessible through script. By using HttpOnly cookies, a web application reduces the possibility that sensitive information contained in the cookie can be stolen via script and sent to an attacker's website.

Example

All HTTP-based applications that use cookies should specify HttpOnly in the cookie definition, by implementing following configuration in web.config:

```
<system.web>
.
.
  <httpCookies requireSSL="false" httpOnlyCookies="true"/>
.
.
</system.web>
```

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Web Forms
Attributes	N/A

TITLE	DETAILS
References	FormsAuthentication.RequireSSL Property
Steps	The RequireSSL property value is set in the configuration file for an ASP.NET application by using the requireSSL attribute of the configuration element. You can specify in the Web.config file for your ASP.NET application whether SSL (Secure Sockets Layer) is required to return the forms-authentication cookie to the server by setting the requireSSL attribute.

Example

The following code example sets the requireSSL attribute in the Web.config file.

```
<authentication mode="Forms">
  <forms loginUrl="member_login.aspx" cookieless="UseCookies" requireSSL="true"/>
</authentication>
```

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	MVC5
Attributes	EnvironmentType - OnPrem
References	Windows Identity Foundation (WIF) Configuration – Part II
Steps	To set httpOnly attribute for FedAuth cookies, hideFromCsript attribute value should be set to True.

Example

Following configuration shows the correct configuration:

```
<federatedAuthentication>
  <cookieHandler mode="Custom"
    hideFromScript="true"
    name="FedAuth"
    path="/"
    requireSsl="true"
    persistentSessionLifetime="25">
  </cookieHandler>
</federatedAuthentication>
```

Mitigate against Cross-Site Request Forgery (CSRF) attacks on ASP.NET web pages

TITLE	DETAILS
Component	Web Application

TITLE	DETAILS
SDL Phase	Build
Applicable Technologies	Generic
Attributes	N/A
References	N/A
Steps	Cross-site request forgery (CSRF or XSRF) is a type of attack in which an attacker can carry out actions in the security context of a different user's established session on a web site. The goal is to modify or delete content, if the targeted web site relies exclusively on session cookies to authenticate received request. An attacker could exploit this vulnerability by getting a different user's browser to load a URL with a command from a vulnerable site on which the user is already logged in. There are many ways for an attacker to do that, such as by hosting a different web site that loads a resource from the vulnerable server, or getting the user to click a link. The attack can be prevented if the server sends an additional token to the client, requires the client to include that token in all future requests, and verifies that all future requests include a token that pertains to the current session, such as by using the ASP.NET AntiForgeryToken or ViewState.

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	MVC5, MVC6
Attributes	N/A
References	XSRF/CSRF Prevention in ASP.NET MVC and Web Pages
Steps	Anti-CSRF and ASP.NET MVC forms - Use the <code>AntiForgeryToken</code> helper method on Views; put an <code>Html.AntiForgeryToken()</code> into the form, for example,

Example

```
@using (Html.BeginForm("UserProfile", "SubmitUpdate")) {
    @Html.ValidationSummary(true)
    @Html.AntiForgeryToken()
    <fieldset>
```

Example

```
<form action="/UserProfile/SubmitUpdate" method="post">
  <input name="__RequestVerificationToken" type="hidden"
value="saTFWpkKN08YazFtN6c4YbZamsEwG0srqlUqqloi/fVgeV2ciIFVmelvzwRZpAns" />
  <!-- rest of form goes here -->
</form>
```

Example

At the same time, `Html.AntiForgeryToken()` gives the visitor a cookie called `__RequestVerificationToken`, with the same value as the random hidden value shown above. Next, to validate an incoming form post, add the `[ValidateAntiForgeryToken]` filter to the target action method. For example:

```
[ValidateAntiForgeryToken]
public ViewResult SubmitUpdate()
{
  // ... etc.
}
```

Authorization filter that checks that:

- The incoming request has a cookie called `__RequestVerificationToken`
- The incoming request has a `Request.Form` entry called `__RequestVerificationToken`
- These cookie and `Request.Form` values match Assuming all is well, the request goes through as normal. But if not, then an authorization failure with message "A required anti-forgery token was not supplied or was invalid".

Example

Anti-CSRF and AJAX: The form token can be a problem for AJAX requests, because an AJAX request might send JSON data, not HTML form data. One solution is to send the tokens in a custom HTTP header. The following code uses Razor syntax to generate the tokens, and then adds the tokens to an AJAX request.

```
<script>
  @functions{
    public string TokenHeaderValue()
    {
      string cookieToken, formToken;
      AntiForgery.GetTokens(null, out cookieToken, out formToken);
      return cookieToken + ":" + formToken;
    }
  }

  $.ajax("api/values", {
    type: "post",
    contentType: "application/json",
    data: { }, // JSON data goes here
    dataType: "json",
    headers: {
      'RequestVerificationToken': '@TokenHeaderValue()'
    }
  });
</script>
```

Example

When you process the request, extract the tokens from the request header. Then call the `AntiForgery.Validate` method to validate the tokens. The `Validate` method throws an exception if the tokens are not valid.

```

void ValidateRequestHeader(HttpRequestMessage request)
{
    string cookieToken = "";
    string formToken = "";

    IEnumerable<string> tokenHeaders;
    if (request.Headers.TryGetValues("RequestVerificationToken", out tokenHeaders))
    {
        string[] tokens = tokenHeaders.First().Split(':');
        if (tokens.Length == 2)
        {
            cookieToken = tokens[0].Trim();
            formToken = tokens[1].Trim();
        }
    }
    AntiForgery.Validate(cookieToken, formToken);
}

```

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Web Forms
Attributes	N/A
References	Take Advantage of ASP.NET Built-in Features to Fend Off Web Attacks
Steps	CSRF attacks in WebForm based applications can be mitigated by setting ViewStateUserKey to a random string that varies for each user - user ID or, better yet, session ID. For a number of technical and social reasons, session ID is a much better fit because a session ID is unpredictable, times out, and varies on a per-user basis.

Example

Here's the code you need to have in all of your pages:

```

void Page_Init (object sender, EventArgs e) {
    ViewStateUserKey = Session.SessionID;
    :
}

```

Set up session for inactivity lifetime

TITLE	DETAILS
Component	Web Application
SDL Phase	Build
Applicable Technologies	Generic

TITLE	DETAILS
Attributes	N/A
References	HttpSessionState.Timeout Property
Steps	Session timeout represents the event occurring when a user does not perform any action on a web site during a interval (defined by web server). The event, on server side, change the status of the user session to 'invalid' (for example "not used anymore") and instruct the web server to destroy it (deleting all data contained into it). The following code example sets the timeout session attribute to 15 minutes in the Web.config file.

Example

```XML code <system.web> </system.web>

|                                                                      |
|----------------------------------------------------------------------|
| ## <a id="threat-detection"></a>Enable Threat detection on Azure SQL |
|----------------------------------------------------------------------|

| TITLE                          | DETAILS                                                          |
|--------------------------------|------------------------------------------------------------------|
| <b>Component</b>               | Web Application                                                  |
| <b>SDL Phase</b>               | Build                                                            |
| <b>Applicable Technologies</b> | Web Forms                                                        |
| <b>Attributes</b>              | N/A                                                              |
| <b>References</b>              | forms Element for authentication (ASP.NET Settings Schema)       |
| <b>Steps</b>                   | Set the Forms Authentication Ticket cookie timeout to 15 minutes |

#### Example

```XML code

| Title | Details |
|--|--|
| ----- | ----- |
| **Component** | Web Application |
| **SDL Phase** | Build |
| **Applicable Technologies** | Web Forms, MVC5 |
| **Attributes** | EnvironmentType - OnPrem |
| **References** | [asdeqa](https://skf.azurewebsites.net/Mitigations/Details/wefr) |
| **Steps** When the web application is Relying Party and ADFS is the STS, the lifetime of the authentication cookies - FedAuth tokens - can be set by the following configuration in web.config: | |

Example

```

<<<XML
<system.identityModel.services>
  <federationConfiguration>
    <!-- Set requireSsl=true; domain=application domain name used by FedAuth cookies (Ex: .gdinfra.com); -->
    <cookieHandler requireSsl="true" persistentSessionLifetime="0.0:15:0" />
    <!-- Set requireHttps=true; -->
    <wsFederation passiveRedirectEnabled="true" issuer="http://localhost:39529/"
realm="https://localhost:44302/" reply="https://localhost:44302/" requireHttps="true"/>
    <!--
      Use the code below to enable encryption-decryption of claims received from ADFS. Thumbprint value varies
      based on the certificate being used.
    <serviceCertificate>
      <certificateReference findValue="4FBBBA33A1D11A9022A5BF3492FF83320007686A" storeLocation="LocalMachine"
storeName="My" x509FindType="FindByThumbprint" />
    </serviceCertificate>
    -->
  </federationConfiguration>
</system.identityModel.services>

```

Example

Also the ADFS issued SAML claim token's lifetime should be set to 15 minutes, by executing the following powershell command on the ADFS server:

```

Set-ADFSRelyingPartyTrust -TargetName "<RelyingPartyWebApp>" -ClaimsProviderName @("Active Directory") -
TokenLifetime 15 -AlwaysRequireAuthentication $true

```

Implement proper logout from the application

| TITLE | DETAILS |
|--------------------------------|-----------------|
| Component | Web Application |
| SDL Phase | Build |
| Applicable Technologies | Generic |
| Attributes | N/A |
| References | N/A |

| TITLE | DETAILS |
|--------------|---|
| Steps | Perform proper Sign Out from the application, when user presses log out button. Upon logout, application should destroy user's session, and also reset and nullify session cookie value, along with resetting and nullifying authentication cookie value. Also, when multiple sessions are tied to a single user identity, they must be collectively terminated on the server side at timeout or logout. Lastly, ensure that Logout functionality is available on every page. |

Mitigate against Cross-Site Request Forgery (CSRF) attacks on ASP.NET Web APIs

| TITLE | DETAILS |
|--------------------------------|--|
| Component | Web API |
| SDL Phase | Build |
| Applicable Technologies | Generic |
| Attributes | N/A |
| References | N/A |
| Steps | Cross-site request forgery (CSRF or XSRF) is a type of attack in which an attacker can carry out actions in the security context of a different user's established session on a web site. The goal is to modify or delete content, if the targeted web site relies exclusively on session cookies to authenticate received request. An attacker could exploit this vulnerability by getting a different user's browser to load a URL with a command from a vulnerable site on which the user is already logged in. There are many ways for an attacker to do that, such as by hosting a different web site that loads a resource from the vulnerable server, or getting the user to click a link. The attack can be prevented if the server sends an additional token to the client, requires the client to include that token in all future requests, and verifies that all future requests include a token that pertains to the current session, such as by using the ASP.NET AntiForgeryToken or ViewState. |

| TITLE | DETAILS |
|--------------------------------|---|
| Component | Web API |
| SDL Phase | Build |
| Applicable Technologies | MVC5, MVC6 |
| Attributes | N/A |
| References | Preventing Cross-Site Request Forgery (CSRF) Attacks in ASP.NET Web API |

| TITLE | DETAILS |
|--------------|--|
| Steps | Anti-CSRF and AJAX: The form token can be a problem for AJAX requests, because an AJAX request might send JSON data, not HTML form data. One solution is to send the tokens in a custom HTTP header. The following code uses Razor syntax to generate the tokens, and then adds the tokens to an AJAX request. |

Example

```
<script>
    @functions{
        public string TokenHeaderValue()
        {
            string cookieToken, formToken;
            AntiForgery.GetTokens(null, out cookieToken, out formToken);
            return cookieToken + ":" + formToken;
        }
    }
    $.ajax("api/values", {
        type: "post",
        contentType: "application/json",
        data: { }, // JSON data goes here
        dataType: "json",
        headers: {
            'RequestVerificationToken': '@TokenHeaderValue()'
        }
    });
</script>
```

Example

When you process the request, extract the tokens from the request header. Then call the `AntiForgery.Validate` method to validate the tokens. The `Validate` method throws an exception if the tokens are not valid.

```
void ValidateRequestHeader(HttpRequestMessage request)
{
    string cookieToken = "";
    string formToken = "";

    IEnumerable<string> tokenHeaders;
    if (request.Headers.TryGetValues("RequestVerificationToken", out tokenHeaders))
    {
        string[] tokens = tokenHeaders.First().Split(':');
        if (tokens.Length == 2)
        {
            cookieToken = tokens[0].Trim();
            formToken = tokens[1].Trim();
        }
    }
    AntiForgery.Validate(cookieToken, formToken);
}
```

Example

Anti-CSRF and ASP.NET MVC forms - Use the `AntiForgeryToken` helper method on Views; put an `Html.AntiForgeryToken()` into the form, for example,

```
@using (Html.BeginForm("UserProfile", "SubmitUpdate")) {
    @Html.ValidationSummary(true)
    @Html.AntiForgeryToken()
    <fieldset>
}
}
```

Example

The example above will output something like the following:

```
<form action="/UserProfile/SubmitUpdate" method="post">
    <input name="__RequestVerificationToken" type="hidden"
value="saTFWpkKN08YazFtN6c4YbZamsEwG0srqlUqqloi/fVgeV2ciIFVmelvzWRZpArs" />
    <!-- rest of form goes here -->
</form>
```

Example

At the same time, `Html.AntiForgeryToken()` gives the visitor a cookie called `__RequestVerificationToken`, with the same value as the random hidden value shown above. Next, to validate an incoming form post, add the `[ValidateAntiForgeryToken]` filter to the target action method. For example:

```
[ValidateAntiForgeryToken]
public ViewResult SubmitUpdate()
{
    // ... etc.
}
```

Authorization filter that checks that:

- The incoming request has a cookie called `__RequestVerificationToken`
- The incoming request has a `Request.Form` entry called `__RequestVerificationToken`
- These cookie and `Request.Form` values match Assuming all is well, the request goes through as normal. But if not, then an authorization failure with message "A required anti-forgery token was not supplied or was invalid".

TITLE	DETAILS
Component	Web API
SDL Phase	Build
Applicable Technologies	MVC5, MVC6
Attributes	Identity Provider - ADFS, Identity Provider - Azure AD
References	Secure a Web API with Individual Accounts and Local Login in ASP.NET Web API 2.2

TITLE	DETAILS
Steps	If the Web API is secured using OAuth 2.0, then it expects a bearer token in Authorization request header and grants access to the request only if the token is valid. Unlike cookie based authentication, browsers do not attach the bearer tokens to requests. The requesting client needs to explicitly attach the bearer token in the request header. Therefore, for ASP.NET Web APIs protected using OAuth 2.0, bearer tokens are considered as a defense against CSRF attacks. Please note that if the MVC portion of the application uses forms authentication (i.e., uses cookies), anti-forgery tokens have to be used by the MVC web app.

Example

The Web API has to be informed to rely ONLY on bearer tokens and not on cookies. It can be done by the following configuration in `WebApiConfig.Register` method: ``C-Sharp code `config.SuppressDefaultHostAuthentication();`
`config.Filters.Add(new HostAuthenticationFilter(OAuthDefaults.AuthenticationType));`

The `SuppressDefaultHostAuthentication` method tells Web API to ignore any authentication that happens before the request reaches the Web API pipeline, either by IIS or by OWIN middleware. That way, we can restrict Web API to authenticate only using bearer tokens.

Given below are screenshots of a previous version of the system/methods which were implemented in software that was disclosed at least as early as 12/31/2011. The 2011 version disclosed the following elements associated with the claims as originally presented and, to speed examination, counsel for applicant presents here a listing of claims as originally presented in the present application, with the items that are known to have been disclosed in the 2011 disclosed version shown in underline and bold and in black font, with the other elements being shown in red font and not underlined which were not included in the 2011 disclosed version.

1. A threat model chaining method, comprising:
providing one or more databases, the one or more databases comprising:
a plurality of threat model components stored therein;
a plurality of threats stored therein, each threat associated with at least one of the threat model components through the one or more databases; and
a plurality of compensating controls stored therein, each compensating control associated with at least one of the threats through the one or more databases;
providing one or more interfaces, including a diagram interface, configured to be displayed on one or more end user computing devices communicatively coupled with the one or more databases;
configuring the diagram interface to display a relational diagram of one of a system, an application, and a process, using visual representations of one or more of the threat model components and visual representations of one or more of the compensating controls, the relational diagram defining a first threat model, and;
configuring the diagram interface to, in response to receiving one or more user inputs, add a component group to the first threat model and thereby redefine the first threat model by including in it a second threat model associated with the component group, wherein the component group comprises a predefined interrelated group of two or more of the threat model components.

2. The threat model chaining method of claim 1, further comprising configuring the diagram interface to, in response to receiving a user selection of one of the diagrammed threat model components of the first threat model, visually display attack paths of all threats associated with the diagrammed threat model components which compromise the selected threat model component.

3. **The threat model chaining method of claim 1, further comprising configuring the one or more interfaces to, in response to receiving one or more user inputs, store one or more additional threat model components in the one or more databases, store one or more additional threats in the one or more databases, associate each additional threat with at least one of the additional threat model components through the one or more databases,** store one or more additional compensating controls in the one or more databases, and associate each additional compensating control with at least one of the additional threats through the one or more databases.

4. An attack simulation method, comprising:
- providing one or more databases, the one or more databases comprising:**
- a plurality of threat model components stored therein;**
- a plurality of threats stored therein, each threat associated with at least one of the threat model components through the one or more databases; and**
- a plurality of compensating controls stored therein, each compensating control associated with at least one of the threats through the one or more databases;
- providing one or more interfaces, including a diagram interface, configured to be displayed on one or more end user computing devices communicatively coupled with the one or more databases;**
- configuring the diagram interface to display a relational diagram of one of a system, an application, and a process, using visual representations of one or more of the threat model components** and visual representations of one or more of the compensating controls, **the relational diagram defining a first threat model; and**
- configuring the diagram interface to, in response to receiving a user selection of one of the diagrammed threat model components of the first threat model through the one or more interfaces, visually display attack paths of all threats associated with the diagrammed threat model components which compromise the selected threat model component.
5. The attack simulation method of claim 4, further comprising configuring the diagram interface to, in response to receiving one or more user inputs through the one or more interfaces, visually display one or more attack paths which pass through multiple diagrammed threat model components of the first threat model.
6. The attack simulation method of claim 4, further comprising configuring the diagram interface to, in response to receiving a user toggle of one of the diagrammed compensating controls (hereinafter "first control") of the first threat model to an on state through the one or more interfaces, toggle all attack paths mitigatable by the first control to a mitigated display state.

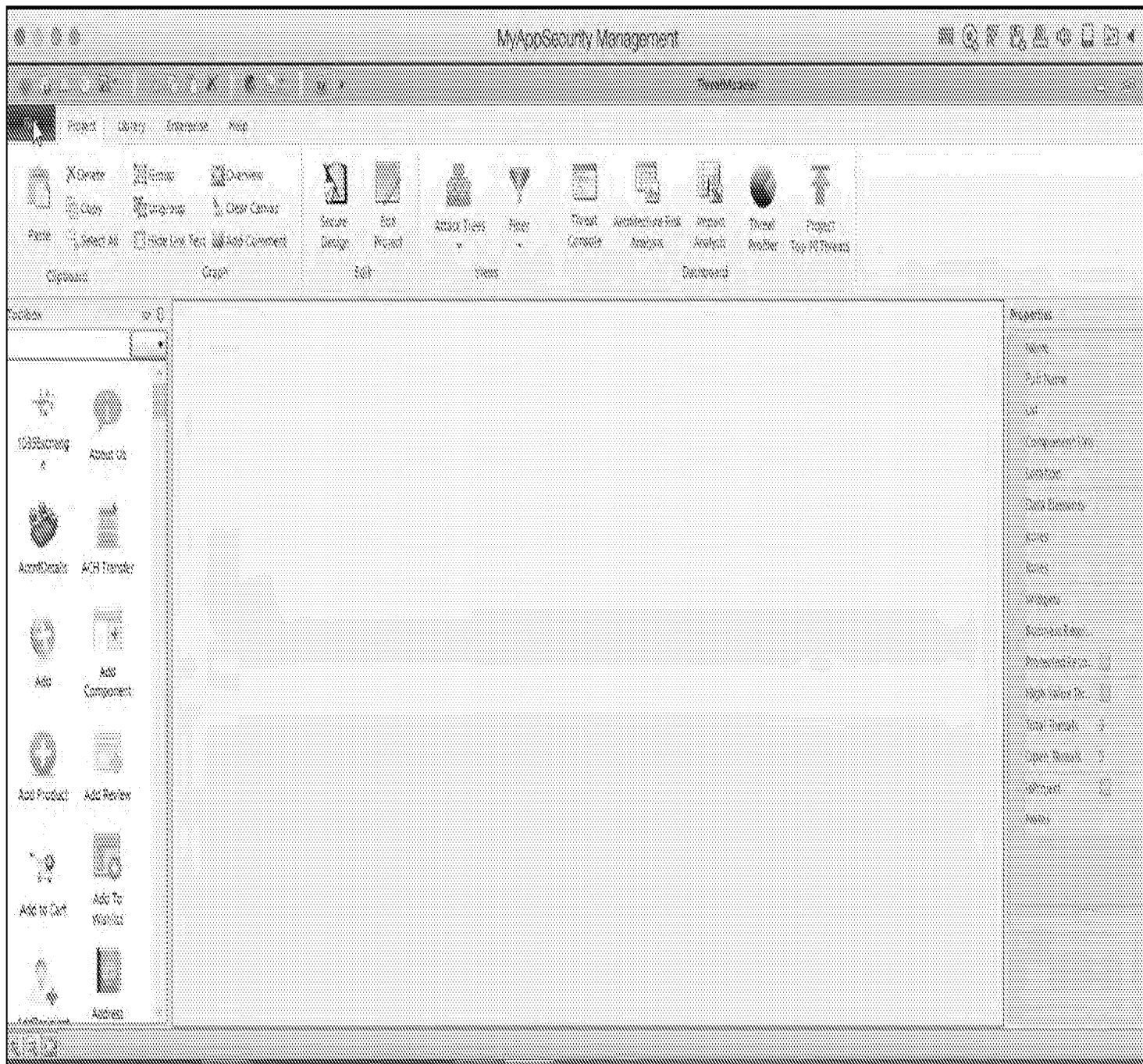
7. The attack simulation method of claim 6, further comprising configuring the diagram interface to, in response to receiving a user toggle of the first control to an off state through the one or more interfaces, toggle all attack paths mitigatable by the first control to an unmitigated display state.
8. The attack simulation method of claim 6, further comprising configuring the diagram interface to, in response to receiving one or more user inputs through the one or more interfaces to remove the first control from the first threat model, toggle all attack paths mitigatable by the first control to an unmitigated display state.
9. The attack simulation method of claim 4, further comprising configuring the diagram interface to, in response to receiving one or more user inputs through the one or more interfaces, visually display only attack paths of all threats associated with a subset of the diagrammed threat model components.
10. The attack simulation method of claim 4, further comprising configuring the diagram interface to, in response to receiving one or more user inputs through the one or more interfaces to alter a profile of an attacking component, alter the visually displayed attack paths.
11. The attack simulation method of claim 3, further comprising configuring the diagram interface to receive one or more user inputs to add a component group to the first threat model and thereby redefine the first threat model by including in it a second threat model associated with the component group, wherein the component group comprises a predefined interrelated group of two or more of the threat model components.

12. An attack simulation system, comprising:
one or more computing devices communicatively coupled with one or more databases, the one or more computing devices displaying, on one or more displays of the one or more computing devices:
one or more input interfaces configured to, in response to receiving one or more user inputs, store a plurality of user-defined threat model components in the one or more databases, store a plurality of threats in the one or more databases, associate each of the threats with at least one of the threat model components through the one or more databases, store a plurality of compensating controls in the one or more databases, and associate each compensating control with at least one of the threats through the one or more databases, and;
a diagram interface configured to, in response to receiving one or more user inputs, diagram one of a system, an application, and a process, the diagram including one or more of the threat model components and one or more of the compensating controls, to define a first threat model, the first threat model including all threats associated through the one or more databases with the diagrammed threat model components;
wherein the diagram interface is configured to, in response to receiving a selection of one of the diagrammed threat model components of the first threat model, visually display attack paths of all threats associated with the diagrammed threat model components which compromise the selected threat model component.
13. The attack simulation system of claim 12, wherein the diagram interface is configured to, in response to receiving a user input to toggle one of the diagrammed compensating controls (hereinafter "first control") to an on state, toggle all attack paths mitigatable by the first control to a mitigated display state.

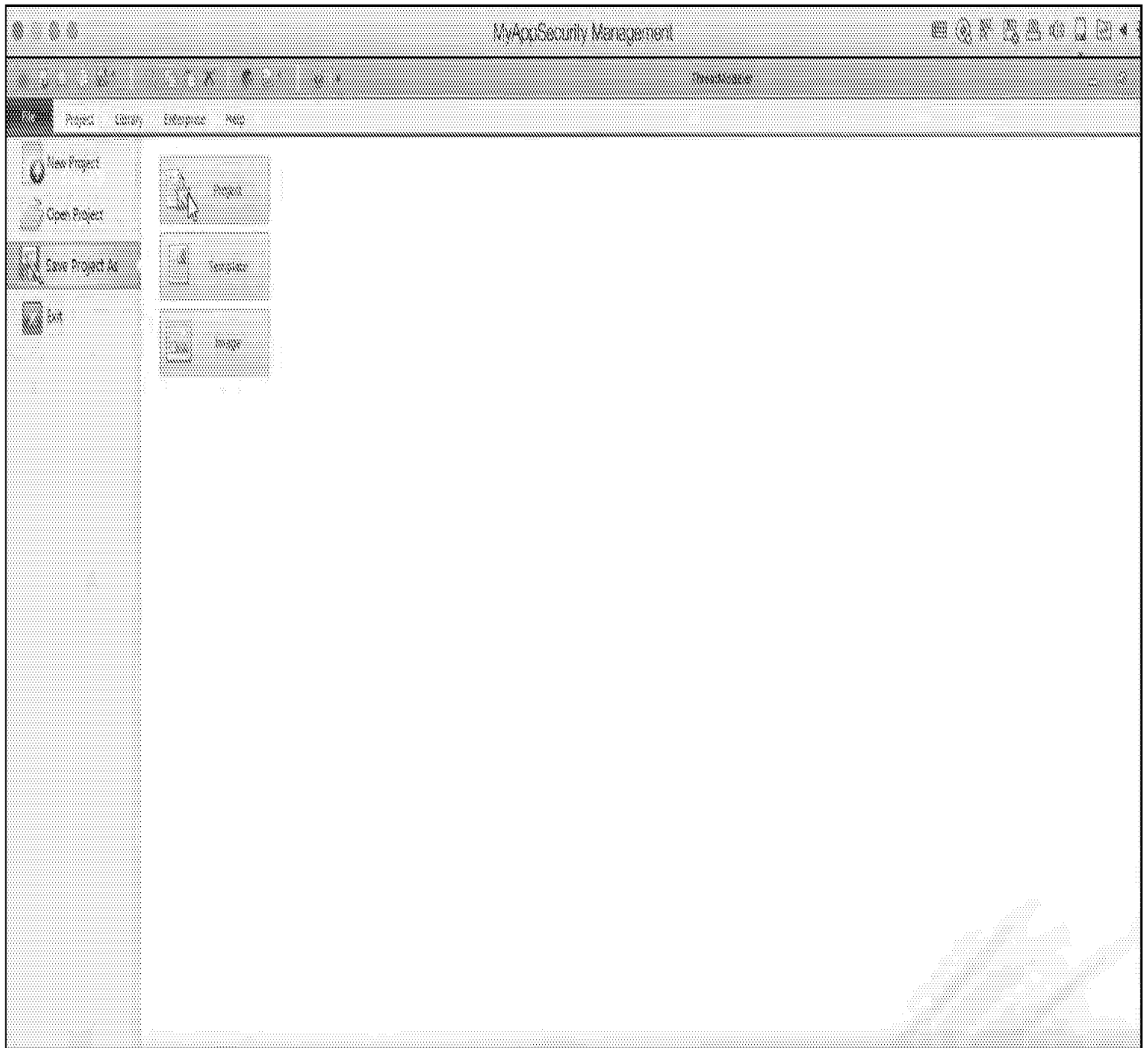
14. The attack simulation system of claim 13, wherein the diagram interface is configured to, in response to receiving a user input to toggle the first control to an off state, toggle all attack paths mitigatable by the first control to an unmitigated display state.
15. The attack simulation system of claim 13, wherein the diagram interface is configured to, in response to receiving one or more user inputs removing the first control from the first threat model, toggle all attack paths mitigatable by the first control to an unmitigated display state.
16. The attack simulation system of claim 12, wherein the diagram interface is configured to, in response to receiving one or more user inputs, visually display only attack paths of all threats associated with a subset of the diagrammed threat model components.
17. The attack simulation system of claim 12, wherein the diagram interface is configured to, in response to receiving one or more user inputs, alter a profile of an attacking component, wherein the alteration alters the visually displayed attack paths.
18. The attack simulation system of claim 12, **wherein the one of a system, an application, and a process comprises a computing network.**
19. The attack simulation system of claim 18, **wherein the diagram interface is further configured to, in response to receiving one or more user inputs, diagram a communication protocol between two of the diagrammed threat model components, the diagrammed communication protocol including an alphanumeric indicator.**
20. The attack simulation system of claim 12, wherein the diagram interface is configured to, in response to receiving one or more user inputs, add a component group to the first threat model and thereby redefine the first threat model by including in it a second threat model

associated with the component group, wherein the component group comprises a predefined interrelated group of two or more of the threat model components.

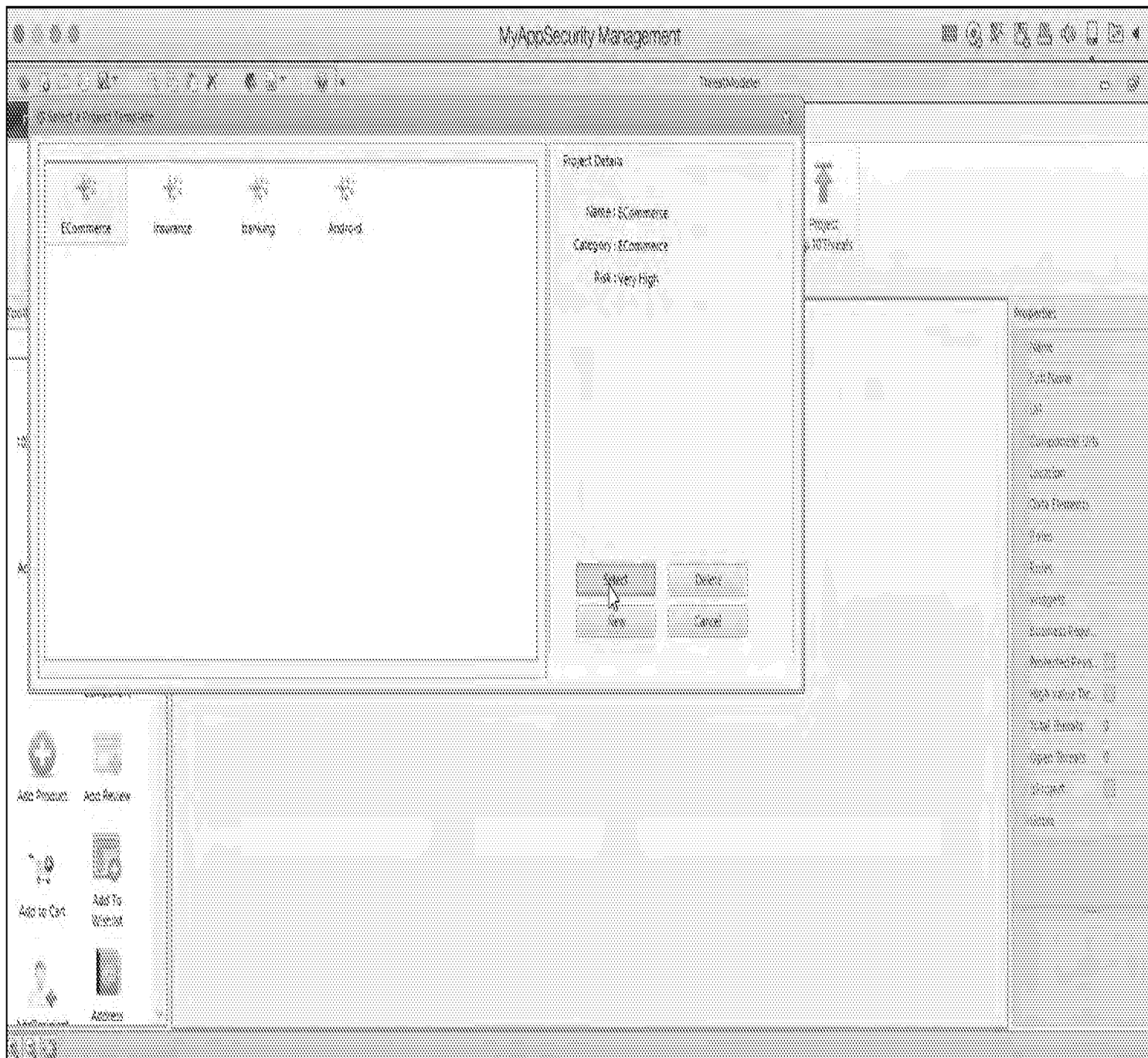
Screenshots and explanations of some of the user interfaces of the 2011 version are shown below:



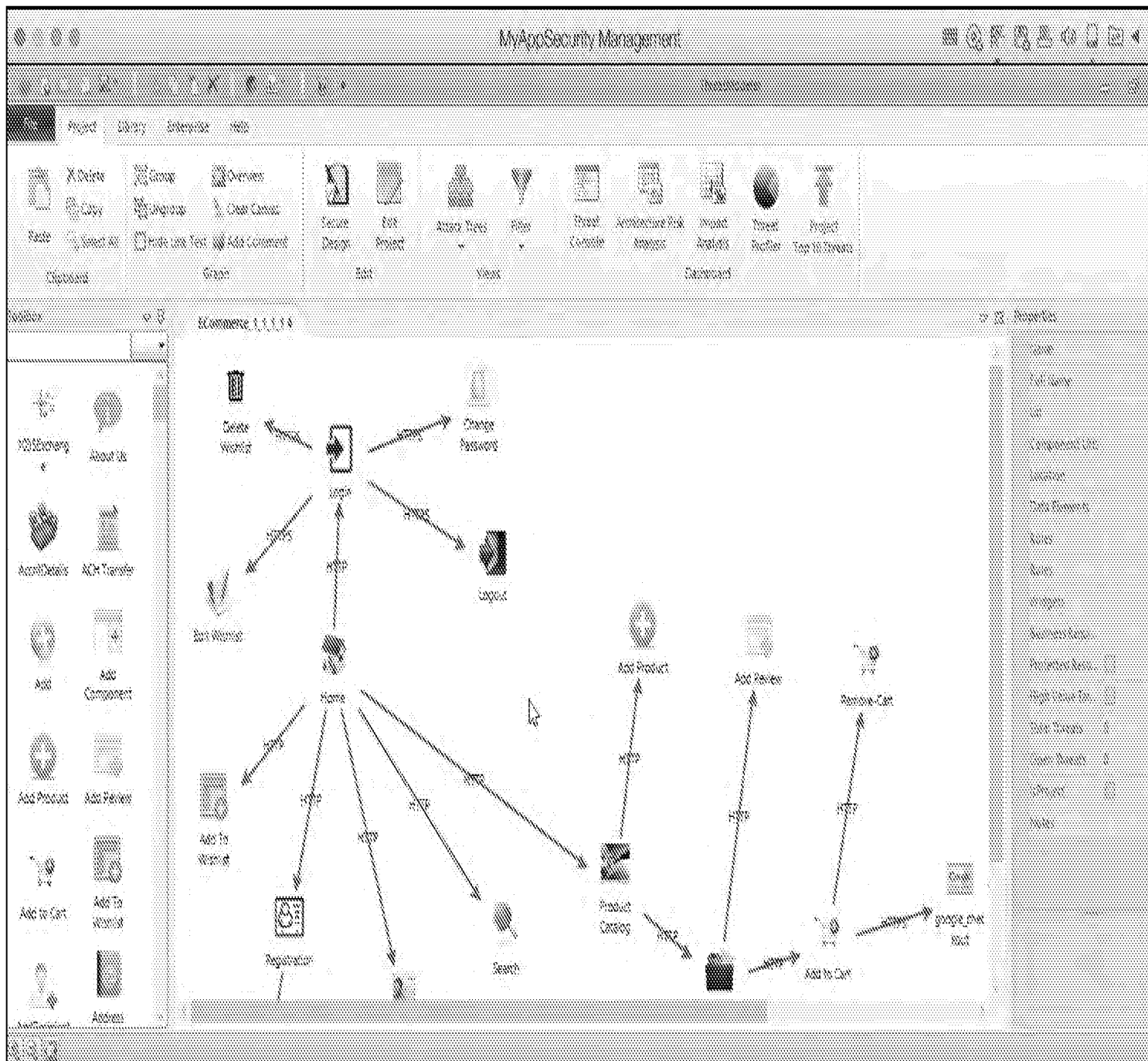
This is a home screen. The user is selecting the “File” menu item at the top left with the cursor.



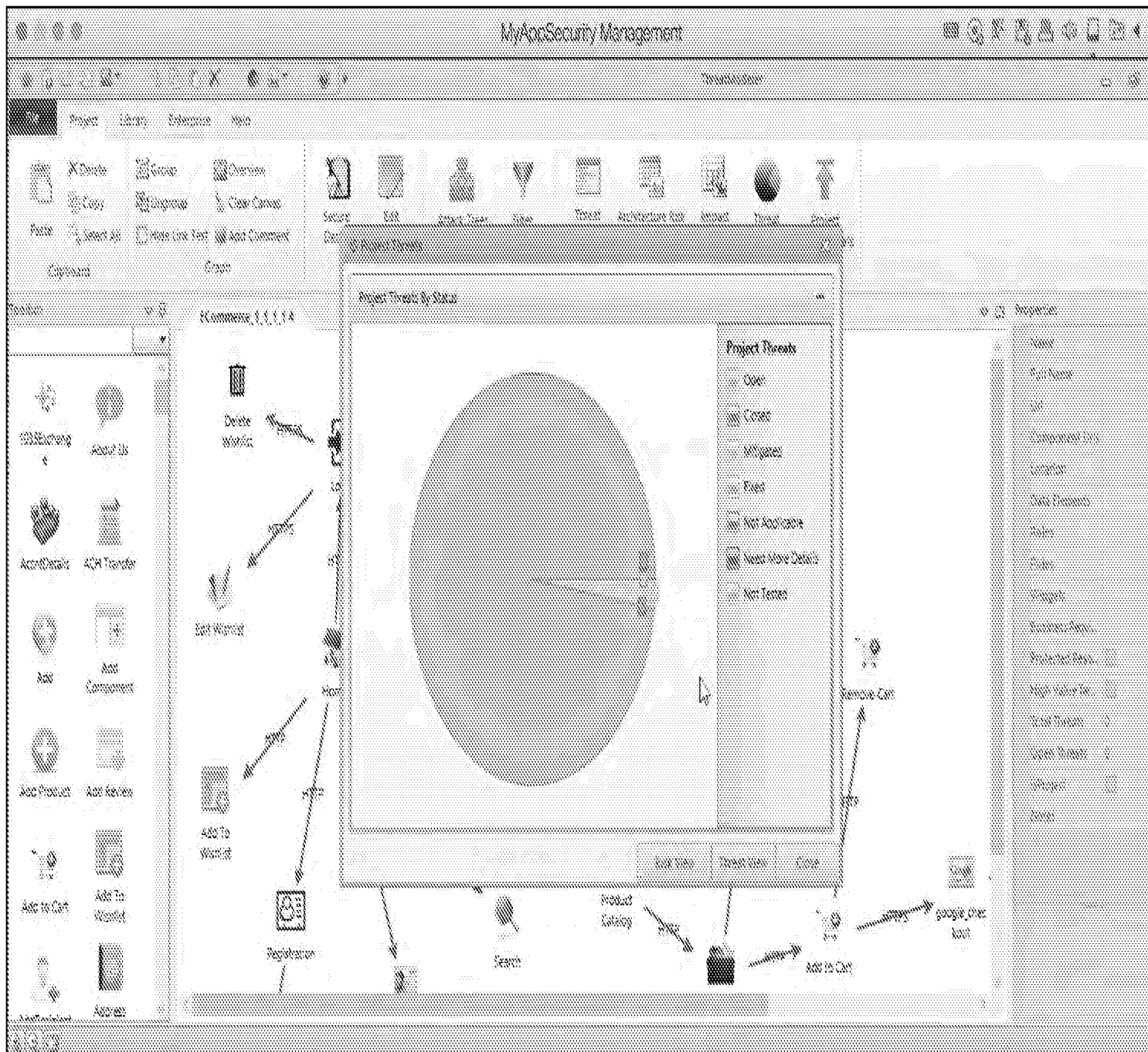
Once the file menu was selected the user was now able to open or create or save a new project, template, or image.



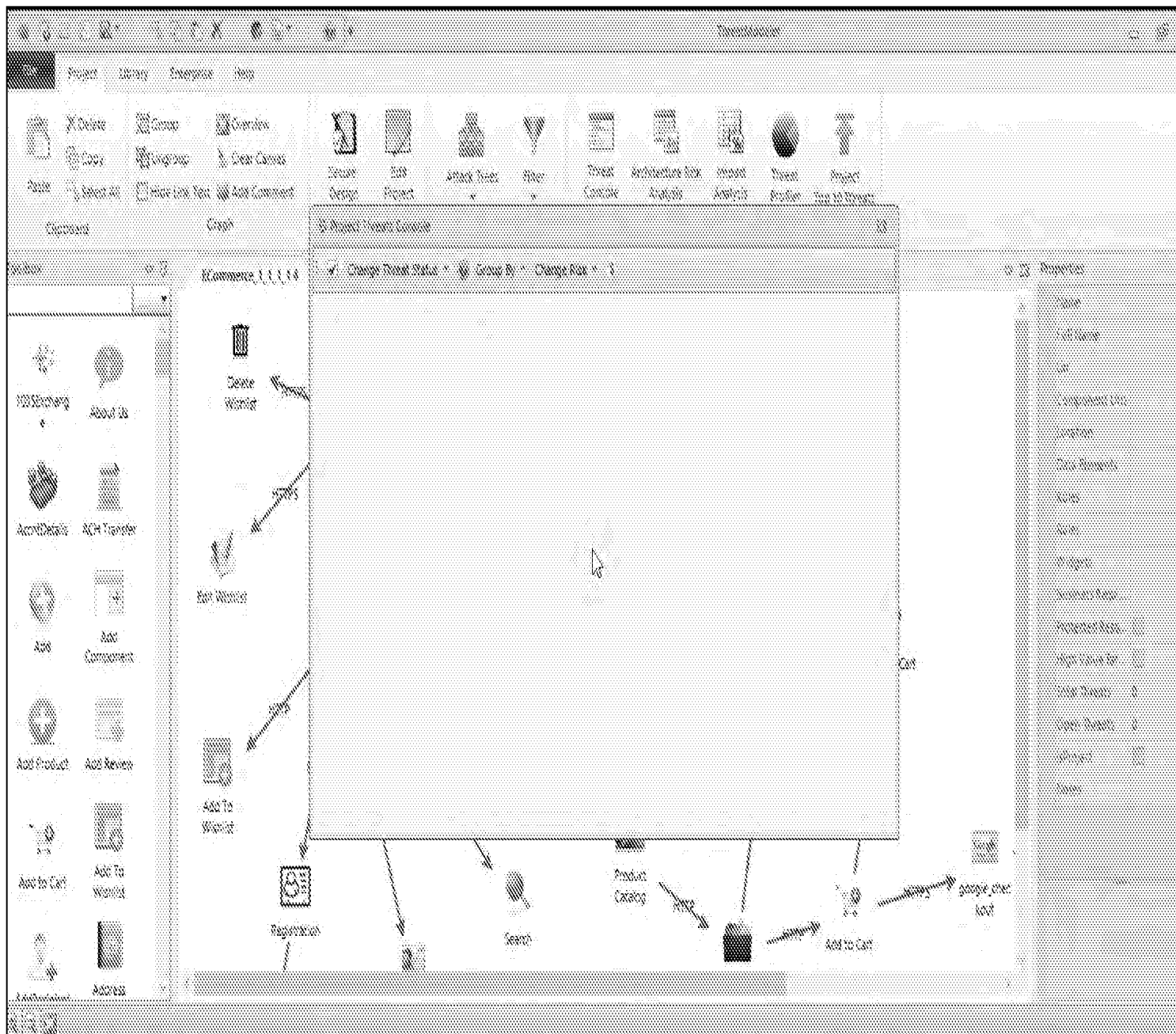
The user is here opening an existing threat model (project) titled Ecommerce, from this screen the user can also delete this threat model, or initiate the creation of a new threat model, or cancel to exit the screen.



Here the user has opened the existing ECommerce threat model, showing components linked to one another using communication protocols. Noticeably absent from this diagram are any compensating controls. This 2011 version did not have compensating controls. It showed the threats in a threat model (discussed later), but had no ability to remove or mitigate threats automatically through compensating controls.

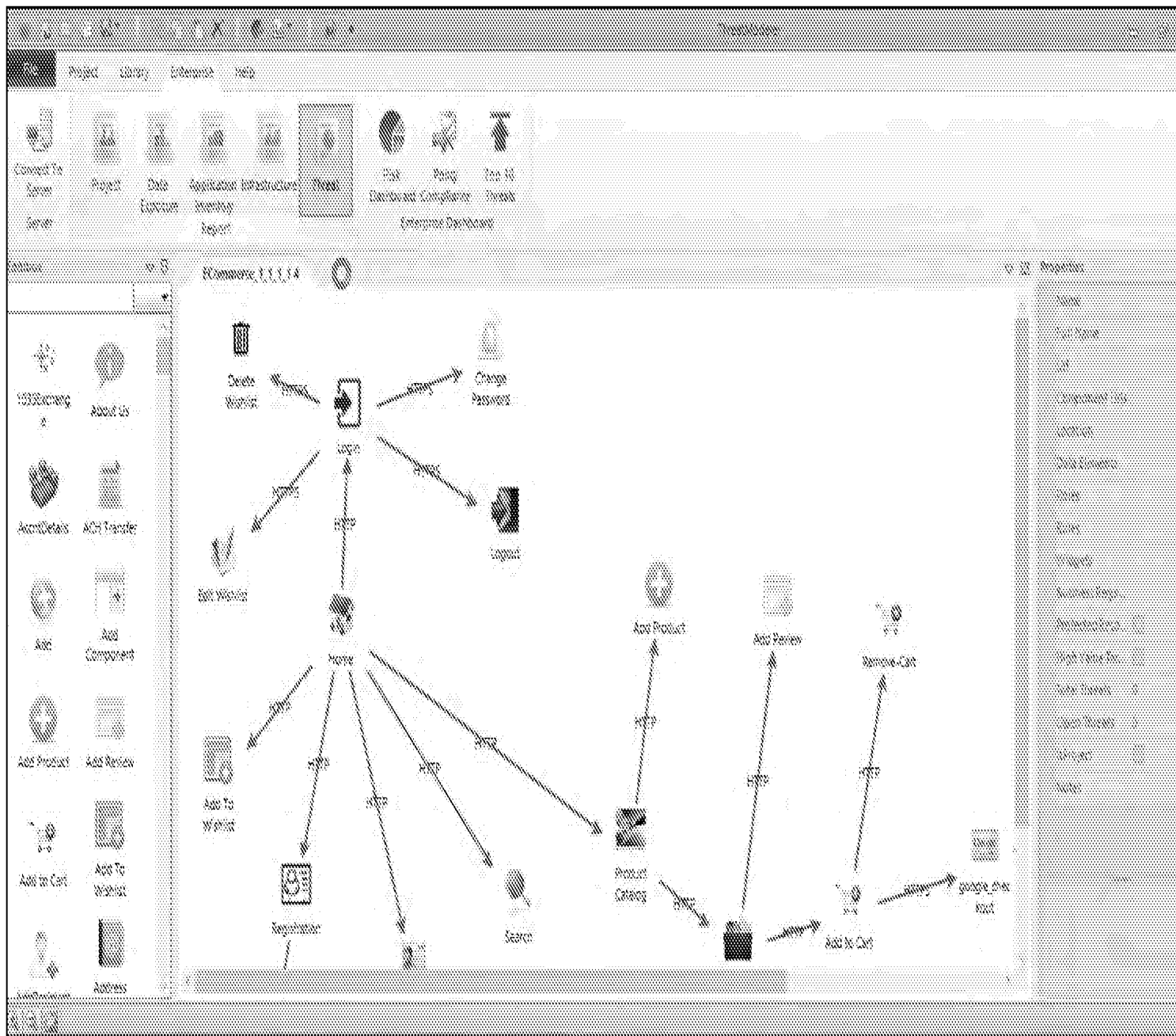


This is an interface showing a pie chart illustrating the percentages of threats by status including open, closed, mitigated, fixed, not applicable, need more details, and not tested.



This interface shows the threat report for the diagrammed threat model, though the threats are now not displaying properly. But the user has the ability to change the threat status manually (between the aforementioned statuses shown in the previous screen), change the risk level of any selected threat, and group the threats in different ways to organize the view. When the threats displayed properly they were displayed inside the popup window similar to what is shown below:

THREAT ▼	RISK ▼	STATUS ▼	SOURCE ▼
ACCT HIJACK	VERY HIGH	OPEN	AWS ELB
INSEC. COMM.	VERY HIGH	OPEN	AWS ELB
PWD BRUTE F...	HIGH	OPEN	APP
SENS DATA EXP	VERY HIGH	OPEN	APP
SQL INJECTION	HIGH	MITIGATED	APP
BLIND SQL INJ	HIGH	MITIGATED	APP
IDENT SPOOF	MEDIUM	OPEN	WEB BR
TARGETED MAL	VERY HIGH	OPEN	WEB BR
CLICKJACKING	VERY HIGH	OPEN	APP
X SITE TRACING	VERY HIGH	OPEN	WEB SVR
IIS SNIFFING	VERY HIGH	OPEN	WCF SVC
XML SVC DEN	VERY HIGH	OPEN	WEB SVC
BUFFER OVFLW	VERY HIGH	OPEN	MS WORD
FILE MANIP	MEDIUM	MITIGATED	FILE SYS
PHYS THEFT	VERY HIGH	MITIGATED	LAPTOP
BLUEJACKING	VERY HIGH	OPEN	BT PORT



From this screen it can be seen that the user has selected the Enterprise tab at the top and, when selecting a Threat selector, can generate a MS Word version of the threat report. This first several pages of this report are shown in the screens below, and the bottommost image shows the user saving a copy of the report as an MS Word file.



MyAppSecurity Management

Threats found in the Projects:

Threat : Cross Site Request Forgery (aka Session Riding)

Risk : Very High

Projects :

ECommerce_1,ECommerce_1_1,ECommerce_1_1_1,aiugius,ECommerce_1_1_1_1,HackMe Bank,POC Application,Android Banking,Strife Insurance,User Registration Auth,Cool Car Company,Demo - Trust Boundary

Description :

This type of attack leverages the persistence and implicit trust placed in user session cookies by many web applications today. In such an architecture, once the user authenticates to an application and a session cookie is created on the user's system, all following transactions for that session are authenticated using that cookie including potential actions initiated by an attacker and simply "riding" the existing session cookie.

Threat : SQL Injection - WASC

Risk : Very High

Projects :

ECommerce_1,ECommerce_1_1,ECommerce_1_1_1,aiugius,ECommerce_1_1_1_1,HackMe Bank,POC Application,Android Banking,Strife Insurance,User Registration Auth,Cool Car Company,Demo - Trust Boundary

Properties

Name

Full Name

ID

Component ID

Description

Data Elements

Notes

Project

Severity

Associated Assets

Associated Risks

High Value For

Total Risks

Open Risks

Controls

Labels

Registration

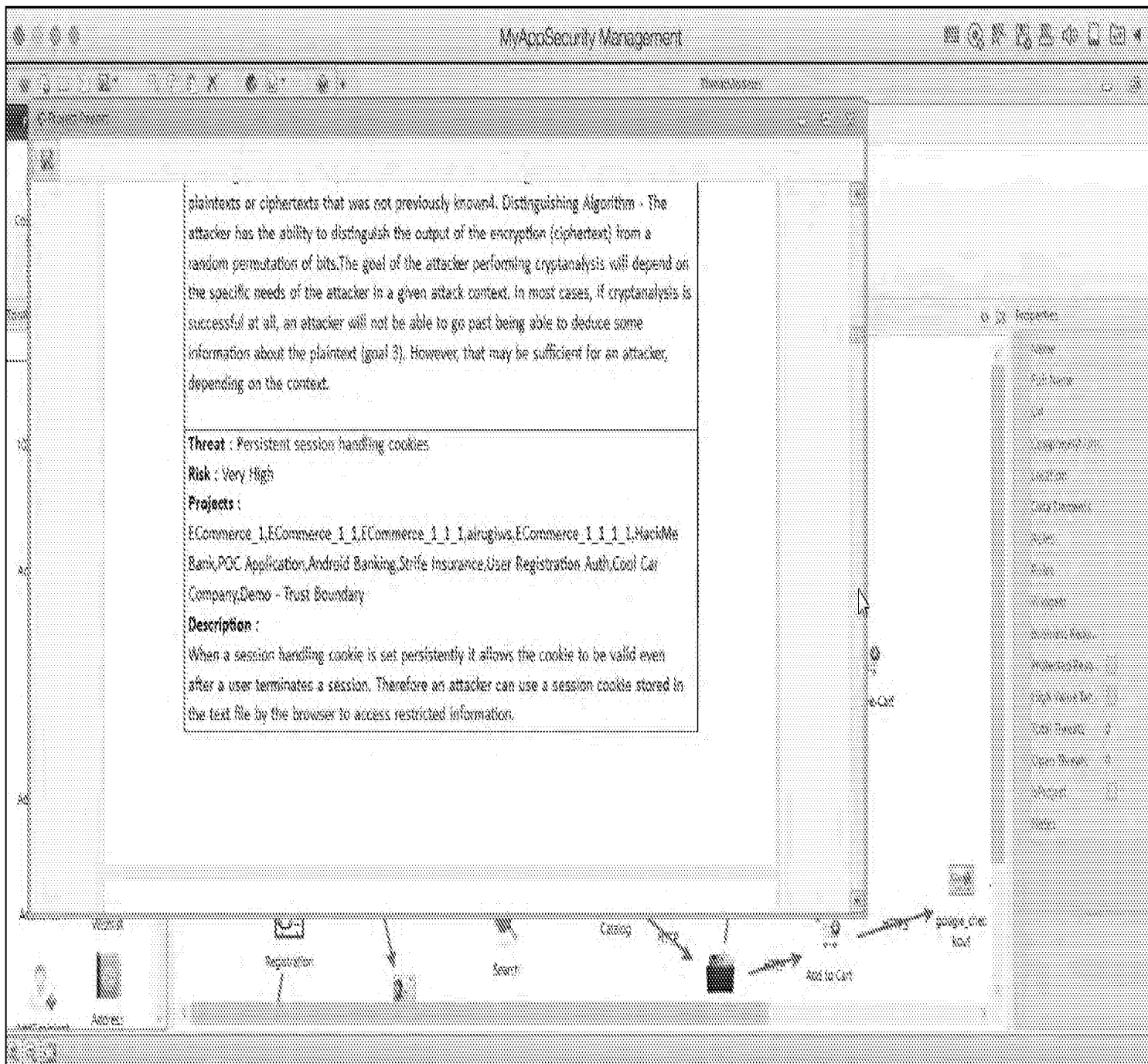
Search

Catalog

Add to Cart

google_checkout

IriusRisk
Ex. 1002 - Page 236



MyAppSecurity Management

Threats

Threat : Overflow Buffers

Risk : High

Projects :

ECommerce_1,ECommerce_1_1,ECommerce_1_1_1,airugios,ECommerce_1_1_1_1,HackMe Bank,POC Application,Android Banking,Strife Insurance,User Registration Auth,Cool Car Company,Demo - Trust Boundary

Description :

Buffer Overflow attacks target improper or missing bounds checking on buffer operations, typically triggered by input injected by an attacker. As a consequence, an attacker is able to write past the boundaries of allocated buffer regions in memory, causing a program crash or potentially redirection of execution as per the attacker's choice.

Threat : Reusing Session IDs (aka Session Replay)

Risk : High

Projects :

ECommerce_1,ECommerce_1_1,ECommerce_1_1_1,airugios,ECommerce_1_1_1_1,HackMe Bank,POC Application,Android Banking,Strife Insurance,User Registration Auth,Cool Car Company,Demo - Trust Boundary

Description :

This attack targets the reuse of valid session ID to spoof the target system in order to gain privileges. The attacker tries to reuse a stolen session ID used previously during a transaction to perform spoofing and session hijacking. Another name for this type of attack is Session Replay

Properties

Name

Author

Unit

Component ID

Location

Category

Rating

Score

Weight

Assigned Paper

Assigned Person

High Value For

How Threats

Open Status

Project

Notes

World

Address

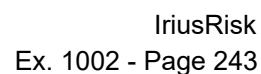
Registration

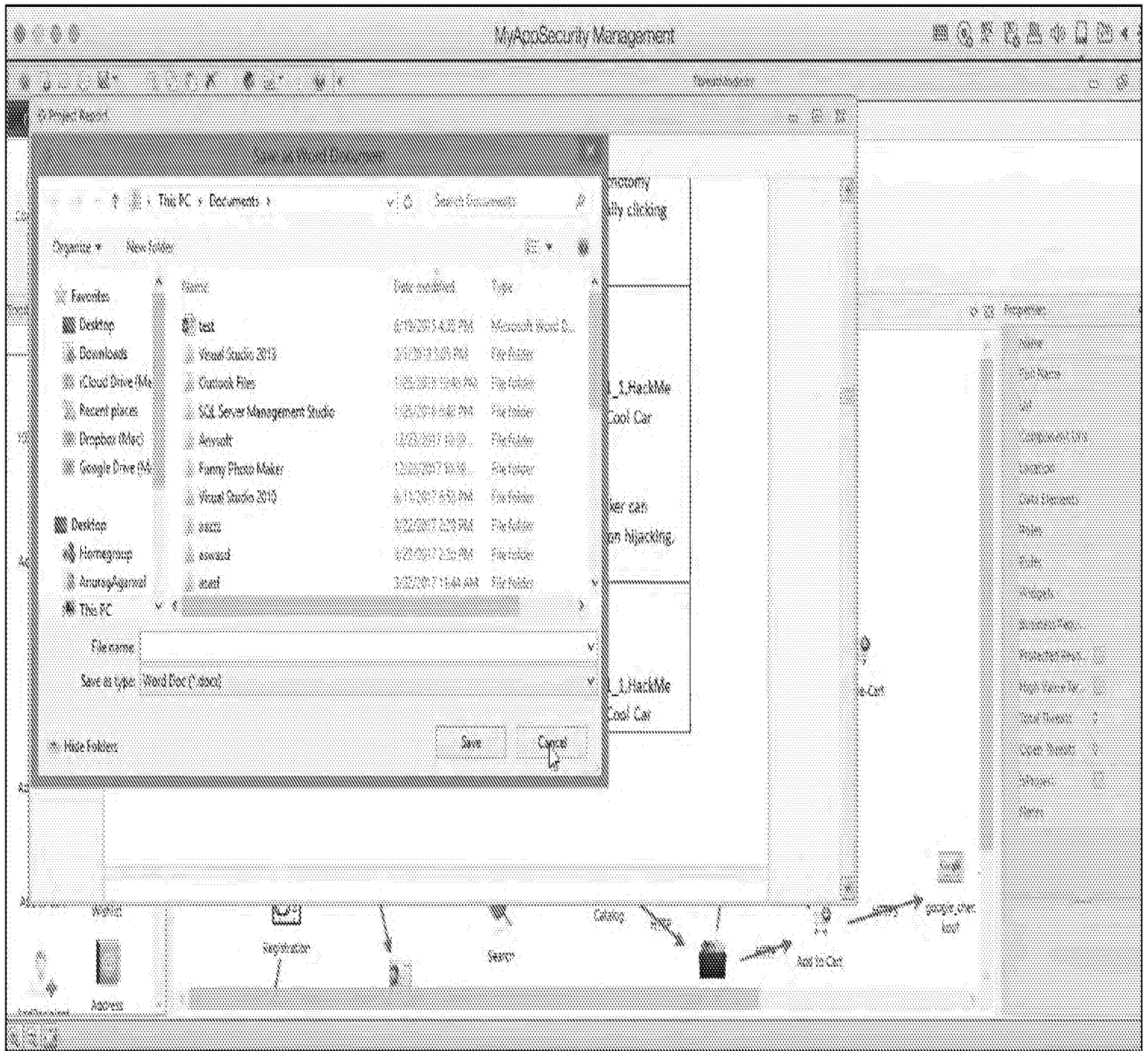
Search

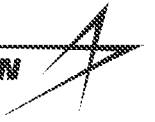
Catalog

Add to Cart

google, the kool







Seven Ways to Apply the Cyber Kill Chain[®] with a Threat Intelligence Platform

A White Paper Presented by:
Lockheed Martin Corporation

ABSTRACT

Threat Intelligence Platforms (TIP) are an emerging technology supporting organizations as they consume and then act on cyber intelligence. Lockheed Martin believes that a TIP helps an organization transition from relying solely on external intelligence sources to producing their own intelligence based on what is observed in their environment. The result is elevated cyber maturity and improved resilience against attackers.

1. INTRODUCTION

Within the past decade, computer network defense has shifted from a culture of sharing minimal information to one of intelligence overload. Previously, information regarding system breaches, malware, or attack attribution was rarely shared between organizations. Today the more common issue is how to sift through all the emails, reports, and indicators to identify actionable intelligence.

Threat intelligence is evidence-based knowledge about a threat that can be used to inform decisions regarding the response to that threat (McMillan, 2013). It includes the details of the motivations, intent, and capabilities of threat actors (Holland, 2014). In order to successfully defend against the multitude of Advanced Persistent Threats (APT) facing an organization, consuming external threat intelligence has become an increasingly important aspect of cybersecurity. However, exactly how to ingest the intelligence and successfully leverage it within an organization's environment often remains a challenge.

In addition to external intelligence, an influx of data from one's own organization can further complicate matters. Alerts from a myriad of technologies including Intrusion Detection Systems (IDS), firewalls, mail scanners, Host Intrusion Prevention Systems (HIPS), and proxies can overwhelm a defender who is trying to respond to and disposition each alert. External intelligence can quickly become an afterthought as there is no

time to evaluate and implement countermeasures, making it useless. Conversely, external intelligence fed directly into these tools results only in more noise if it is not properly vetted for one's environment and mitigations are not appropriately tuned.

In order to process all of this internal and external data and have it result in actionable intelligence, a TIP can be employed. A TIP is the central management repository for all external and internal intelligence, and can provide the mechanism to act upon this intelligence.

This paper is organized as follows: section two of this paper documents related work on defining the requisite components of a TIP. Section three introduces an expansion of this definition that is based upon the Intelligence Driven Defense® approach to computer network defense. Section four outlines the seven ways an organization can apply the Cyber Kill Chain® framework in their environment using a TIP. Section five introduces the Palisade™ solution, Lockheed Martin's Threat Intelligence Platform, and section six summarizes the paper.

2. RELATED WORK

The analyst firms Gartner, Inc. and Forrester Research provide insight on the value of employing a Threat Intelligence Platform, and have each offered definitions of its components.

Gartner states that organizations looking for effective, automated methods to ingest multiple sources and formats of threat intelligence, correlate them, and act upon the information more efficiently should investigate using a TIP. They suggest that a TIP has the following six capabilities: 1) Collect intelligence from multiple sources in multiple formats and automatically enrich the intelligence; 2) Correlate intelligence and help determine which sources are the most effective; 3) Categorize intelligence to help perform analytics on threats, recognize tactics, techniques, and

procedures (TTPs), and understand relationships through modeling and visualizations; 4) Integrate the intelligence with workflow processes to further enrich existing data; 5) Provide action by supporting various integrations via Application Program Interfaces (APIs) and email notifications; and 6) Support the sharing of intelligence among trusted communities (Lawson, 2014).

Forrester states that the specific benefit of a TIP is providing Security Operation Centers with analysis platforms that can orchestrate intelligence activities. The platform should be capable of ingesting and normalizing threat intelligence, rating the value of intelligence sources, providing an analyst workspace, supporting visualization and pivoting, enriching the intelligence, and enabling the internal and external collaboration and sharing of the intelligence (Holland, 2015).

3. EXPANDING THE OBJECTIVE OF A THREAT INTELLIGENCE PLATFORM

Gartner and Forrester laid out the intention and minimal requirements for a TIP. Lockheed Martin expands on those requirements to ensure a TIP provides value in all environments. While a TIP should enable the collection of vast amounts of external intelligence, it should also provide the foundation for an organization to mature into a producer of intelligence.

An Intelligence Driven Defense® approach to computer network defense has been advocated by Lockheed Martin for over five years (Hutchins et al., 2010). The central theme is that you can learn from past intrusion attempts to generate actionable intelligence that can be put back into your defenses to make you more resilient against advanced adversaries. Connecting the dots between seemingly different data points allows a defender to recognize relationships among incidents and identify common characteristics. This allows the analyst to understand the tactics, techniques, and procedures (TTPs) of their adversary. The more complete the profile that is built on an adversary directly targeting your

network, the greater the opportunity to be one step ahead of them.

Without a dedicated tool to store all the information and context gleaned from an investigation, identifying those commonalities becomes impractical. In the absence of a TIP, an analyst has few options to capture the data that otherwise could provide keen insights on an attack. Spreadsheets or text files on a SharePoint site do not allow an analyst much flexibility to correlate events by leveraging historical knowledge. For example, if you identify malware that beacons out to an IP address, how can you determine whether you have previously seen that IP address in your environment? A TIP should provide the location for an analyst to input all the indicators, context, notes, intelligence, recommendations, evidence, and adversary profile information discovered during their investigation and ensure everything is easily searchable. Furthermore, a TIP should support the Cyber Kill Chain® framework for this analysis as it is central to an Intelligence Driven Defense® posture. This approach can help an analyst quickly determine the severity of an attack and then identify gaps in either the analysis or the organization's defenses.

External intelligence can be an important facet of computer network defense, and a TIP should support its consumption. However, dependence on those sources is reduced as an organization's cyber posture matures. External intelligence can provide insight into potential threats while an organization works to adopt an intelligence driven approach and develop their own analytic capabilities. As internal capabilities mature, external sources should be used to correlate what is seen in the environment and provide greater context and insight to the adversary's TTPs. The TIP should also provide a mechanism to measure those external sources, making it easy to determine which provide value. Those not beneficial to your organization should be removed – an action that will reduce noise and save money.

4. SEVEN WAYS TO APPLY THE CYBER KILL CHAIN® WITH A THREAT INTELLIGENCE PLATFORM

The Cyber Kill Chain® framework for computer network defense is not something that can be placed into an enterprise's defenses. However, there are seven ways to apply it within an organization to mitigate risk, build true resilience, better communicate, and meaningfully measure results (Hutchins, 2014). Each of these applications can be achieved or augmented with the use of a TIP that manages threat intelligence and enables an analyst to identify broader campaign activity by leveraging historical data.

4.1 PRIORITIZE SENSOR ALERTS

With so many alerts coming from so many sources, which alerts should be analyzed first? Which sensor is the most important and demands immediate attention? In order to answer these questions, an organization should list the events that are produced by each sensor and map them to the relevant stage of the Cyber Kill Chain®. The further down the attack chain to which an event maps, the higher the priority of that alert. For example, an event produced from a Network Intrusion Detection System (NIDS) maps to either the first stage, Reconnaissance, or the third stage, Delivery. However, an event from a HIPS maps to stage four or five, Installation or Exploitation, or perhaps even stage seven, Action on Objectives. Thus alerts coming from the HIPS should be evaluated first, because they correspond to the potentially more damaging event.

A proper Threat Intelligence Platform can help an organization with this by automatically ingesting alerts from a Security Information and Event Management (SIEM) or SIEM-like device that aggregates and normalizes events. The alerts should be ingested by the TIP and prioritized based on the original source of the alert. Those mapped to later stages of the Cyber Kill Chain® process should be associated with a higher priority and be displayed for an analyst to respond to first. The TIP

should also provide the functionality for the analyst to respond to the alert within the tool. That way if it is an intrusion or attempted intrusion, indicators can be documented, and notes recorded to explain how the investigation was conducted. Standard courses of action can also be documented so that similar alerts are dispositioned and addressed in a repeatable and efficient manner.

4.2 PRIORITIZE ESCALATION

Another common problem for analyst teams is knowing with whom to report each attempted intrusion into the network. Investigation of an incident through the Cyber Kill Chain® framework can quickly determine the severity of the attack. Was a malicious email delivered to a recipient's inbox? Is a computer on the network communicating with a known command and control IP address? Once again, the further down the kill chain the attack progresses, the higher the incident should be reported in an organization's management chain. If an attack results in Action on Objectives – meaning the adversary was able to exfiltrate data from the network – the impact could potentially require notification to the organization's CEO or Board of Directors. On the other hand, a “shielded” attack, where malware has been installed on a host computer, but there is no command and control, may only need to be reported up to a Chief Information Security Officer, as this sort of compromise has much less impact on the organization.

A TIP can provide the platform for an analyst to record all results from the investigation of an intrusion. When an alert comes in, an analyst gathers data points and documents them in the TIP. The analyst then studies the data to understand the alert and identify how far the intrusion progressed. The TIP can map key pieces of intelligence to the stage of the Cyber Kill Chain® stage from which it was pulled to determine how far the intrusion advanced and the level of impact to the organization. This can then align directly to the level of escalation required to communicate the impact.

4.3 PRIORITIZE INVESTMENT

Establishing a return on investment for security products can be an arduous task. Building a business case for future investments can be even more difficult. An organization can visually display where protections exist that detect, deny, disrupt, degrade, and/or deceive an intrusion attempt at each stage of the Cyber Kill Chain® process. A graph showing each stage of the process on one axis and the countermeasure along the other can then identify the tool in place to perform the mitigation at that stage in the box where each intersects. An empty box indicates gaps where investments can be made to further enhance the organization's protections. An example of this type of graph is shown in Figure 1.

A TIP can provide this type of graph as output by associating mitigations for every intrusion attempt to each stage of the attack. A case can then be made to invest in tools that would fill the gap areas in an organization's countermeasures. Not every box would need to be filled for every organization, but the further along the Cyber Kill Chain®, the greater the priority. For example, there may not be a need to degrade Reconnaissance activity, but there would certainly be a need to detect a command and control channel.

Additionally, TIPs could help an organization identify data gaps and source collection requirements. As analysts investigate intrusions and intrusion attempts, those stages lacking data can lead to investments in tools to close the gap.

4.4 MEASURE EFFECTIVENESS

As previously stated, attempted attacks should be investigated, analyzed, and synthesized using the Cyber Kill Chain® framework. The stage at which the attack was stopped should be documented. The goal is that over time attacks would be stopped earlier and earlier in the process.

Once again, a TIP can be the platform for an analyst to record the results of the investigation, analysis, and synthesis of each attempted intrusion. The identification of where in the Cyber Kill Chain® process the intrusion was stopped can also be recorded as a data point within the TIP. Another output of the TIP could then be a graphical representation of these stopping points over a period of time. Ideally the TIP would show a trend of analysts recognizing and preventing adversaries from reaching later stages of the kill chain, and thus becoming more effective. An example of this graph is shown in Figure 2.

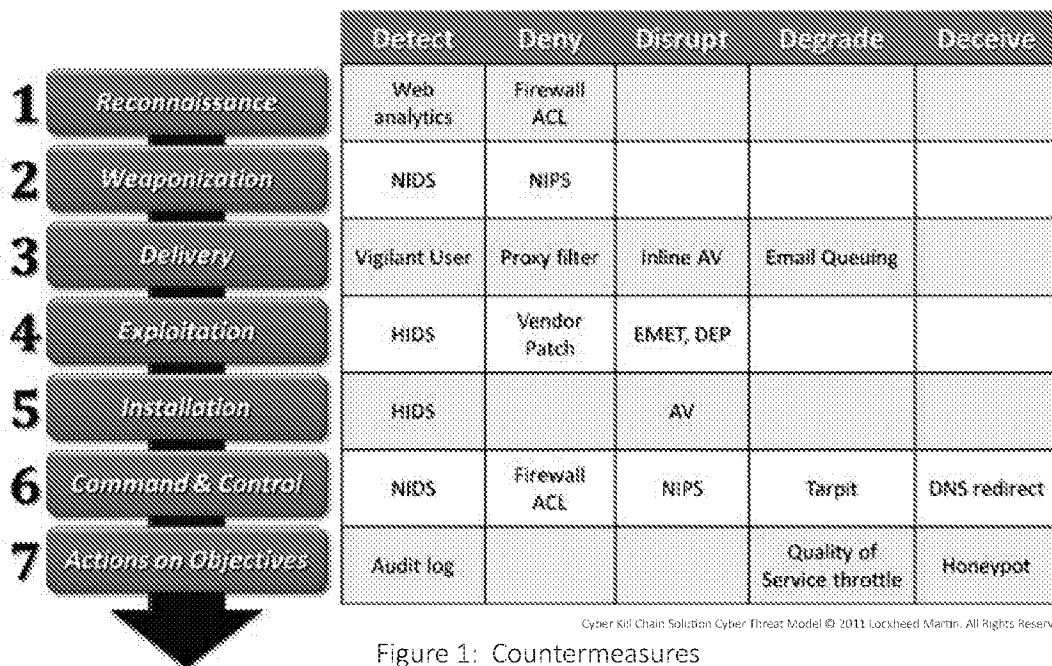


Figure 1: Countermeasures

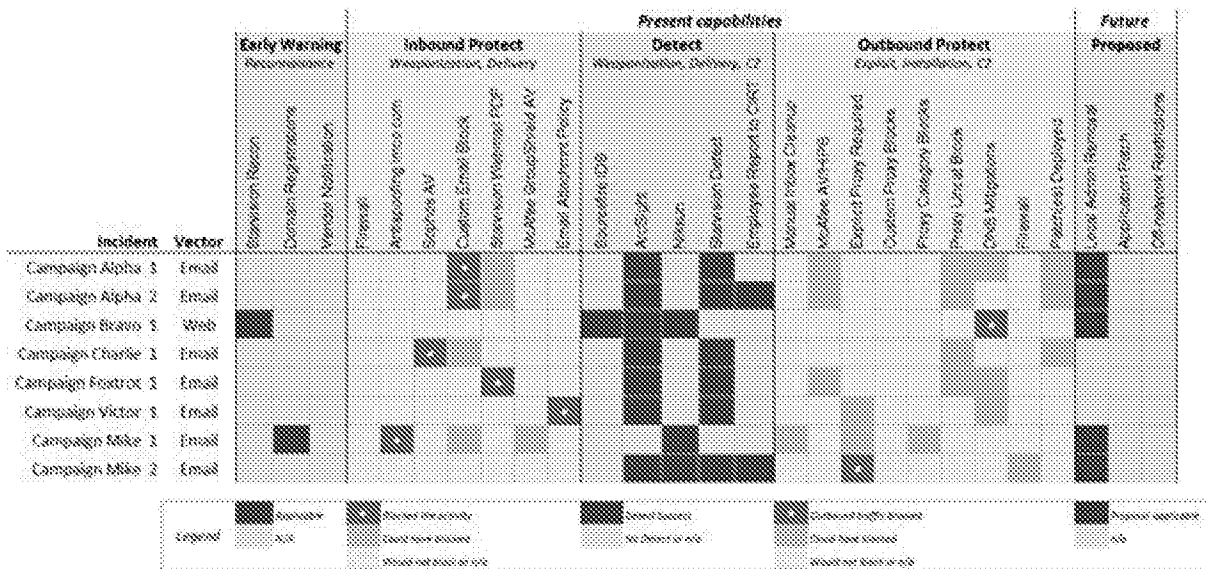


Figure 2: Measuring Effectiveness Scorecard

4.5 MEASURE RESILIENCE

If an intrusion is blocked in an early Cyber Kill Chain® stage and the organization's detections and mitigations are effective, is the organization also resilient? In other words, if it weren't identified and stopped at that point, would it have been caught lower down the kill chain? Were there layers of security in place that would protect an organization in a future attack if the adversary changed that one particular tactic, signature, or indicator? This is determined after synthesizing an attack through the entire Cyber Kill Chain® cycle – regardless of how early it is stopped. Understanding what would have happened had the adversary been successful is just as important as understanding what did happen. Understanding where defenses are in place at every stage of the kill chain – including those past the point of initial detection – is understanding how resilient the organization is. The goal is to be effective and block intrusions early in the Cyber Kill Chain®, but also to be resilient and have layers of countermeasures in place.

Once again a TIP can associate countermeasures to each stage of the Cyber Kill Chain® framework for every intrusion attempt. This includes those later stages that the attempted intrusions did not actually reach. The TIP can reveal the level of

resiliency for each investigated incident and could provide the mechanism to compare the resiliency against intrusions over a period of time. As with effectiveness, the trend would hopefully show that the organization is becoming more resilient over time, with multiple countermeasures in place for each intrusion attempt.

4.6 MEASURE ANALYTIC COMPLETENESS

As has been previously discussed, an analyst's investigation does not stop with identifying and stopping an attack. One must also analyze how the attack occurred, and synthesize what would have happened had it not been stopped. This complete understanding of the potential incident can be measured against the Cyber Kill Chain® framework, which can be used as a pseudo checklist to ensure the analyst has mined all possible information. The idea of pulling data from every piece of the investigation is a critical component of the Intelligence Driven Defense® approach, as it is what provides the intelligence that informs one's defenses.

The output of good analysis is a lot of data and intelligence. Managing that data and intelligence is a function of a TIP. Furthermore, a TIP that employs the Cyber Kill Chain® framework

for an investigation can help ensure analytic completeness. It can force the analyst to think about each step to encourage a complete understanding of the intrusion attempt: Do I understand each of the stages for this attack? Have I fully investigated, analyzed, and synthesized all the information I could possibly collect? A TIP should enable an analyst to become effective at computer network defense by providing a mechanism to produce one's own threat intelligence.

4.7 IDENTIFY AND TRACK CAMPAIGNS

The final way to apply the Cyber Kill Chain® process within an organization is to group intrusions from the same adversary together into campaigns. This broader view provides insight into when a particular adversary attacks, the TTPs they commonly use, and how effective and resilient an organization is against that adversary. An organization can then prioritize and measure themselves against each adversary to further enhance their defenses. Establishing campaigns also provides a standard nomenclature to improve communications. This enables analysts to be more efficient when identifying intrusions, understanding

potential next steps, and applying the proper defensive response.

An effective Threat Intelligence Platform enables analysts to determine patterns of malicious behavior learned from previous events to better address future attacks. All the data from previous investigations and the intelligence that is mined from analyzing and synthesizing intrusion attempts must be centrally stored and easy to search to enable identification of these patterns and quick recognition of relationships. Analysts connect intrusions by correlating indicators or TTPs from multiple Cyber Kill Chain® stages to determine that an attack is coming from a previously seen adversary. A powerful example of this is being able to recognize that the adversary scraping your external Web servers looking for documents is the same adversary who then sends a phishing email with a weaponized press release.

The TIP can facilitate this correlation because it holds all the intelligence and previous incident analysis.

Since the TIP is also the central location for all campaign mapping, a Campaign Heat Map or similar measurement of adversary activity should

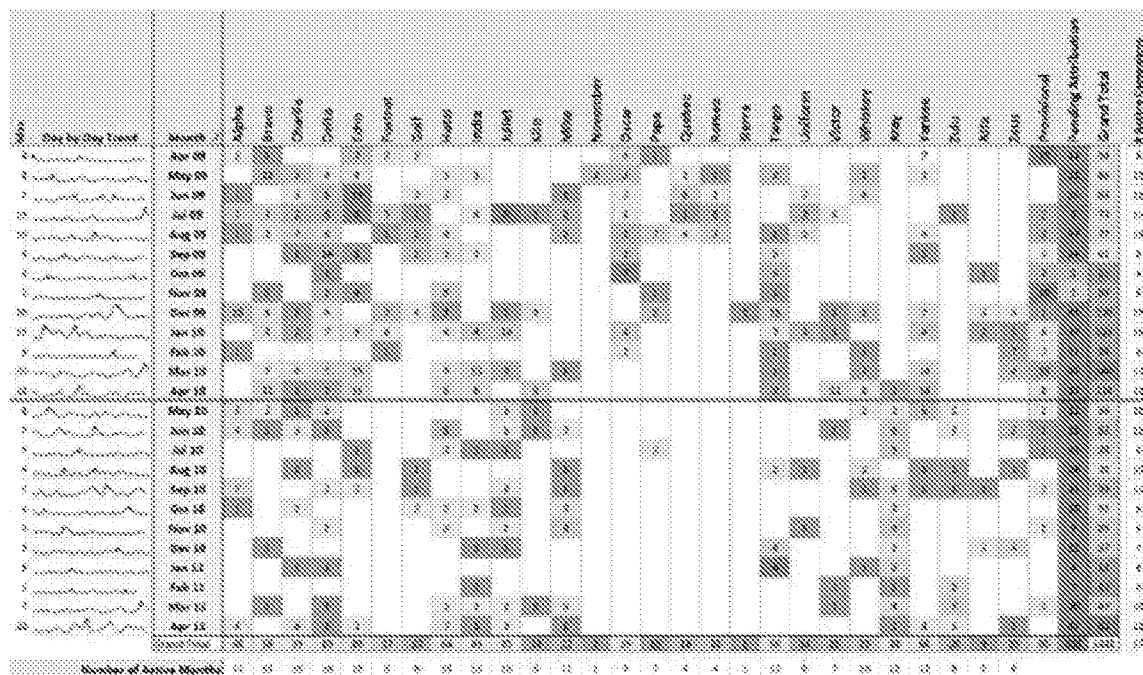


Figure 3. Campaign Heat Map

be an output of a TIP. This heat map can help organizations build a profile of past and current adversary activity, helping them better understand when, where, and how they will attack again in the future. A sample Campaign Heat Map is shown in Figure 3.

5. THE PALISADE™ PLATFORM

Lockheed Martin developed the Palisade™ solution to enable analysts to correlate malicious activity. It is not just an intelligence ingest engine or broker; it is a tool for analysts that supports the adoption of the Intelligence Driven Defense® methodology. Palisade™ is a centralized platform that integrates with an organization's SIEM to provide enterprise-wide alerting capability and manage all threat intelligence – internal and external. It also enables the tracking of countermeasures to ensure the organization is properly acting upon the intelligence that has been mined.

The Palisade™ solution empowers analysts to focus their efforts and adopt an Intelligent Driven Defense® approach by providing a platform for incident investigation and response that is aligned to the Cyber Kill Chain® framework. It allows for the identification of broader campaign activity through the connection of disparate events by leveraging historical intelligence from previous incidents. The Palisade™ platform is an enabler for a more effective approach to computer network defense – an approach that is fueled by analyst tradecraft and intelligence.

6. CONCLUSION

Threat Intelligence Platforms are an emerging technology that help an organization to consume and then act on cyber intelligence. Rather than relying solely on external intelligence, a TIP can also enable an organization to transition to producing their own actionable intelligence. This is a more reliable, sustainable, and cost-effective model for the long-term defenses of an organization. Properly employing a TIP can enhance an

organization's cyber maturity and improve resiliency against advanced adversaries through the adoption of an Intelligence Driven Defense® approach to computer network defense. This can mean the difference between a reactive and vulnerable security state and a proactive, or even predictive, computer network defense posture that is capable of identifying and stopping current attacks, and preventing future attacks.

References

Rick Holland. Threat Intelligence Buyer's Guide, 10 February 2014. URL

https://digital-forensics.sans.org/summit-archives/cti_summit2014/Threat_Intelligence_Buyers_Guide_Rick_Holland.pdf

Rick Holland. Threat Intelligence is Like Three Day Potty Training, April 2015.

URL https://www.rsaconference.com/writable/presentations/file_upload/cxo-t08r-threat-intelligence-is-like-three-day-potty-training.pdf

Eric Hutchins. Understanding the Cyber Kill Chain™: Applying Intelligence to Computer Network Defense, Rick Holland. Threat Intelligence is Like Three Day Potty Training, April 2015.

URL https://www.rsaconference.com/writable/presentations/file_upload/cxo-t08r-threat-intelligence-is-like-three-day-potty-training.pdf

Eric Hutchins, Michael Cloppert, Rohan Amin. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains, 2010. URL

<https://isgs-gen.external.lmco.com/sites/ECS/Marketing/White%20Papers%20and%20External%20Publications/Cyber%20Kill%20Chain%20Whitepaper.pdf>,

Craig Lawson. Technology Overview for Threat Intelligence Platforms, 10 December 2014. URL

<https://www.gartner.com/doc/2941522/technology-overview-threat-intelligence-platforms>

Rob McMillan. Definition: Threat Intelligence, 16 May 2013. URL

<https://www.gartner.com/doc/2487216/definition-threat-intelligence>

For more information on cybersecurity solutions
Email: cyber.security@lmco.com
Phone: 855-LMCYBER (856) 562-9237
www.lockheedmartin.com/cyber

PIRA# CMK201507003

© 2015 Lockheed Martin Corporation

Cyber Kill Chain Solution Cyber Threat Model © 2011 Lockheed Martin. All Rights Reserved.

LOCKHEED MARTIN, LOCKHEED, the STAR design, CYBER KILL CHAIN, LOCKHEED MARTIN CYBER KILL CHAIN, and INTELLIGENCE DRIVEN DEFENSE trademarks used throughout are registered trademarks in the U.S. Patent and Trademark Office owned by Lockheed Martin Corporation.

TRANSMITTAL FOR POWER OF ATTORNEY TO ONE OR MORE REGISTERED PRACTITIONERS

NOTE: This form is to be submitted with the Power of Attorney by Applicant form (PTO/AIA/82B) to identify the application to which the Power of Attorney is directed, in accordance with 37 CFR 1.5, unless the application number and filing date are identified in the Power of Attorney by Applicant form. If neither form PTO/AIA/82A nor form PTO/AIA82B identifies the application to which the Power of Attorney is directed, the Power of Attorney will not be recognized in the application.

Application Number	
Filing Date	
First Named Inventor	Anurag Agarwal
Title	Threat Model Chaining and Attack Simulation Systems and Related Methods
Art Unit	
Examiner Name	
Attorney Docket Number	ThreatM008

SIGNATURE of Applicant or Patent Practitioner

Signature	/Paul B. Johnson/	Date (Optional)	2018-12-20
Name	Paul B. Johnson	Registration Number	59375
Title (if Applicant is a juristic entity)			
Applicant Name (if Applicant is a juristic entity)			

NOTE: This form must be signed in accordance with 37 CFR 1.33. See 37 CFR 1.4(d) for signature requirements and certifications. If more than one applicant, use multiple forms.



*Total of 1 forms are submitted.

This collection of information is required by 37 CFR 1.131, 1.32, and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

POWER OF ATTORNEY BY APPLICANT

I hereby revoke all previous powers of attorney given in the application identified in either the attached transmittal letter or the boxes below:

Application Number

Filing Date

(Note: The boxes above may be left blank if information is provided on form PTO/AIA/82A.)



I hereby appoint the Patent Practitioner(s) associated with the following Customer Number as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above:

145042

OR



I hereby appoint Practitioner(s) named in the attached list (form PTO/AIA/82C) as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the patent application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above. (Note: Complete form PTO/AIA/82C.)

Please recognize or change the correspondence address for the application identified in the attached transmittal letter or the boxes above to:



The address associated with the above-mentioned Customer Number

OR



The address associated with Customer Number:

OR



Firm or
Individual Name

Address

City

State

Zip

Country

Telephone

Email

I am the Applicant (if the Applicant is a juristic entity, list the Applicant name in the box):

ThreatModeler Software Inc.



Inventor or Joint Inventor (title not required below)



Legal Representative of a Deceased or Legally Incapacitated Inventor (title not required below)



Assignee or Person to Whom the inventor is Under an Obligation to Assign (provide signer's title if applicant is a juristic entity)



Person Who Otherwise Shows Sufficient Proprietary Interest (e.g., a petition under 37 CFR 1.46(b)(2) was granted in the application or is concurrently being filed with this document) (provide signer's title if applicant is a juristic entity)

SIGNATURE of Applicant for Patent

The undersigned (whose title is supplied below) is authorized to act on behalf of the applicant (e.g., where the applicant is a juristic entity).

Signature

Date (Optional)

12/14/2018

Name

Anurag Agarwal

Title

CEO of ThreatModeler Software Inc. of NJ

NOTE: Signature - This form must be signed by the applicant in accordance with 37 CFR 1.33. See 37 CFR 1.4 for signature requirements and certifications. If more than one applicant, use multiple forms.



Total of 1 forms are submitted.

This collection of information is required by 37 CFR 1.131, 1.37, and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 5 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete the form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FILES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Electronic Patent Application Fee Transmittal

Application Number:

Filing Date:

Title of Invention:

Threat Model Chaining and Attack Simulation Systems and Related Methods

First Named Inventor/Applicant Name:

Anurag Agarwal

Filer:

Paul B. Johnson

Attorney Docket Number:

ThreatM008

Filed as Small Entity

Filing Fees for Utility under 35 USC 111(a)

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
Basic Filing:				
UTILITY FILING FEE (ELECTRONIC FILING)	4011	1	75	75
UTILITY SEARCH FEE	2111	1	330	330
UTILITY EXAMINATION FEE	2311	1	380	380

Pages:

Claims:

Miscellaneous-Filing:

Petition:

Patent-Appeals-and-Interference:

IriusRisk

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Miscellaneous:				
Total in USD (\$)				785

Electronic Acknowledgement Receipt

EFS ID:	34664814
Application Number:	16228738
International Application Number:	
Confirmation Number:	3702
Title of Invention:	Threat Model Chaining and Attack Simulation Systems and Related Methods
First Named Inventor/Applicant Name:	Anurag Agarwal
Customer Number:	145042
Filer:	Paul B. Johnson
Filer Authorized By:	
Attorney Docket Number:	ThreatM008
Receipt Date:	20-DEC-2018
Filing Date:	
Time Stamp:	23:18:25
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	CARD
Payment was successfully received in RAM	\$ 785
RAM confirmation Number	122118INTEFSW23192900
Deposit Account	
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

IriusRisk

File Listing:					
Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Application Data Sheet	WebADS.pdf	130893	no	8
			eed13869226b37bac129f5a894cc78ed4f8b827d		
Warnings:					
Information:					
2	Oath or Declaration filed	DEC008.pdf	59104	no	1
			ff592d3878ccacdff2513c6b65b08f3f3bf7fcac		
Warnings:					
Information:					
3	Drawings-only black and white line drawings	Drawings008.pdf	2993564	no	22
			d431922264d5e2a352420e2e82eac268bfd0fa73		
Warnings:					
Information:					
4		UtilityApp008.pdf	322292	yes	54
			6ac4fb00efbc3e333c19d3ec335f907a950fba31		
	Multipart Description/PDF files in .zip description				
	Document Description		Start	End	
	Abstract		54	54	
	Claims		48	53	
	Specification		1	47	
Warnings:					
Information:					
5	Information Disclosure Statement (IDS) Form (SB08)	IDS1.pdf	1035752	no	6
			1d167eb243ecd80ba9a65c58320b2b99305cca53		
				IriusRisk	

Warnings:					
Information:					
6	Non Patent Literature	NPL01.pdf	328259	no	12
			571e01737a3f1ae312f9da4f512dbda505ec e66f		
Warnings:					
Information:					
7	Non Patent Literature	NPL02.pdf	334884	no	6
			566bd40ca5886bf7002becae8b261a6d89e 5a2e8		
Warnings:					
Information:					
8	Non Patent Literature	NPL03.pdf	166989	no	5
			4bf1903d6f4f870e231d3edac4d5e7fd21c5 e13b		
Warnings:					
Information:					
9	Non Patent Literature	NPL04.pdf	1592606	no	190
			277a72c180f25e15c9a997353566cba5ac32 b3bc		
Warnings:					
Information:					
10	Non Patent Literature	NPL05.pdf	944170	no	25
			04f7192c24adf4626e02e0ad346ca2e838f0 93c7		
Warnings:					
Information:					
11	Non Patent Literature	NPL06.pdf	967495	no	12
			d91c689b892a8dd53787024c0b24d12494 ce0c66		
Warnings:					
Information:					
12	Power of Attorney	POA008.pdf	1425781	no	2
			36ce907c0f5d2b8d2da1d34299ade1f31ed 8bca7		
Warnings:					
Information:					
IriusRisk					

13	Fee Worksheet (SB06)	fee-info.pdf	35088	no	2
			7efd773e393ad9221a97395c29e400c104a4f8af		

Warnings:

Information:

Total Files Size (in bytes): 10336877

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	ThreatM008
		Application Number	
Title of Invention	Threat Model Chaining and Attack Simulation Systems and Related Methods		
The application data sheet is part of the provisional or nonprovisional application for which it is being submitted. The following form contains the bibliographic data arranged in a format specified by the United States Patent and Trademark Office as outlined in 37 CFR 1.76. This document may be completed electronically and submitted to the Office in electronic format using the Electronic Filing System (EFS) or the document may be printed and included in a paper filed application.			

Secrecy Order 37 CFR 5.2:

☐ Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2 (Paper filers only. Applications that fall under Secrecy Order may not be filed electronically.)
--

Inventor Information:

Inventor 1				
Legal Name				
Prefix	Given Name	Middle Name	Family Name	Suffix
	Anurag		Agarwal	
Residence Information (Select One) ☒ US Residency ☐ Non US Residency ☐ Active US Military Service				
City	Jersey City	State/Province	NJ	Country of Residence ⁱ US
Mailing Address of Inventor:				
Address 1	101 Hudson St. Ste. 2100			
Address 2				
City	Jersey City	State/Province	NJ	
Postal Code	07302	Country ⁱ	US	
All Inventors Must Be Listed - Additional Inventor Information blocks may be generated within this form by selecting the Add button. Add				

Correspondence Information:

Enter either Customer Number or complete the Correspondence Information section below. For further information see 37 CFR 1.33(a).			
☐ An Address is being provided for the correspondence information of this application.			
Customer Number	145042		
Email Address	paul@pauljohnsonlaw.com	Add Email	Remove Email
Email Address	pbjdocket@gmail.com	Add Email	Remove Email

Application Information:

Title of the Invention	Threat Model Chaining and Attack Simulation Systems and Related Methods		
Attorney Docket Number	ThreatM008	Small Entity Status Claimed	☒
Application Type	Nonprovisional		
Subject Matter	Utility		
Total Number of Drawing Sheets (if any)	22	Suggested Figure for Publication (if any)	1

IriusRisk

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	ThreatM008
		Application Number	
Title of Invention	Threat Model Chaining and Attack Simulation Systems and Related Methods		

Filing By Reference:

Only complete this section when filing an application by reference under 35 U.S.C. 111(c) and 37 CFR 1.57(a). Do not complete this section if application papers including a specification and any drawings are being filed. Any domestic benefit or foreign priority information must be provided in the appropriate section(s) below (i.e., "Domestic Benefit/National Stage Information" and "Foreign Priority Information").

For the purposes of a filing date under 37 CFR 1.53(b), the description and any drawings of the present application are replaced by this reference to the previously filed application, subject to conditions and requirements of 37 CFR 1.57(a).

Application number of the previously filed application	Filing date (YYYY-MM-DD)	Intellectual Property Authority or Country

Publication Information:

☐ Request Early Publication (Fee required at time of Request 37 CFR 1.219)

☐ **Request Not to Publish.** I hereby request that the attached application not be published under 35 U.S.C. 122(b) and certify that the invention disclosed in the attached application **has not and will not** be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication at eighteen months after filing.

Representative Information:

Representative information should be provided for all practitioners having a power of attorney in the application. Providing this information in the Application Data Sheet does not constitute a power of attorney in the application (see 37 CFR 1.32). Either enter Customer Number or complete the Representative Name section below. If both sections are completed the customer Number will be used for the Representative Information during processing.

Please Select One:						
☒ Customer Number		☐ US Patent Practitioner		☐ Limited Recognition (37 CFR 11.9)		
Customer Number		145042				
Prefix	Given Name	Middle Name	Family Name	Suffix	Remove	
Registration Number						
Prefix	Given Name	Middle Name	Family Name	Suffix	Remove	
Registration Number						
Additional Representative Information blocks may be generated within this form by selecting the Add button.						

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	ThreatM008
		Application Number	
Title of Invention	Threat Model Chaining and Attack Simulation Systems and Related Methods		

Domestic Benefit/National Stage Information:

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121, 365(c), or 386(c) or indicate National Stage entry from a PCT application. Providing benefit claim information in the Application Data Sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78.

When referring to the current application, please leave the "Application Number" field blank.

Prior Application Status	Pending	Remove	
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)
	Continuation in part of	15922856	2018-03-15
Prior Application Status	Pending	Remove	
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)
15922856	Continuation in part of	15888021	2018-02-03
Prior Application Status	Expired	Remove	
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)
15888021	Claims benefit of provisional	62507691	2017-05-17
Prior Application Status	Expired	Remove	
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)
15888021	Claims benefit of provisional	62527671	2017-06-30
Prior Application Status	Expired	Remove	
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)
15888021	Claims benefit of provisional	62530295	2017-07-10
Prior Application Status	Expired	Remove	
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)
15888021	Claims benefit of provisional	62520954	2017-06-16
Additional Domestic Benefit/National Stage Data may be generated within this form by selecting the Add button.			

Foreign Priority Information:

IriusRisk

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	ThreatM008
		Application Number	
Title of Invention	Threat Model Chaining and Attack Simulation Systems and Related Methods		

This section allows for the applicant to claim priority to a foreign application. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55. When priority is claimed to a foreign application that is eligible for retrieval under the priority document exchange program (PDX) the information will be used by the Office to automatically attempt retrieval pursuant to 37 CFR 1.55(i)(1) and (2). Under the PDX program, applicant bears the ultimate responsibility for ensuring that a copy of the foreign application is received by the Office from the participating foreign intellectual property office, or a certified copy of the foreign priority application is filed, within the time period specified in 37 CFR 1.55(g)(1).

Application Number	Country ⁱ	Filing Date (YYYY-MM-DD)	Access Code ⁱ (if applicable)

Additional Foreign Priority Data may be generated within this form by selecting the **Add** button.

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications

This application (1) claims priority to or the benefit of an application filed before March 16, 2013 and (2) also contains, or contained at any time, a claim to a claimed invention that has an effective filing date on or after March 16, 2013.

☐ NOTE: By providing this statement under 37 CFR 1.55 or 1.78, this application, with a filing date on or after March 16, 2013, will be examined under the first inventor to file provisions of the AIA.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	ThreatM008
		Application Number	
Title of Invention	Threat Model Chaining and Attack Simulation Systems and Related Methods		

Authorization or Opt-Out of Authorization to Permit Access:

When this Application Data Sheet is properly signed and filed with the application, applicant has provided written authority to permit a participating foreign intellectual property (IP) office access to the instant application-as-filed (see paragraph A in subsection 1 below) and the European Patent Office (EPO) access to any search results from the instant application (see paragraph B in subsection 1 below).

Should applicant choose not to provide an authorization identified in subsection 1 below, applicant **must opt-out** of the authorization by checking the corresponding box A or B or both in subsection 2 below.

NOTE: This section of the Application Data Sheet is **ONLY** reviewed and processed with the **INITIAL** filing of an application. After the initial filing of an application, an Application Data Sheet cannot be used to provide or rescind authorization for access by a foreign IP office(s). Instead, Form PTO/SB/39 or PTO/SB/69 must be used as appropriate.

1. Authorization to Permit Access by a Foreign Intellectual Property Office(s)

A. Priority Document Exchange (PDX) - Unless box A in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the European Patent Office (EPO), the Japan Patent Office (JPO), the Korean Intellectual Property Office (KIPO), the State Intellectual Property Office of the People's Republic of China (SIPO), the World Intellectual Property Organization (WIPO), and any other foreign intellectual property office participating with the USPTO in a bilateral or multilateral priority document exchange agreement in which a foreign application claiming priority to the instant patent application is filed, access to: (1) the instant patent application-as-filed and its related bibliographic data, (2) any foreign or domestic application to which priority or benefit is claimed by the instant application and its related bibliographic data, and (3) the date of filing of this Authorization. See 37 CFR 1.14(h)(1).

B. Search Results from U.S. Application to EPO - Unless box B in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the EPO access to the bibliographic data and search results from the instant patent application when a European patent application claiming priority to the instant patent application is filed. See 37 CFR 1.14(h)(2).

The applicant is reminded that the EPO's Rule 141(1) EPC (European Patent Convention) requires applicants to submit a copy of search results from the instant application without delay in a European patent application that claims priority to the instant application.

2. Opt-Out of Authorizations to Permit Access by a Foreign Intellectual Property Office(s)

☐ A. Applicant **DOES NOT** authorize the USPTO to permit a participating foreign IP office access to the instant application-as-filed. If this box is checked, the USPTO will not be providing a participating foreign IP office with any documents and information identified in subsection 1A above.

☐ B. Applicant **DOES NOT** authorize the USPTO to transmit to the EPO any search results from the instant patent application. If this box is checked, the USPTO will not be providing the EPO with search results from the instant application.

NOTE: Once the application has published or is otherwise publicly available, the USPTO may provide access to the application in accordance with 37 CFR 1.14.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	ThreatM008
	Application Number	
Title of Invention	Threat Model Chaining and Attack Simulation Systems and Related Methods	

Applicant Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

Applicant 1

If the applicant is the inventor (or the remaining joint inventor or inventors under 37 CFR 1.45), this section should not be completed. The information to be provided in this section is the name and address of the legal representative who is the applicant under 37 CFR 1.43; or the name and address of the assignee, person to whom the inventor is under an obligation to assign the invention, or person who otherwise shows sufficient proprietary interest in the matter who is the applicant under 37 CFR 1.46. If the applicant is an applicant under 37 CFR 1.46 (assignee, person to whom the inventor is obligated to assign, or person who otherwise shows sufficient proprietary interest) together with one or more joint inventors, then the joint inventor or inventors who are also the applicant should be identified in this section.

[Clear](#)

☒ Assignee ☐ Legal Representative under 35 U.S.C. 117 ☐ Joint Inventor

☐ Person to whom the inventor is obligated to assign. ☐ Person who shows sufficient proprietary interest

If applicant is the legal representative, indicate the authority to file the patent application, the inventor is:

Name of the Deceased or Legally Incapacitated Inventor:

If the Applicant is an Organization check here. ☒

Organization Name ThreatModeler Software Inc.

Mailing Address Information For Applicant:

Address 1	101 Hudson St., Ste. 2100		
Address 2			
City	Jersey City	State/Province	NJ
Country ⁱ	US	Postal Code	07302
Phone Number		Fax Number	
Email Address			

Additional Applicant Data may be generated within this form by selecting the Add button.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	ThreatM008
		Application Number	
Title of Invention	Threat Model Chaining and Attack Simulation Systems and Related Methods		

Assignee Information including Non-Applicant Assignee Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

Assignee 1						
Complete this section if assignee information, including non-applicant assignee information, is desired to be included on the patent application publication. An assignee-applicant identified in the "Applicant Information" section will appear on the patent application publication as an applicant. For an assignee-applicant, complete this section only if identification as an assignee is also desired on the patent application publication.						
If the Assignee or Non-Applicant Assignee is an Organization check here.			☒			
Organization Name	ThreatModeler Software Inc.					
Mailing Address Information For Assignee including Non-Applicant Assignee:						
Address 1		101 Hudson St., Ste. 2100				
Address 2						
City	Jersey City	State/Province	NJ			
Country i	US	Postal Code	07302			
Phone Number		Fax Number				
Email Address						
Additional Assignee or Non-Applicant Assignee Data may be generated within this form by selecting the Add button.						

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	ThreatM008
		Application Number	
Title of Invention	Threat Model Chaining and Attack Simulation Systems and Related Methods		

Signature:

NOTE: This Application Data Sheet must be signed in accordance with 37 CFR 1.33(b). **However, if this Application Data Sheet is submitted with the INITIAL filing of the application and either box A or B is not checked in subsection 2 of the "Authorization or Opt-Out of Authorization to Permit Access" section, then this form must also be signed in accordance with 37 CFR 1.14(c).**

This Application Data Sheet **must** be signed by a patent practitioner if one or more of the applicants is a **juristic entity** (e.g., corporation or association). If the applicant is two or more joint inventors, this form must be signed by a patent practitioner, **all** joint inventors who are the applicant, or one or more joint inventor-applicants who have been given power of attorney (e.g., see USPTO Form PTO/AIA/81) on behalf of **all** joint inventor-applicants.

See 37 CFR 1.4(d) for the manner of making signatures and certifications.

Signature	/Paul B. Johnson/			Date (YYYY-MM-DD)	
First Name	Paul	Last Name	Johnson	Registration Number	59375
Additional Signature may be generated within this form by selecting the Add button.					

Declaration for Patent Application

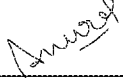
As the below named inventor, I hereby declare that:

This declaration is directed to the attached application.

The above-identified application was made or authorized to be made by me. I believe that I am the original inventor or an original joint inventor of a claimed invention in the application.

I acknowledge that any willful false statement made in this declaration is punishable under 18 U.S.C. 1001 by fine or imprisonment of not more than five (5) years, or both.

Legal Name: Anurag Agarwal
Print or Type

Signature: 

Date: 12/14/2018

Mailing Address: 101 Hudson St Ste 2100
Jersey City NJ 07302

Residence:

☐ Same as mailing address, or:
105 Greene St Unit 1801
Jersey City NJ 07302

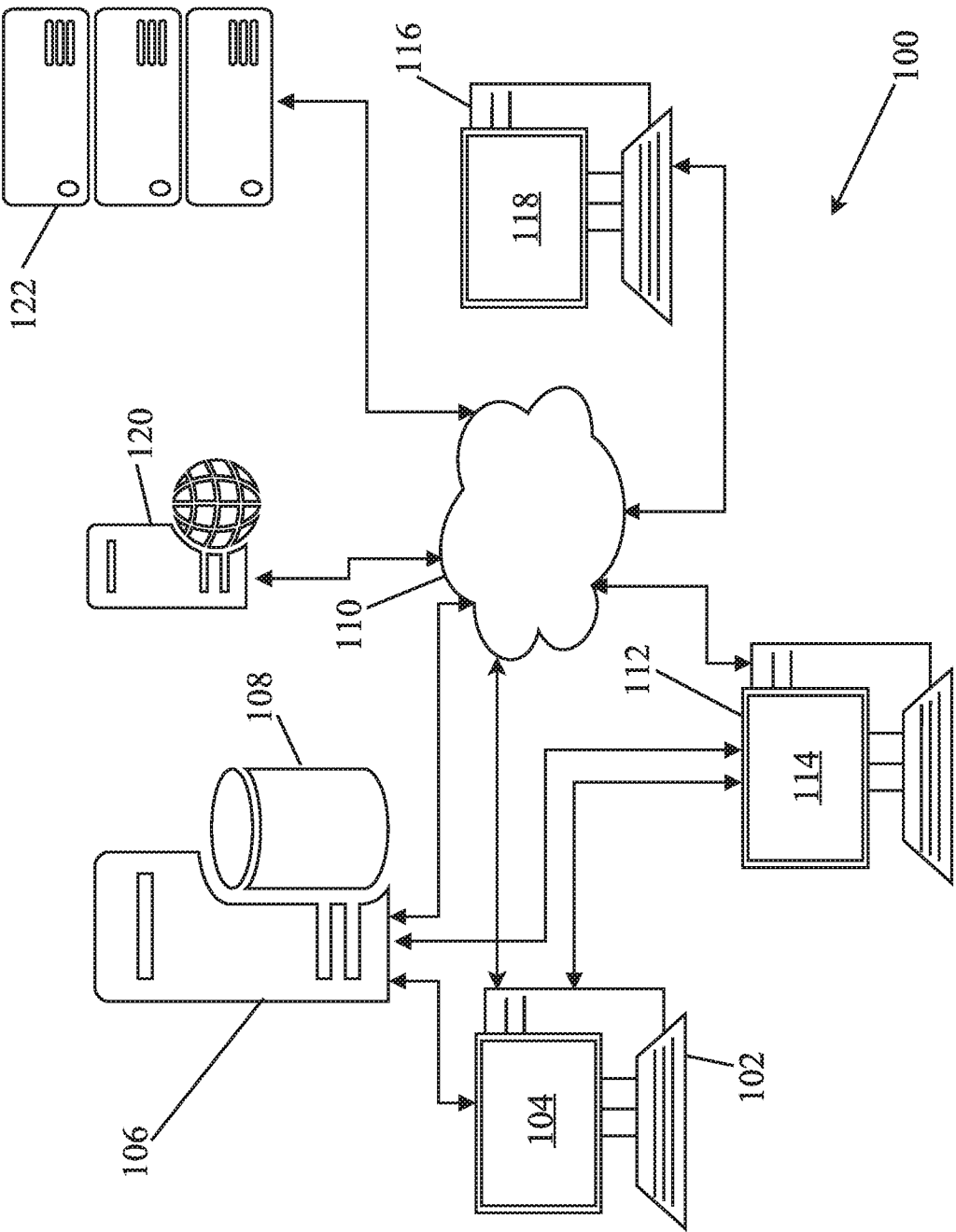
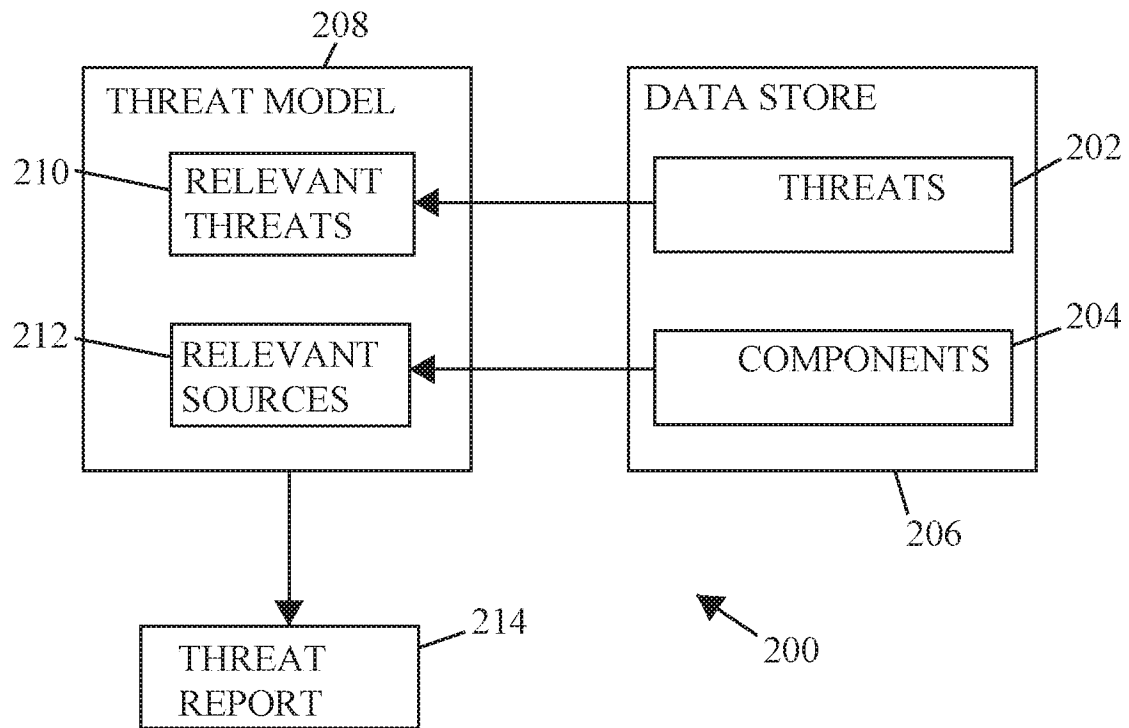


FIG. 1

**FIG. 2**

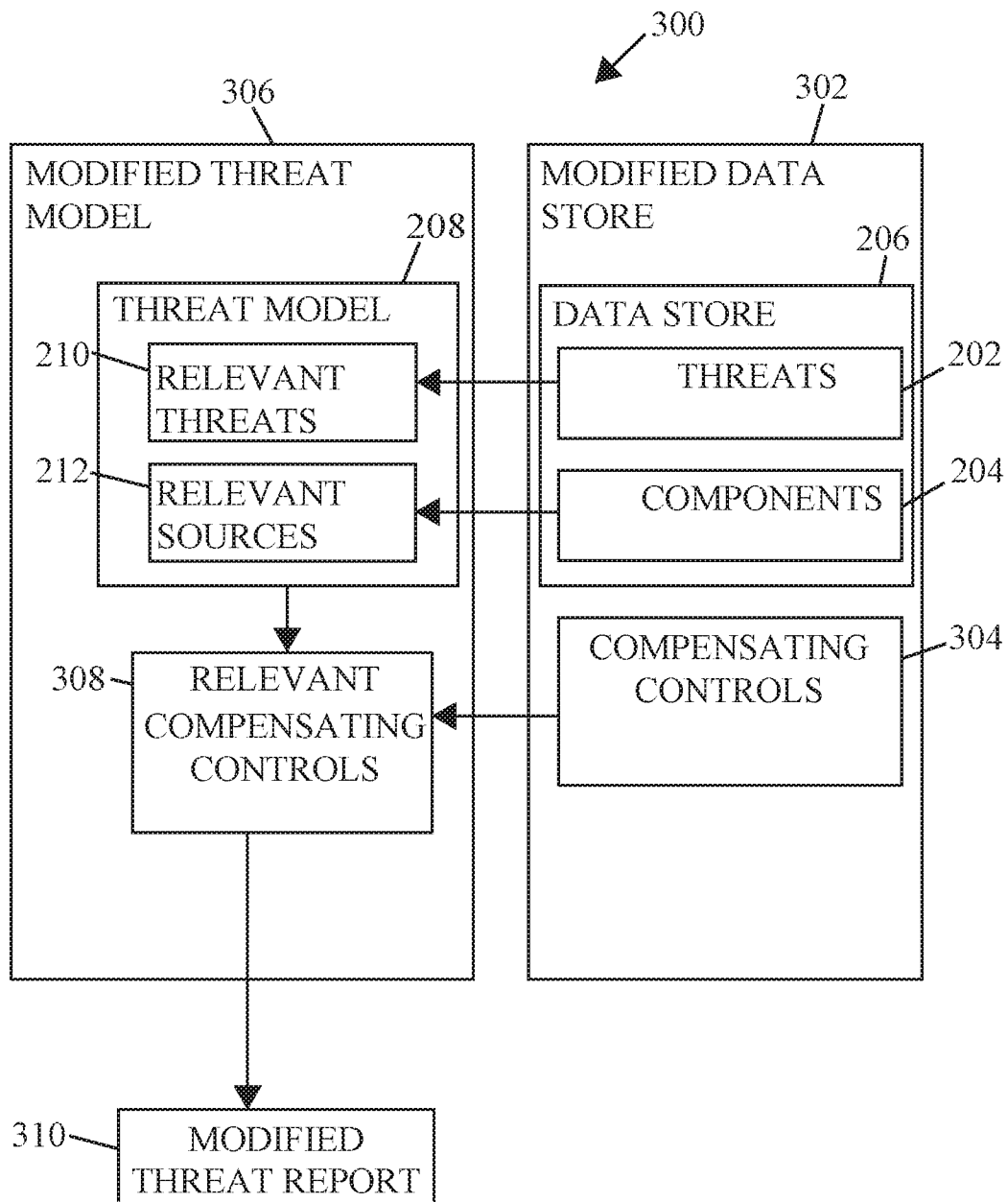


FIG. 3

4/22

400

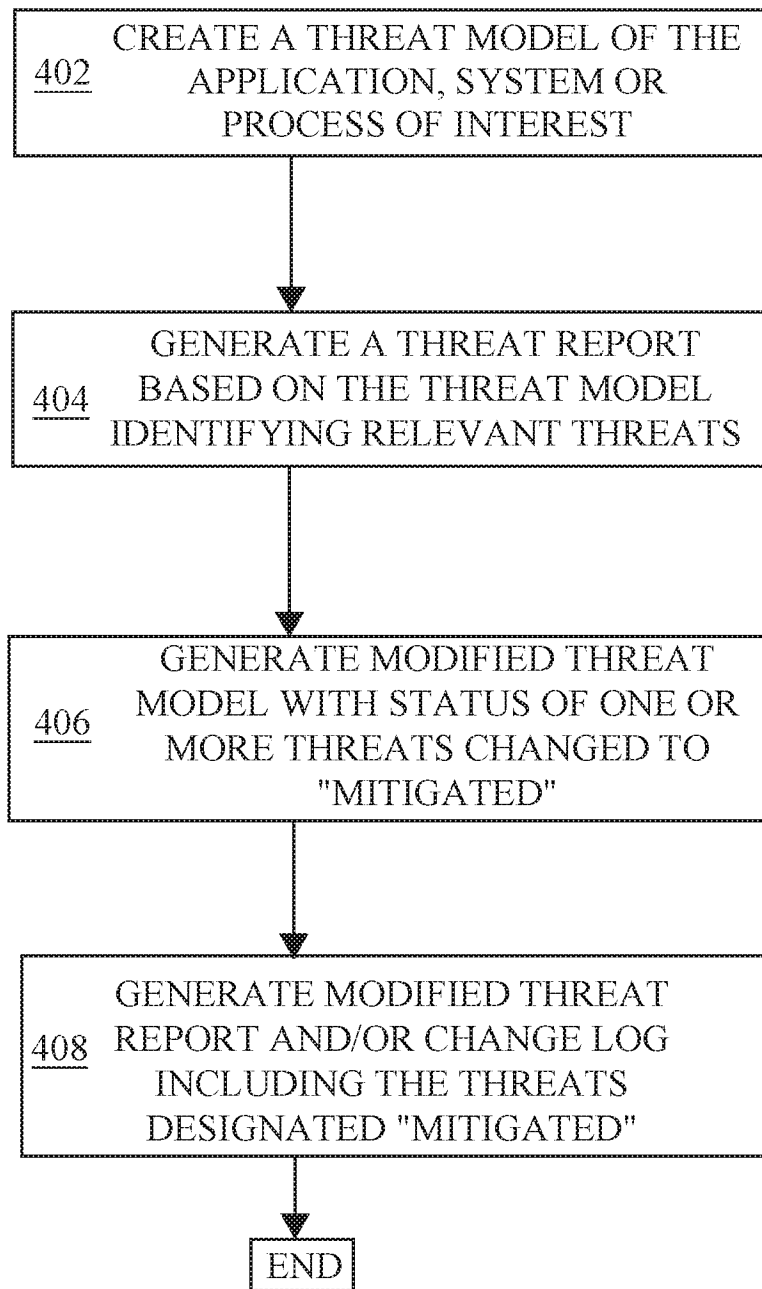


FIG. 4

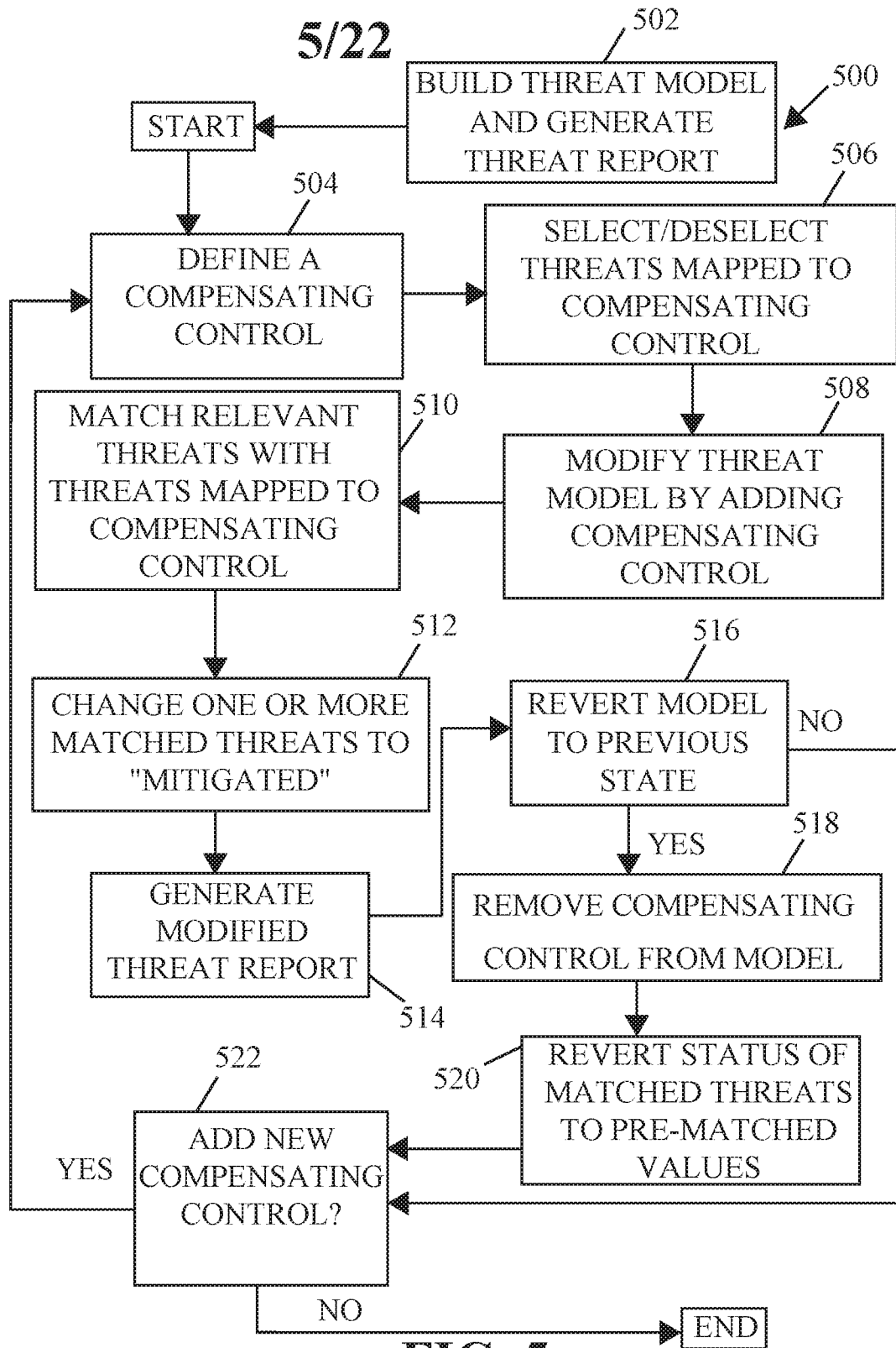


FIG. 5

6/22

SELECT ALL NEW DELETE COPY TO LIBRARY EDIT SELECT LIBRARY THREAT MODELER ▼				HOME KNOWLEDGE BASE TEMPLATES PROFILE		CHANGE PASSWORD SIGN OUT	
COMPONENTS				LOGIN		PROPERTIES	
NAME	TYPE	LABELS		TYPE: APPLICATION COMPONENT		THREATS +	
				WEB APPLICATION		INDUCING X	
WIFI TR	DEVICE			PASSWORD		ACCT LOCKOUT	
MODEM	DEVICE	EMBED. SYS.		AUTHENTICATION		SESSION X	
WIFI PORT	IOT DEVICE	IOT		FEATURE		HIJACKING	
HTTPS	PROTOCOLS			DESCRIPTION		CLICK X	
COOKIES	GEN DT STORE	TMT				JACKING	
DATABASE	DB	HARDWARE				SECURITY +	
LOGIN	APP. COMPONENT.	WEB, APP, FEAT.				REQUIREMENTS	
COMMENT	APP. COMPONENT.	WEB, APP, FEAT.				RANDOMIZING X	
SHOP CART	APP. COMPONENT.	WEB, APP, FEAT.				SECRET	
ACH TRANS.	APP. COMPONENT.	WEB, APP, FEAT.				ENCRYPTION X	
USB PORT	IOT DEVICE	IOT				ALGORITHM	
FILE SVR	DEPLOY. COMP.					PARAM. X	
EMAIL SVR	DEPLOY. COMP.					DYNAMIC SQL	
FIREWALL	APP. COMPONENT.						

FIG. 6

600

7/22

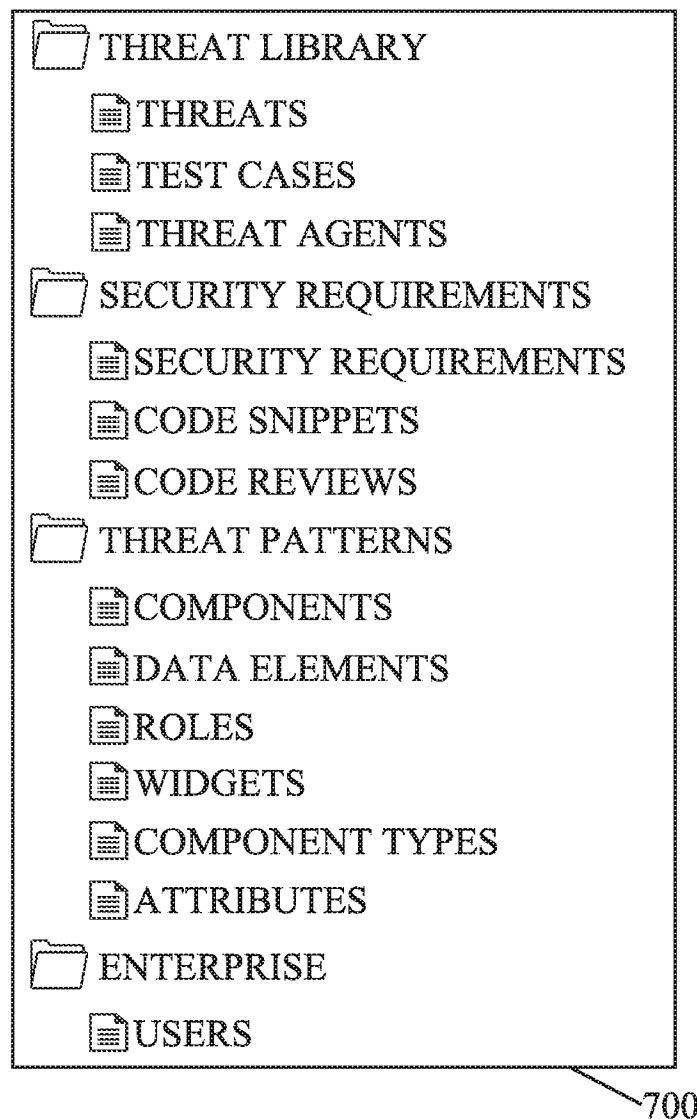


FIG. 7

SELECT LIBRARY THREAT MODELER ▼																																																																			
<table border="1"> <thead> <tr> <th colspan="2">THREATS</th> <th>SENSITIVE DATA</th> <th>PROPERTIES</th> </tr> </thead> <tbody> <tr> <td>NAME</td> <td>RISK</td> <td>EXPOSURE</td> <td>THREAT AGENTS</td> </tr> <tr> <td></td> <td></td> <td>RISK: VERY HIGH</td> <td></td> </tr> <tr> <td>BUFFER OVERFLOW</td> <td>VERY HIGH</td> <td>OWASP-A6</td> <td></td> </tr> <tr> <td>SESSION HIJACK</td> <td>VERY HIGH</td> <td>CAPEC-610</td> <td></td> </tr> <tr> <td>PHYSICAL THEFT</td> <td>VERY HIGH</td> <td rowspan="4"> OWASP-A5- SECURITY MIS- CONFIGURATION DESCRIPTION WWW.OWASP.ORG </td> <td>TEST CASES</td> </tr> <tr> <td>HTTP SPLITTING</td> <td>HIGH</td> <td>TEST CASE 01</td> </tr> <tr> <td>BRUTE FORCE</td> <td>HIGH</td> <td>TEST CASE 02</td> </tr> <tr> <td>DBL ENCODING</td> <td>MEDIUM</td> <td>TEST CASE 03</td> </tr> <tr> <td>SENS. DATA EXP.</td> <td>VERY HIGH</td> <td></td> <td>SECURITY REQUIREMENTS</td> </tr> <tr> <td>INTEGER ATTACKS</td> <td>MEDIUM</td> <td></td> <td>CLICKJACK CHEAT SHEET</td> </tr> <tr> <td>POINTER ATTACK</td> <td>MEDIUM</td> <td></td> <td>TEST COMP CONTROL</td> </tr> <tr> <td>SYMLINK ATTACK</td> <td>MEDIUM</td> <td></td> <td>SS-COMPENS. CONTROL</td> </tr> <tr> <td>EMAIL INJECT.</td> <td>MEDIUM</td> <td></td> <td></td> </tr> <tr> <td>FORMAT STR. INJ.</td> <td>HIGH</td> <td></td> <td></td> </tr> <tr> <td>AUTOMATION ATK</td> <td>VERY HIGH</td> <td></td> <td></td> </tr> <tr> <td>SPAM</td> <td>VERY HIGH</td> <td></td> <td></td> </tr> </tbody> </table>			THREATS		SENSITIVE DATA	PROPERTIES	NAME	RISK	EXPOSURE	THREAT AGENTS			RISK: VERY HIGH		BUFFER OVERFLOW	VERY HIGH	OWASP-A6		SESSION HIJACK	VERY HIGH	CAPEC-610		PHYSICAL THEFT	VERY HIGH	OWASP-A5- SECURITY MIS- CONFIGURATION DESCRIPTION WWW.OWASP.ORG	TEST CASES	HTTP SPLITTING	HIGH	TEST CASE 01	BRUTE FORCE	HIGH	TEST CASE 02	DBL ENCODING	MEDIUM	TEST CASE 03	SENS. DATA EXP.	VERY HIGH		SECURITY REQUIREMENTS	INTEGER ATTACKS	MEDIUM		CLICKJACK CHEAT SHEET	POINTER ATTACK	MEDIUM		TEST COMP CONTROL	SYMLINK ATTACK	MEDIUM		SS-COMPENS. CONTROL	EMAIL INJECT.	MEDIUM			FORMAT STR. INJ.	HIGH			AUTOMATION ATK	VERY HIGH			SPAM	VERY HIGH		
THREATS		SENSITIVE DATA	PROPERTIES																																																																
NAME	RISK	EXPOSURE	THREAT AGENTS																																																																
		RISK: VERY HIGH																																																																	
BUFFER OVERFLOW	VERY HIGH	OWASP-A6																																																																	
SESSION HIJACK	VERY HIGH	CAPEC-610																																																																	
PHYSICAL THEFT	VERY HIGH	OWASP-A5- SECURITY MIS- CONFIGURATION DESCRIPTION WWW.OWASP.ORG	TEST CASES																																																																
HTTP SPLITTING	HIGH		TEST CASE 01																																																																
BRUTE FORCE	HIGH		TEST CASE 02																																																																
DBL ENCODING	MEDIUM		TEST CASE 03																																																																
SENS. DATA EXP.	VERY HIGH		SECURITY REQUIREMENTS																																																																
INTEGER ATTACKS	MEDIUM		CLICKJACK CHEAT SHEET																																																																
POINTER ATTACK	MEDIUM		TEST COMP CONTROL																																																																
SYMLINK ATTACK	MEDIUM		SS-COMPENS. CONTROL																																																																
EMAIL INJECT.	MEDIUM																																																																		
FORMAT STR. INJ.	HIGH																																																																		
AUTOMATION ATK	VERY HIGH																																																																		
SPAM	VERY HIGH																																																																		

FIG. 8

800