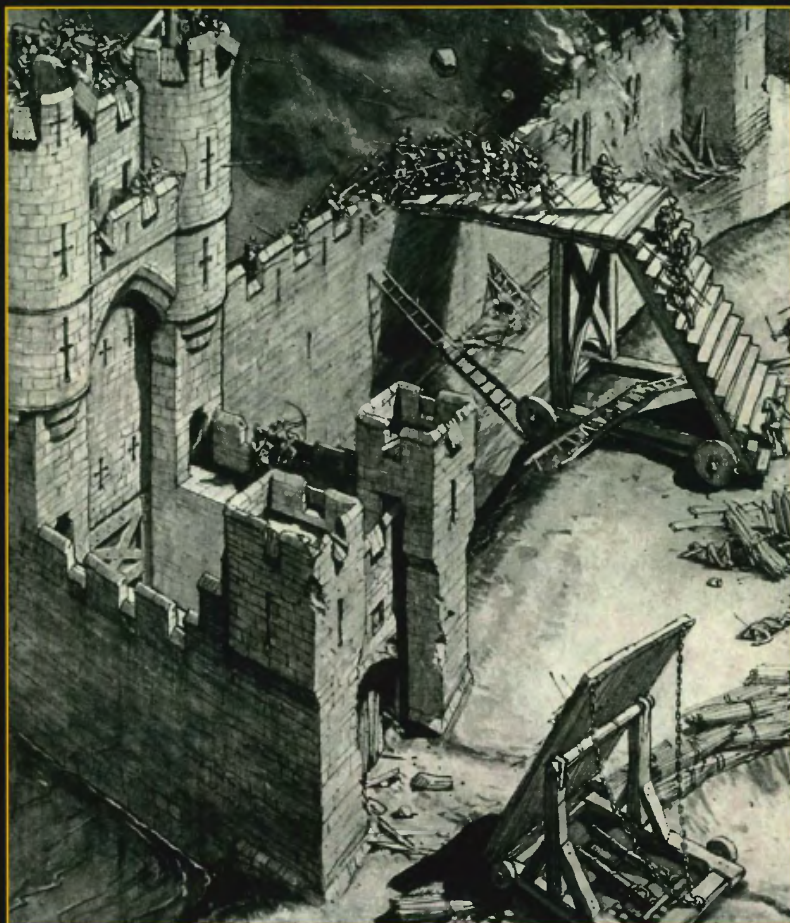




RISK CENTRIC THREAT MODELING

Process for Attack Simulation and Threat Analysis

Tony Uceda Vélez • Marco M. Morana

WILEY

Copyright © 2015 by John Wiley & Sons, Inc. All rights reserved

Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4470, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic formats. For more information about Wiley products, visit our web site at www.wiley.com.

Library of Congress Cataloging-in-Publication Data:

Tony UcedaVélez
Risk Centric Threat Modeling : process for attack simulation and threat analysis / Tony UcedaVélez,
Marco M. Morana
pages cm

Summary: "This book describes how to apply application threat modeling as an advanced preventive form of security"— Provided by publisher.

Includes bibliographical references and index.

ISBN 978-0-470-50096-5 (hardback)

1. Data protection. 2. Computer security. 3. Management information systems—Security measures.
4. Computer networks—Security measures. 5. Risk assessment. I. UcedaVélez, Tony, 1976- II. Title.
HF5548.37.M67 2015

658.4'7011—dc23

2015000692

Cover Image: Courtesy of Fromold Books, <http://www.fromoldbooks.org/>
Typeset in 10pt/12pt TimesLTStd by SPi Global, Chennai, India

1 0 9 8 7 6 5 4 3 2 1

1 2015

2

OBJECTIVES AND BENEFITS OF THREAT MODELING

DEFINING A RISK MITIGATION STRATEGY

“There are known unknowns; that is to say there are things that we now know we don’t know.

But there are also unknown unknowns – there are things we do not know, we don’t know.”

United States Secretary of Defense Donald Rumsfeld

In today’s digital economy, businesses provide valuable information and services to their customers online. The value of this information might vary depending on different factors such as the sensitivity of the information content such as intellectual property and confidential data. Since sensitive data flows through online channels and between the customer web and mobile clients and the web applications that are managed by the businesses, it is suitable target by value driven threat actors. Examples include cyber-criminals seeking to steal confidential data from bank customers for committing various crimes such as identify theft, stealing money from bank accounts, account take over, money laundering, credit/debit card counterfeiting, and online fraud. Fraudsters have an arsenal of cybercrime tools at their disposal for targeting online bank applications and bank customers. Online banking money movement transactions such as transferring money between bank accounts, for example, are targeted by fraudsters using banking malware. Personal customer details are targeted for identity theft and to impersonate the real customers in online banking transactions. Credit card data is targeted for online fraud such as for card-non-present transactions

Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis, First Edition.
Tony UcedaVélez and Marco M. Morana.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

and debit card data is targeted for counterfeit debit cards and to withdraw money from the victim's bank accounts.

The Importance of Threat Analysis and Risk Management

To protect customer personal data and business sensitive data from cyber-criminals and fraudsters, businesses typically enforce information security processes in compliance with information security standards and risk management processes to manage the cyberthreat risks. The initial step toward enforcing information security policies is typically to create an inventory of assets that need to be protected with an assignment of the level of risk. This level of risk depends on the sensitivity levels of the information assets. Once the data sensitivity is classified according to the confidentiality levels (e.g. public, internal, confidential, and restricted), it is possible to enforce the necessary levels of protection for the sensitive and confidential data. For example, a certain business might decide that all customer sensitive information should be protected with encryption when it is transferred outside the business such as when it is processed online by customer's use of web applications. For banks and for what concern online banking transactions that allow making purchases and money transfers, there is a risk of these being targeted for compromise by fraudsters, therefore banks will implement additional levels of security such as multifactor authentication as well as additional types of data validations. The risk associated with these money movement transactions might vary depending on the money amount that can be transacted online and have associated different level of security control and transactional risk. For example, a payment limit of small money amounts for online purchases can be considered low risk per each user-transaction while a transfer of money from one person bank account to another person bank account might be considered high risk. Once the risk associated with the value of the data and the value of the business transaction is being assessed, the next step consists of determining if the level of security controls that are put in place is sufficient. Typically, the higher the value of the asset, the higher is the level of risk and the stronger is the level of protection required.

In the case of sensitive data, the determination of risk level depends on different factors such as the type of threats targeting specific data attributes such as confidentiality, integrity, and availability of the data. The level of risk of the data also depends on the presence of vulnerabilities on the security controls such as encryption, authentication, and authorization that are put in place to protect the data and the access to this data from unauthorized malicious threat actors. Such vulnerabilities might be the ones that are exploited by threat actors to compromise the data and the business functions. Since vulnerabilities represent a "condition" for an exploit and a threat represent a "possibility" to cause an impact on the data and functional assets it is important to determine the level of risk of the threat by association of factors such as threats, vulnerabilities, and assets. By analyzing and associating these factors, a risk manager can determine the overall level of risk. Threats can be analyzed based upon information of the type of threat agent, his or her capabilities, and motives. This is the "who" factor for threat analysis. The other aspect is to analyze the "how," that is, the

type of attacks that are used including the types of attack techniques, tools, tactics, and procedures. The final aspect of threat analysis is the “what,” that is, what type of assets and vulnerabilities are targeted.

Once assets, threats, attacks, and vulnerabilities have been analyzed, the next step consists of determining the level of impact of the threats being analyzed on the company assets and to determine the level of risk. Besides impact, such level of risk depends on factors of probability of the threat to be realized. The determination of the level of risk based upon the factors of likelihood and impact is the main goal of risk analysis. Once risks have been analyzed, the next step consists of deciding how to manage this risk to keep it at a level that is considered acceptable by the business/organization. At this stage, risk decisions are made on how to handle the risk to make it acceptable by the business, examples include apply countermeasures, to reduce the risk likelihood and impact, transferring the risk to another party, avoiding the risk by either not exposing the asset to the threat or not implementing the application feature that will increase exposure and do nothing, that is, to accept the risk based on the realization of compensating controls and measures.

Threat Intelligence

Knowing which threats might target the business’s digital assets and how a compromise of these assets might affect the business is essential for determining the level of risk that the organization/business might face. Threat knowledge can be gathered through threat intelligence sources. Threat intelligence sources can provide businesses information on the types of threats targeting the business and the tools, techniques, and procedures used by the threat actors to compromise the organization’s assets. Example of these threat intelligence sources might be public open sources, also referred as OSINT Open Source Intelligence, as well as nonpublic sources such as Information Sharing Assurance Center (ISACS). Oftentimes, the threat intelligence information might come from internal organizational sources such as reported instances of internal security incidents from Security Incident Report Teams (SIRTs), security events from monitoring-alert tools and last but not least fraud instances.

Threat intelligence is critical for the determination of the possible risk levels and also for the implementation of an effective risk strategy. By looking at threat intelligence, risk managers can proactively determine the level of alert and focus on monitoring specific events to trigger measures and actions and be prepared to respond to possible security incidents.

To proactively mitigate application threats, businesses can gather intelligence on threat agents and analyze threat events to determine whether the business is under the target of possible attacks. Threat intelligence helps business to build a threat intelligence knowledge base and derive threat libraries to analyze how threats can affect the application before the application is being attacked. After the threat and the threat agents are analyzed in detail, the next step consists of understanding the type of attacks and attack techniques that might be used by the threat agents and the vulnerabilities that are potentially targeted by these threats so that these can be prioritized for remediation. Often the answer to whether a digital asset is the target of

a threat agent is strictly dependent on the type of assets and the value they represent to a threat agent when stolen and used for malicious and fraudulent purposes. For example, if an application stores confidential data such as personally identifiable information of a customer, it might be the target of data identity thieves who would use this information to impersonate a victim (a threat usually referred to as identity theft). If the application stores bank and/or credit card records of its customers, it might be the target of fraudsters who could profit from using this type of data to make counterfeit credit cards and execute fraudulent payments online. If the application stores proprietary software, algorithms, and trading secrets, it might be the target of either country- or business-sponsored threat agents that seek to gain competitive and/or strategic advantage.

Risk Management Strategies

A risk management strategy of the company can be proactive or reactive depending on the culture and appetite for risk. An example of reactive risk management strategy consists of focusing on the security problem as it occurs and devote resources to manage the risk when the business is impacted such as in the occurrence of a security incident such as a data breach. When a security incident such as a data breach occurs, the company will often follow a security incident response procedure. After the security incident has been escalated for investigation and response, the next step consists of determining its severity and take measures to contain the business impact and the damage. Examples of measures that organizations can take to limit business impact and the damage include suspending the compromised credit cards, blocking transactions, and issuing new user credentials such as passwords. Once the damage of a security incident such as a data breach incident has been contained, the next step is to conduct a “postmortem analysis” to determine the possible causes of the incident and take measures to prevent similar incidents in the future. Actions that can be taken by businesses in response to security incidents include fixing the root cause of the incident such as fixing vulnerabilities and apply new countermeasures to reduce the impact to the business if similar security incidents will occur in the future. A proactive risk management strategy consists of adopting preventive risk mitigation measures, investment on risk management tools, people and process that reduce the likelihood of security incidents such as data compromise and impact in the future. Often proactive risk management decisions are taken because the organization has suffered a security incident that caused an impact to the business such as a massive breach of confidential data.

The initial assessment of risks such as the analysis of the probability for threats to occur and their impact levels are one of the objectives of risk analysis. One possible factor for determining the probability of a threat to occur is the availability of vulnerabilities that allow the realization of such threat, the complexity and reproducibility of the attack, the skills required by the threat agent to conduct the attack, and the costs of tools and resources to conduct the attack.

Examples of factors that might influence the threat impact are the exposure to threats by the presence of vulnerabilities in application and software, the value of the

asset (data classified as sensitive), the volume of such data, and the type of access controls necessary to get these data. For example, a threat can be facilitated by different factors, such as the availability of tools to the attacker and the knowledge required for using them. Often the tools used by the attackers are the same of the tools used by security testers to test a given asset for vulnerability such as an application vulnerability scanner. For example, consider a tool that is used by a tester to scan a given web application for identifying an instance of a common vulnerability such as cross-site scripting. Once the web application is found vulnerable to such vulnerability, it is flagged as issue in a report and this issue is remediated in compliance with the business information security policy. If the organization does not fix such vulnerability before the product/feature is deployed in production environments and released to customers, an attacker can use a similar vulnerability scanning tool to discover the vulnerability and then seek to exploit it for his advantage such as for compromising the web application and gather unauthorized access to sensitive customer data as well as web application functionality that is only available to authenticated and authorized users.

The secure testing of the applications and software, such as by conducting an ethical hacking and source code analysis are important risk mitigation activities and should be considered part of the application risk management process.

The Importance of Focus on the Emerging Threats

Identifying and fixing vulnerabilities before these can be exploited by possible attackers should be one of the sound security principles that businesses and organizations need to follow. Nevertheless, in today's threat landscape that is characterized by threat actors seeks not just to exploit known vulnerabilities but also nonpublicly disclosed vulnerabilities such as zero-days and design flaws that have been undetected by vulnerability assessment processes, the standard security testing to identify and fix common vulnerabilities is not enough.

Besides zero-days and design flaws, the other factor to consider by the businesses is the emergence of sophisticated and targeted threats relying on cybercrime ware tools. These are the tools that can be acquired by threat actors and are designed and configured to attack specific targets, for example, online banking with malware. Once the malware is configured to attack a specific target, it is delivered on the targeted client machine through phishing and social engineering as well as drive by download. Once the malware is installed on the target machine, it will be under the control of the attacker and will seek to compromise the customer sensitive data and the business transactions such as money transfers. Today cybercrime tools for attacking bank customers can be acquired and rented from cyber-criminals whose main activity consists of designing cybercrime tools for later selling or renting for profit.

Since cybercrime tools are developed in the economic underground with a sizable on tools and people that is reinvested from the proceeds of sales of these tools to the fraudsters and cyber-criminals who use them, there has been recently an increase of sophistication of the features of these tools such as use of malware clients and

command and control servers that use bot agents and web proxy for controlling the traffic of stolen data.

The sophistication of these cybercrime tools today is of big concerns for certain industry sectors such as banks and clients that are targeted but also other business sectors that deal with sensitive and intellectual propriety data. The increased sophistication of the cybercrime tools also goes hand in hand with the challenge for business to react proactively to these threats and deploy controls that mitigate the risk. Because of this lagging behind the threats, today's cyber-ware tools such as malware and bot-nets can bypass most of application and network security controls deployed by the business such as firewalls, intrusion detection systems and multifactor authentication. The effectiveness of cybercrime ware tools today is as such that the cybercrime tool herders offer a money-back guarantee in case the fraudster is not satisfied because missed the fraudulent proceeds from the use of this tools against the victims.

Threat Resilience and Application Security

Because of the increase sophistication of cyber threats such as malware, hacking, data breaches, online fraud, identity theft, money theft, exploits of zero-days, and denial of service attacks and others, there is a need today to engineer web applications and software that are designed, implemented, and tested to be resilient to these types of cyberthreat. For this to occur, it is important that the different application stakeholders in the organization such as the software development teams that include application architects and software developers collaborate with application security architects and penetration testers and provide the risk information such as reports about application threats, vulnerabilities, and countermeasures that risk and information security managers can use to make informed risk decisions and apply the risk mitigation strategy that is best suited based upon the risk management process of the organization. Fostering this collaboration among different users involved in securing web application assets requires the adoption of a risk management process that is beneficial to each of the user roles and responsibilities involved. Table 2.1 provides an example of application security stakeholders, their role and responsibilities, and the benefits such as the value added to application security/risk management that this role can play for the organization/business.

Threat Analysis

After the value of the assets has been analyzed, the main factors for the analysis of risks are to focus on the analysis of the threats and the attacks. At high level, the goal of threat analysis is to analyze the “who” that is the threat agent, the “what” that are the targets that reattacked such as the data assets and vulnerabilities and the “how” that is how the threats are realized in attacks and specifically the types of attack tools, attacking techniques, tactics, and procedures used by the threat agents.

Essential for the threat analysis is the knowledge of threats. For most organizations, businesses, and governments, the acquisition of threat knowledge relies on sources of threat such as threat intelligence. Threat intelligence means different things

TABLE 2.1 Application Security Roles, Responsibilities, and Benefits

Application Security Stakeholder	User Role and Responsibility	Benefits
Application architects, software developers	Design and implement (e.g. coding) new applications as well as change the design and coding of an existing application	Adherence to the application security standards and secure coding requirements
Application security architects and pen testers	Conduct secure architecture risk analysis and threat modeling assessments	Identify application vulnerabilities, determine technical risks, and recommend fixes or countermeasures.
Risk and information security managers	Manage risks to the business in compliance with governance and risk management processes	Make decisions on fixing vulnerabilities based upon risk and business criteria

to different types of organizations. For example, for government organizations, threat intelligence can be a matter that concerns intelligence agencies, national security, and the military. For businesses, such as private companies, threat intelligence gathering can be considered a service to which the company subscribes to learn about potential threat agents targeting their business assets. The threat intelligence information that is captured can help the business determine to whether these threats can target the company’s business assets and how they can be attacked. This type of threat intelligence information if properly disseminated to information security and risk management can be used for proactively manage the security of applications and the risk of threat agents targeting specific application assets before these threats are realized in attacks. Threat intelligence could come from different sources and can be gathered by different entities and disseminated through different channels.

Today there are several private threat intelligence services that provide this type of threat intelligence gathering for companies. This type of service varies from threat reports from public sources, such as publicly reported data breaches for similar businesses and organizations, to classified information, such as information gathered through secret services and law enforcement sources. From the risk awareness perspective, threat intelligence shifts the perception of risk, helps reduce the appetite for risk, and drives the risk mitigation strategy to take necessary preventive security measures to manage risks.

Analyzing the potential threat scenarios that target the digital assets is critical for the risk mitigation strategy, and particularly both “known” and “unknown” threats. A threat is considered known when it has already occurred either directly or indirectly (e.g. to a similar business and application), and it has been possible to analyze what the threat agents are, the attack tools and methodologies used, the type

of vulnerabilities exploited, and the impacts to the company assets. For example, an online banking application that allows customers to conduct financial transactions online might become the target for fraud and data compromise by banking malware threat agents. The analysis of the attacks used by the threat agents might include information about the type of the banking malware used as well as the type of actions taken to drop the malware on the targeted machine such as by using social engineering tactics including phishing campaigns and infecting vulnerable websites for drive-by-download infections. Once the malware is installed and has compromised the target, it will attack the assets such as the user credentials using a key logger and then attack the authenticated online session using a man-in-the-middle attack. With the control of the user session, the threat agent will gather customer's credit card and bank accounts and conduct fraudulent transactions, such as transferring money from the victim's account to an account under the attacker's control. The knowledge of this attack scenario is critical to determine the type of preventive security measures that need to be implemented to mitigate the risk, for example, antimalware software, device fingerprinting, multifactor authentication, and detective measures such as fraud detection and event monitoring.

The unknown threats are not yet fully documented and analyzed but are hypothetically possible. These types of threats are also referred as "emerging" threats. These are threats for which not enough information is gathered to determine the "who," the "what," and the "how." Examples of emerging threats include threats that do not have a history of associated events such as security incidents that can be used for learning about the threat actors, the targets and the tools, techniques, and tactics used. The drive of such emerging threats can be business or political events as well as new opportunities presented to the attackers such as the availability of new tools for conduct exploits, the availability of new channels such as social media, mobile channels, and the emergence of new technology that represents an opportunity to attack such as cloud computing, virtualized environments, mobile devices and so on.

From risk management perspective, it is important to be able to learn about emerging threats as well as known threats, identify the type of assets and vulnerabilities that are targeted, and how these are targeted, that is, the type of attacks and attack techniques used to determine the type of countermeasures that are effective in mitigating the risks.

The analysis of the different threats and attacks potentially affecting an application helps determine the likelihood of a threat scenario for the given application. Specifically, this determines how different threats can be realized by a threat agent using different attack techniques. In the case of malware attacking the application, for example, the threat agent might seek first to compromise the application interfaces with the user (e.g. the browser) and subsequently attack either the transmission media (e.g. the online channel) or the servers where the data is stored (e.g. database). Once the attack scenario has been analyzed and it has been determined that the application is at risk, the next step of the risk mitigation strategy is to determine if similar digital assets and businesses are also exposed and can be affected by either the same or similar type of attacks. To conduct this type of analysis, it is important to look more in

depth at the application architecture and specifically how the application processes the data and the type of layered controls available to secure the data.

Analyzing risk at the architectural level to determine how a threat can affect an application is one of the main steps of architectural risks analysis. The risk to an application asset is determined by the chances (likelihood) of a threat agent to be capable (skillful and resourceful) of exploiting vulnerabilities of the application and of causing an impact (fraud, theft) to an asset.

Analyzing the Attacks

The relevance of a threat for the application asset depends on the likelihood of the threat to occur and the potential negative impact. After it is found that a certain threat can be relevant to the asset, the next step is to determine the impact that this might pose to the assets. Since threats are realized through attacks, analyzing the different attack scenarios allows an analyst to describe the various ways threat agents can attack the application, analyze the attack vectors, and determine how these can exploit vulnerabilities to cause an impact to the application assets.

To analyze these attack scenarios from the perspective of risk mitigation, it is important to consider both the attacker and defender perspectives. From the defender perspective, the classification of data drives the scope for specific security requirements to protect the data. For example, any data that can be used to uniquely identify a customer, such as the customer demographic information that includes his name, address, and date of birth, is considered confidential by the data owners and it is mandatory that the confidentiality, integrity, and availability of such data are protected from unauthorized access and compromise. This is usually documented in the company information security policy as a security requirement and is validated by putting the application in the scope of security processes such as application security reviews and vulnerability assessments.

From the defensive perspective, the scope of which assets need to be protected depends on the intrinsic value of the asset and is different for different types of organizations. In the case of a financial organization, for example, data assets whose business transactions such as payments and money movements are usually considered confidential and business sensitive information and need to be protected from unauthorized access and compromise.

A company that produces Customer Off the Shelf (COTS) software most likely will consider this software as proprietary and in some cases a trading secret whose intellectual property needs to be protected. When the software is considered intellectual property, the main threats against it might include spying, reverse engineering, copying, and cloning. A company whose application allows customers to register and subscribe to news and receive alerts on new products and services can be considered low risk since the information that is handled and stored is mostly nonconfidential and nonproprietary.

As the data are classified according to their intrinsic value, the web application's inherent risk depends on the data classification that the application uses for business. The higher is the value of the data that is stored and handled by the application, the

higher is the risk for the application. Moreover, other factors need to be considered as well, such as the volume of confidential and sensitive data that is stored and handled as well as the type of service and transactions that are dependent on such data.

From the threat agent perspective, the higher is the gain resulting from the compromise of the target, the higher is the interest on pursuing the target attempting different attacks and vulnerability exploits. In a targeted attack, a threat agent will target specific exploits of vulnerabilities that allow him to reach his goal. Since vulnerability exploits are a cost for the attacker, the more common and easy is the vulnerability to exploit, the lower the cost is for the attacker and the higher are his chances to succeed in the attack.

From the risk management and application security perspective, the approach is much different than the threat agent attacker perspective: a defender ought to fix all possible vulnerabilities that can be exploited by an attacker. That means any potential weakness in any of the security controls of the application. For the defender perspective, it is therefore necessary to protect the application data from attack vectors targeting all the user entry points and analyze any malicious data that can be entered from these entry points to gain unauthorized access to the application. A threat analyst has to continuously shift from the attack perspective to the defensive perspective to identify which data flows can be attacked assuming that these entry points are in the control of an attacker. From the defensive perspective, the threat analyst ought to assume that these entry points and data flows are compromised and identify any potential vulnerability that can be exploited by an attacker.

From the defender perspective, protecting the application data means being able to protect the application confidentiality, integrity, and availability. From an application design perspective, this means building preventive controls in the application such as encryption, digital signatures, authentication and authorization, session management, input validation, secure fail over, and safe exception handling. Detective controls might consist of logging and monitoring security events as well as feeding the application parameters in secure incident and event monitoring controls.

This type of threat analysis often requires the analyst to think like a threat agent that is trying to break into the application defenses and exploiting any possible design flaws and vulnerabilities of the application. If the main goal of the attacker is getting the customer's confidential data, for example, he or she might exploit flaws in authorization, such as escalating privileges, breaking into the application through weak authentication, impersonating or reusing a session through weak session management, injecting commands into the application through weak input validation such as SQL injection to gain unauthorized access to the application data and functions. Other possible vulnerability exploits for this threat agent include the exploitation of poor or nonexistent cryptographic controls, malicious file execution vulnerabilities, injection flaws, remote file inclusion-upload vulnerabilities, weak user authentication, and lack of transaction authentication, and flaws in authorization and session management controls.

An important factor for analyzing attacks is that attacks can be either opportunistic or persistent. Opportunistic attacks take advantage of the application vulnerabilities already present in the application and exploit the ones that allow the threat agent

to affect the asset sought. Persistent attacks take advantage of the unlimited time for trying different attack vectors to break into the application and affect the assets. By persistently trying to attack an application with different attack vectors and techniques, eventually the threat agent will succeed in finding a new vulnerability that can be exploited to conduct the attack. Vulnerabilities that are identified for the first time and for which there is no remediation (e.g. patch) available are considered by security researchers zero-day vulnerabilities. When these are found, they represent a high risk for the application because there is no known mitigation being implemented, tested, and deployed, therefore, exposing the application to the potential attacks. For these zero-day vulnerabilities, it is also not entirely known which threat agents will use them since these were never exploited before.

Analyzing Vulnerabilities and Impacts

One of the standard definitions of risk is that a risk is realized by a threat agent seeking to exploit vulnerabilities with an attack to cause an impact to the targeted asset such as data and functionality. Within this definition of risk, a vulnerability represent a condition of realize the impact. When vulnerabilities are identified and fixed, the risk is mitigated. In the case of web applications and software, the identification of vulnerabilities is the main goal of vulnerability assessments such as pen testing and source code analysis. Analyzing the exposure of vulnerabilities to potential threats such as by using threat modeling is also essential to determine the risk severity of the vulnerabilities in presence of threats. For example, the threat model can provide the information to assess if vulnerabilities' exposure to a threat agent is either external or internal. If the exposure to the threat agent is external, this might represent a higher risk to the asset rather than when the exposure is only internal.

For penetration testers, using a risk scoring method to assign the severity of the risks of vulnerabilities such as CVSS allows to determine the risk level of the vulnerability. This can be done by considering factors such as exposure of the vulnerability and the confidentiality, integrity and availability of the data that is exposed by the vulnerability as well as other factors that depend on the threat environment and exposure of the vulnerability to the threat agents. For example, a high-risk level can be given to an SQL injection vulnerability that might be exploitable pre-authentication to access a database that stores authentication credentials where a low-risk severity level will be given to a vulnerability such as XSS that is only exploitable by injecting scripts via internal files, and therefore exposed only to internal threat agents.

Besides the exposure of vulnerabilities to a threat agent, an important factor to consider is determining the impact of vulnerabilities to the assets. The impact against an asset might be factored in terms of impact to the security properties of an asset such as the confidentiality, integrity, and availability of the asset in case it is compromised. A data asset that represents value to the business, such as customers' sensitive data like bank and credit card accounts and PII, might cause a monetary impact to the business if this data is compromised. In general, impacts to the assets can be both tangible and nontangible: examples of tangible impacts include monetary losses because of data loss and/or fraud, unlawful noncompliance, legal fees, and fines. Examples

of nontangible impacts to the business are reputation loss, impact to the brand and franchise, loss of trust in the company business, and any type of negative information such as news of the data breach that will impact the shareholders’ trust in the business.

In application risk analysis, quantifying risks by factoring the loss or compromise of an asset is the next step after the technical risks have been assessed. That is, determining the risk of the assets’ confidentiality, integrity, and availability being exposed by potential threats. For example, this is how a risk manager determines the possible impact to the business caused by a threat agent targeting the application by exploiting a system weakness and/or vulnerability. From the technical risk perspective, exploiting a technical vulnerability might cause system failure, the loss or compromise of the asset, and fraudulent use of the asset (fraudulent financial transaction). From the business impact perspective, exploiting the vulnerability might cause a monetary loss due to system failure, regulatory fines, and lawsuits because of data breaches, direct and indirect costs for recovering from data breach incidents, and monetary losses due to fraudulent transactions.

Table 2.2 shows some examples associating threats to applications with the technical impact (e.g. system failure, data compromise, vulnerability exploits) and with business impacts (e.g. monetary losses and liabilities).

From the perspective of analyzing the threats targeting a web application, we should analyze first the context that is the threat landscape that includes specific threats to web applications as targets to understand how these threats could

TABLE 2.2 Example of Threats and the Technical and Business Impacts

Threat	Technical Impact	Business Impact
Malware infected PC taking over online banking credentials	Loss of users’ authentication data allowing fraudsters to take over the account (impersonation)	Money loss due to fraudulent transactions by impersonating the logged user to move money to fraudulent accounts through third-party accounts (money mules)
External threat agent exploiting application’s SQL injection vulnerabilities	Unauthorized access to users’ data including confidential and PII, trading secrets, and intellectual property	Liabilities for loss of users’ PII, lawsuits for unlawful noncompliance, security incident recovery costs, and revenue loss
Denial of service attack against the application	Unavailability of web server due to exploit of application and network vulnerabilities and lack of redundancies to cope with traffic overloads	Revenue loss due to loss and/or disruption of service denying customer access to services and goods. Lawsuits from customers and businesses and recovery costs

compromise the application assets. Application level threats at a minimum should include threats to both the application data and the application functions. One main subject to consider in this type of analysis is the application assets and how they are managed by the application. If the data stored by the application includes customer confidential data, for example, it is a potential target. When customer confidential data is stolen, it could be used to impersonate the customer in a fraudulent transaction.

Since the risk strategy seeks to minimize the residual risk to an acceptable level for the business, mapping attacks to vulnerabilities is a critical step in application threat modeling since it allows a risk manager to determine whether the exposure of the vulnerability to an attack can lead to an exploit and increased risk.

Protecting the confidentiality, integrity, and availability of data assets is usually addressed as a security requirement and depends on the classification of the data. Regarding the determination of impact on data assets, an important factor to consider is the classification of the data. Information disclosure of customer's PII or business sensitive information (e.g. bank accounts, credit card accounts) to unauthorized users, for example, is more critical than exposure of data that is not deemed confidential and is just classified as internal. The localization of the application assets in the physical and logical architecture of the application whose classification is either sensitive or confidential is critical to determine the exposure of these assets to potential attacks and is one of the objectives of the architectural risk analysis.

Architectural risk analysis is a critical assessment for determining whether the application's security controls mitigate impacts, such as using encryption to protect data confidentiality in storage and in transit between the end user and the application. Part of the architectural risk analysis is also to determine the data entry points to the application. The entry points' risk can be ranked by considering the type of exposure it provides to the threat agent (e.g. internal, external) as well as the likelihood and potential impact to the data asset value (e.g. public, internal, confidential, and restricted) whose data the entry point leads to.

An important factor of the risk mitigation strategy is whether one of the measures recommended addresses process gaps, such as when failing to apply information security requirements leads to a potential risk to the business. For example, failing to protect confidential data can be considered as either a gap in the documentation of a security requirement or in the application of the security requirement. Similar examples include unenforced authentication and the need to know (e.g. minimum privileges for the users).

The enforcement and validation of these information security requirements are critical not only to mitigate threats and attacks but also to mitigate unlawful non-compliance and audit risks. When security measures to mitigate such attacks are not implemented as required by information security policies and standards, the business faces a noncompliance risk.

For example, the countermeasure designs need to mitigate potential attacks to satisfy at minimum, the security requirements in compliance with the organization's information security standards and policies. For this reason, the risks posed by design

flaws identified by application threat modeling need to consider noncompliance risks as well as business and technical risks.

To consider all possible impacts of exploitable vulnerabilities, we need to consider design flaws identified through threat modeling, security bugs identified through secure code reviews, and insecure configuration issues that can be identified with application penetration tests. Some of the design flaws that lead to attack exposure can only be identified by looking in depth at the application design that includes detailed examples of session management vulnerabilities. Session management vulnerabilities, for example, might include uncoordinated single logout between applications that share the same session when the user logs in. When the user logs out of one of the services used by the application, some sessions are left open for the user so that they can be still reused. Such vulnerabilities have a HIGH impact to the business since they might allow a session to be replayed so an attacker can gain access to the application. However, the severity of such vulnerabilities is MEDIUM in the presence of short-lived sessions and idle time outs.

Some application vulnerabilities that application threat modeling can identify include identifying weak authentication controls such as weak authentication. The attack agent targeting weak authentication might be a malicious user who installed malware to capture data in transit between the victim's browser and the application. Without compensating controls such as a channel to protect the confidentiality of the authentication data in transit, SSL, for example, the vulnerability in the authentication being used by the site might expose authentication credentials to an attacker. From the technical risk perspective, this could be a risk that can be classified as HIGH considering both likelihood of the exploit (e.g. BASE64 credentials can be decoded in transit such as in the case of HTTP basic authentication) and exposure (the application is external facing to the Internet). The business impact of this exploit can be rated HIGH since the exploit of the authentication weakness might allow unauthorized access to the application and the application data, which might include confidential data and PII. The impact for noncompliance audit risk is also HIGH because the disclosure of confidential PII data is usually a violation of the organization's information security policies. Other potential negative impacts from exploiting this vulnerability include reputational damage because of the requirement enforced by local data breach notification laws to publicly disclose a data breach of confidential PII, as well as potential fines and liabilities from regulators.

Estimating Risks and Impacts An information risk assessment consists of estimating the risk caused by a threat seeking to compromise the confidentiality, integrity, and availability of the data in the presence of vulnerabilities. The estimation of the level of risk is usually done using qualitative and quantitative risk calculation methods. Qualitative risk analysis factors the likelihood of the exploit of vulnerabilities and the impact of the confidentiality integrity and availability of the asset. For a given threat, the higher is the sensitivity of the data and the higher is the exposure to the data caused by a vulnerability, the higher is the information security risk. In quantitative risk analysis, the risk is quantified in terms of monetary losses that the business might incur in case of either a loss of the asset (e.g. data) or loss of business service that

process that asset (e.g. business transaction with the data). To determine the business impact caused by the loss and compromise of an asset, is it important to first estimate the value of the asset and calculate the likelihood of such asset being compromised due to the exposure of the asset to a vulnerability by a given threat agent.

Determining the value of assets depends mostly on the value given by the business in the context of the service that application provides using that asset. An application that generates revenue for the company by selling goods online, for example, might suffer a revenue loss when that application service is not available to the customer because of a denial of service attack or other type of application vulnerability exploit.

Different types of threats expose different types of information, such as PII or intellectual property, and that information loss may cause negative business impact. For example, when a threat agent exploits a technical vulnerability, such as a design flaw, coding bug, or misconfiguration, it causes a negative impact to the application assets.

It is important to make a distinction between information security and business impact: An information security risk is the risk that affects the security properties of the data such as the confidentiality, integrity, and availability of the data. A business impact is the economic impact caused by either the loss or the compromise of the value of the data asset.

Sometimes the difference between information security risks and business impact is subtle: information security risks are not per se associated with business impact until asset values and negative impacts to the business due to the loss or damage of the asset are also considered.

From the technical risk perspective, mitigating information security risks is a risk management decision, while from the business risk perspective, mitigating the risk of a business impact is a business decision. The essential steps of a risk mitigation strategy need to consider both information security and business risk factors, which can be done by following a two-step process:

1. Perform the technical risk analysis that focuses on the determination of technical vulnerabilities and the analysis of their impact on the assets. Categorizing the vulnerability or technical risk framework is critical to determine the impact from a technical risk perspective. Since a technical risk is in essence a weakness in a security control, it can simply be described as weakness in protecting the confidentiality, integrity, and availability of the data. Such weakness can be identified as vulnerabilities in application controls, such as authentication, authorization, encryption, session management, data validation, error and exception handling, and auditing and logging.
2. Correlate the technical vulnerability with the asset value to determine in monetary impact such as the cost to the business when the data compromised or lost.

For an application risk mitigation strategy to be useful to the business, it is important to monetize the loss that may result from an exploited vulnerability. A risk assessment typically includes the evaluation of information security risks and the technical

risks posed by exploit of vulnerabilities and the business impacts. By factoring both information security and business impact (e.g. monetary impact caused by the loss of data), risk managers can make informed risk decisions on how to best manage the information security risks and business impacts. For example, the business can choose either to accept the risks since they are of low or no business impact, or to fix the vulnerability and implement a countermeasure for the threat to reduce the potential impact to a level that is acceptable to the business.

The exposure of an asset to information security risks is also a factor of the severity of the vulnerability found in a security control whose function is to protect the asset. The severity of the vulnerability as well as the impact on the asset confidentiality, integrity, and availability can be used to determine the overall risk of the vulnerability.

A known vulnerability, such as failing to encrypt a password in transit, can be ranked HIGH because of both likelihood to exploit (the password is passed in clear text and can be eavesdropped with a web proxy), as well as impact (hijacking a password can lead to application access). From the business risk analysis perspective, the risk of disclosing customer confidential PII (such as a data loss incident) is generally considered a HIGH risk, both in terms of unlawful noncompliance and negative impact to the business due to the public release of the data loss incident.

The assigned risk levels to the severity of the vulnerability can be used by information security managers to prioritize the remediation effort. From the technical risk mitigation perspective, vulnerabilities can be prioritized for remediation based on their risk level (e.g. HIGH, MEDIUM, and LOW). High-risk vulnerabilities are prioritized for mitigation over MEDIUM and LOW risk vulnerabilities since these represent a higher probability of being exploited by a threat agent, which might lead to the higher impact to the asset. From the business risk mitigation perspective, the vulnerabilities whose exploits lead to the compromise of the more business valuable assets are the ones that should be prioritized first. When a risk mitigation measure is applied to the current efforts of protecting the asset, the risk that is left over is referred as residual risk. One of the main objectives of risk mitigation strategy is to reduce this residual risk to a level that is acceptable to the business. Determining whether the risk posed by vulnerabilities is acceptable will consider risk mitigation factors such as enforcement of security requirements, security by design and by configuration, and system and infrastructure security measures.

A risk assessment can be actionable for managing business impacts caused by threats by determining how security measures need to be designed, implemented, and configured to reduce the exposure of the assets to vulnerability exploits sought by that threat. Since security measures can be applied to mitigate the risk and vulnerabilities should be remediated to reduce exposure to threats, determining which vulnerabilities need to be fixed ultimately depends on their potential impact to the business caused by a threat potential exploiting them. In essence, a risk mitigation strategy that considers both technical and business impacts need to factor the risk caused by vulnerabilities and the business impacts caused on the assets.

From the perspective of managing risk by implementing technical controls, it is important first to understand the application technical functionality and discussing it with the business owners. The main objective of this business discovery stage is to

capture essential information about the business functionality of the application and the basic business functions. Some of the application functions might be common across different applications, such as user credential registration, user authentication, and query data. Other application functions might be business specific, such as ordering a product or service, paying online bills, wiring money to a beneficiary account, trading stocks, or betting for on an online auction.

Once the business context is understood, the next step is to understand the technical context of the application. This might require gathering information about the application design (architecture documents) and the various components of the technical architecture, such as the client-browser, web and application servers, databases, and backend mainframes, including the protected assets that store data such as customer sensitive, bank, and credit card information.

Once the data assets of the application are identified and depicted in the application architecture, to the risk analyst will analyze each application's business function and how the application security controls (authentication, authorization, encryption etc.) provide the necessary security features to protect the data assets, specifically the confidentiality, integrity, and availability of the data.

To determine the business impact, it is important to consider the business context in which the application operates. This can be determined just by looking at high level application assets and the application functionality that the application provides based upon these assets.

The final stage consists of determining the business impacts to the business caused by threat agents targeting the assets. For this type of risk analysis, it is important to have a risk framework of threats, vulnerabilities and controls and to follow a methodology to evaluate the impact to the application assets of various potential threats targeting the application. This can be done by following a risk-based threat modeling methodology step by step including the execution of a predetermined set of activities with expected outcomes such as artifacts.

From the perspective of managing application security risks, there are several types of risk frameworks that can be used for analyzing risks and threats to applications, some of which will be covered in this book, simply for comparative purposes. This book also includes a new risk-based threat modeling methodology whose details are initially introduced in Chapter 6. In this chapter, we will cover more breadth rather than depth what the important areas that a threat analyst needs to cover for a preliminary risk analysis of an application using threat modeling techniques. In order to conduct a thorough analysis of the threats affecting an application, it is important to commit to specific goals and follow a set of predefined activities step by step.

This type of risk assessment is in essence the assessment of the risk posed by application technical issues. The main scope of this technical risk assessment is to analyze how these design flaws and gaps might cause a security impact to the data assets, such as causing a compromise of the confidentiality, integrity, and availability of the data asset. Nevertheless, to determine the technical risks, it is important to qualify the likelihood and impact of the vulnerability. This type of technical risk analysis needs to consider other factors besides likelihood and impact, such as the exposure to potential

threat agents and attack vectors to these assets. It is therefore critical to consider different threat scenarios and determine the different types of attack vectors that threat agents can use against the assets and components of the application architecture and determine the impact when the application vulnerabilities are exploited. Essentially, technical risk analysis consists of assessing the risks posed by threats to the assets and attack vectors seeking to exploit vulnerabilities of the application. The application vulnerabilities that are in the scope of the threat analysis, that is, the analysis of the exposure to threats to assets and attack vectors, can be discovered by other application vulnerability assessments, such as secure design review, secure code reviews, and ethical hacking/pen testing.

In any case, it is important to consider a threat model as a dynamic risk assessment that needs to be revisited over time to consider new threats to the application. As any other risk assessment methodology, threat modeling cannot be considered a one-time application security process but rather an iterative process to keep updates the threat model with new threats targeting the application as well as with the new exposure of these threats due to changes introduced to the application.

An event that should trigger a revisit of an application threat model is the emergence of new threats potentially directed toward the same class of web application and/or software products used by the company. The objective of updating the threat model in this case is to consider if these new threats need to be mitigated by adopting new countermeasures. In today's application threat landscape, for example, web applications and application software might become the target of new attacks and new threat agents seeking to attack the application and the business that runs it for different reasons, such as monetary and financial gain, political gain, spying-intelligence, or terrorism. By updating the threat model with these emerging threats, it is possible to identify the adequacy of the security measures in place and plan for the implementation of new preventive and detective measures as needed to mitigate the risk.

Applying Security Measures

To determine the security controls that need to be designed and implemented to mitigate the risks, it is necessary to consider the potential vulnerabilities that can be exploited by the threat agent and the potential impacts of these on the assets for each specific threat agent and attack vector. Once the main threats, threat agents, attacks, and vulnerabilities are identified, the next step is to determine the risk to the assets in terms of likelihood and impact. By using an application risk framework, it is possible to systematically consider different types of threats and attack scenarios and to analyze the likelihood of the application assets to be impacted as well as the business impact. Only by understanding how the different threats might cause a technical and business impact, it is possible to determine which security controls in the application can effectively mitigate the risks to the business posed by these threats. A technical impact can be described in terms of impact to the security controls, such as compromised authentication, authorization, and encryption controls. A business impact can be described in terms of tangible and intangible losses to the business.

Identifying technical and business risks in a given application is a critical piece of information for deciding which countermeasures should be implemented to mitigate these risks. For example, the type of risk information that might help risk managers make informed risk decisions includes the type of threats targeting the application, the control gaps and vulnerabilities, and the countermeasures to mitigate such gaps and vulnerabilities.

Once the application risks have been analyzed and the necessary security measures have been identified, the next step is to communicate to the application development teams how these security measures need to be designed and implemented. The security measures to mitigate technical risks of the application are included in the technical risk assessment that is done after threats are analyzed, application vulnerabilities identified, and technical impact to the assets have been determined, such as the impact to confidentiality, availability, and integrity of the data assets.

When technical risks posed by vulnerability exploits have been identified in a threat model, the next step is to factor threats in the determination of the risk/severity of these vulnerabilities. Since threats, attacks, and vulnerabilities are correlated in a threat model, it is possible to determine risk severity by looking at their exposure to the attacks visualized in the attack tree. Since an attack tree provides the information about the step-by-step attack scenarios, it is possible to determine the exposure of the assets to these attacks and factor the exposure in determining the risk probability. Determining the factors of risk to assign a severity level to the vulnerability, such as probability and impact, allows the business to decide which vulnerabilities should be prioritized for risk mitigation and determine the risk strategy: higher risk vulnerabilities remediated before medium risk vulnerabilities and medium risk vulnerabilities before low-risk ones.

For a risk mitigation strategy to be consistent, it has to be the first and foremost based upon the objective evaluation of risks. Objective evaluation means that risks are identified and managed as a result of following a standard process rather than best practices and improvised ad hoc processes. The results' consistency depends on following standard process, tools, and technology used and training and awareness. Among these basic elements, probably the most important factor is the training and the experience of the personnel involved in executing the risk strategy.

Specifically in analyzing threats to the application, the threat analyst's experience in identifying threats and vulnerabilities is one of the most critical factors and should not be overlooked. Since threat modeling is not an assessment that can be automated by using threat modeling tools alone and is very dependent on human knowledge, the training of the threat/risk analyst who conducts the threat/risk analysis is the most critical factor contributing to the overall quality of a threat model. Even if using risk/threat modeling tools can help in the execution of the assessment, these are certainly not necessary for a good quality threat/risk model of the application. Ultimately a threat analyst with several years of experience in application security and technical risk assessment is probably the most important asset in conducting this type of analysis. Nevertheless, what is really important for a successful application risk strategy is providing actionable tactical results that can be used by the development teams to

implement the necessary countermeasures and reduce the impact to the business to acceptable levels.

IMPROVING APPLICATION SECURITY

“When solving problems, dig at the roots instead of just hacking the leaves”

Anthony J. D’ Angelo

We previously explored how business can more effectively manage the risk caused by cyberthreat agents by gathering threat intelligence on the threat actors and targets and by analyzing the type of attacks used against the targets including the type of vulnerabilities that could be exploited. The rationale of this analysis of threats and attacks is to identify countermeasures that are effective in reducing the impact to the application assets. Moving forward, we look at best practices and risk management strategies that organizations can business and follow to improve application security.

Documenting Application Security Requirements

A good place to start for improving application security is security requirements. Some of these security requirements can be derived by understanding the potential threats that might affect the application. For example, modeling how a threat agent can abuse an application by assuming that the application’s use cases can be abused to cause harm to the application and the data, helps derive a set of security risk mitigation requirements.

The model of threats affecting the application and the various components of the application architecture also helps determine application assets’ risk exposure to potential attack vectors. Security issues identified by architecture risk analysis and vulnerabilities identified with secure code reviews and penetration tests also help to improve the security of applications and software throughout the SDLC. By adding threat modeling activities during secure architecture reviews, secure code reviews, and penetration testing, it is possible to determine the likelihood and impact, both technical and business, of potential threats exploiting these vulnerabilities and determine the necessary measures to mitigate the risks.

Applications that are built for the first time or existing applications that are undergoing changes for introducing new features planned for future releases of the application and software represent an opportunity to improve security by applying new security requirements to either the application as a whole or just the changes that are introduced.

Parallel to documenting business requirements (also covered in functional requirement specifications), deriving security requirements leads to improved security by building it into the new functionality. Oftentimes, this effort is limited to documenting security requirements for protecting access to the application assets from unauthorized users, granting users the entitlements to read, create, and modify the application’s

data, protect the confidential and authentication data in storage and transit, validate the application inputs and outputs, manage the user session securely, and audit and log security events as well as requirements for the secure configuration and operation of the application. Threat modeling can help improve security during this stage by deriving security requirements for security controls as risk mitigation features. For example, threat modeling determines the strength of the authentication required to mitigate potential threats against assets when the value and the risks for these assets are high. More specifically, when the analysis includes the possibility of different attack vectors attacking the authentication control with brute force attacks, credentials enumeration, data injection, session hijacking by exploiting potential vulnerabilities such as not locking the account login, credentials harvesting through login error messages, SQL injection, session fixation helps derive and document a set of technical security requirements that are more control specific and help in the design of a more secure application and software.

Security requirements that are derived using a threat model and validated with security tests also provide the business a certain level of assurance that the application's security has been tested to mitigate the risks from known threats and attacks. If, for example, security requirements are derived using use and misuse cases, validating these security requirements provides a reasonable assurance to the business that the mitigation of the risks posed by these threats has been considered.

The drivers deriving application security requirements include applicable compliance risks with privacy laws, regulations, and standards. Typically, compliance with security requirements can be assessed using checklists that qualify which security controls should be in scope and when and how they should be implemented. In the case of financial applications, for example, the design of multifactor authentication needs to comply with the FFIEC guidance for implementing multifactor authentication for authenticating high-risk transactions.

In the case of a design flaw such as authentication that is too weak to comply with a regulatory requirement, application threat modeling can point architects to a set of recommendations to fix the design flaw, including the type of recommended countermeasures. Such countermeasures might include previously analyzed and approved multifactor authentication (MFA) controls, such as knowledge-based authentication, risk-based authentication, one-time passwords, soft and hardware tokens, challenge/response questions/answers, biometrics.

At the architectural level, secure architecture requirements can be derived from the documented application security standards and guidelines. These standards and guidelines provide development teams direction on how to design and implement secure application and software by covering all the critical aspects of secure application design, such as protecting the authentication data in transit and storage, session management, error logging and auditing, role base access controls, and configuration management.

An application threat model performed specifically as a secure architecture risk analysis also needs to validate the design's compliance with the security requirements set by the secure architecture guidelines. This validation can be done in a variety of

ways that include the analysis of the architecture design diagrams and data flows as well as a threats and vulnerability assessment using checklists.

The depth and breadth required for the secure architecture analysis might vary depending on the overall objectives. A security requirement can be as generic as validating that simple requirements, such as external facing web applications that handle customer PII, are compliant with the privacy requirements stated in the privacy questionnaire.

A more in-depth security requirement to validate might require reviewing the design to validate that customer PII is masked when displayed in each documented user interface and is protected in storage, transit, and all data flow between the users and the application.

Application Security “By Design”

When an application security expert reviews the application design, one of the important aspects to consider is whether there are risks in the design of the application architecture, such as security design flaws, and to determine whether any assets (e.g. data) are exposed to these design flaws. When a design flaw that exposes the application asset to a certain risk and attack vector is identified, the security expert can determine the likelihood and the impact to the business and proactively mitigate this risk before the application software is implemented and the application/product is rolled out in production. Besides proactively mitigating risks, fixing application vulnerabilities during design and ahead of coding produces a return of investment to the business since it is much cheaper to fix these during design change when the application has already been implemented and tested. In summary, when security activities such as security requirement derivation, architectural risk analysis, secure code reviews, and penetration testes are integral part of the SDLC, we have the opportunity to also integrate threat modeling and analyze the likelihood and impact of potential security issues identified by these security activities.

Applications and software that are already developed and released into production environments can also be improved from the application security perspective. Application threat modeling helps improve the security of existing applications by putting them in scope of a risk assessment review to validate the application security against a set of security measures that are necessary to mitigate the risks posed by new threats targeting the application assets. When performing the assessment for the first time, this type of application threat model constitutes a baseline and can be leveraged to analyze new threats and vulnerabilities. These vulnerabilities can be introduced by design changes to the application and software during the product life cycle. Such an application threat model baseline can be stored in a repository and used throughout the product life cycle. The baseline threat model of the application can be updated as needed, such as when either the application is being changed or when the application risks need to be reassessed to consider emerging threats.

In terms of standard design documentation, threat modeling can also use standard modeling artifacts such as use cases, architecture diagrams, and sequence diagrams. As in the case of software development teams that follow formal design

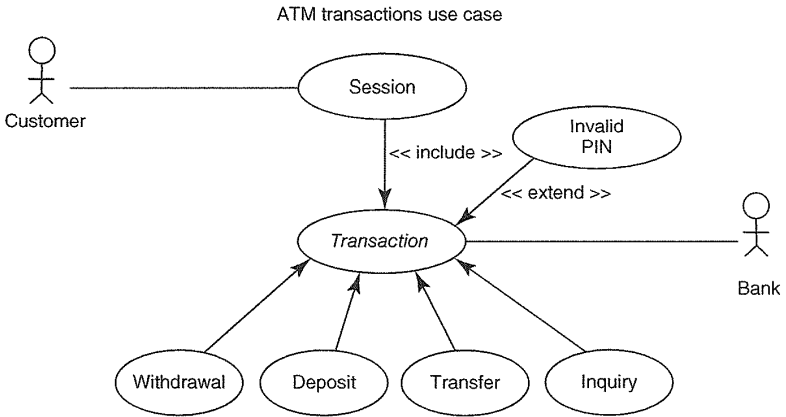


Figure 2.1 Example of Use Case Diagram 1

methodologies such as Unified Modeling Language (UML) artifacts such as use cases can be used to depict at high level how customers interact with the application functions of an application. An example of such use case diagram for ATM transactions is shown in Figure 2.1.

Other examples of design artifacts that can be leveraged for threat modeling include sequence diagrams that show how data flows sequentially through the different elements of the application architecture, architectural design diagrams that show the different tiers of the architecture and the components and servers, as well as network-physical architecture diagrams. When these design artifacts are documented by following a template, they can be very useful for the security design reviews and threat modeling since they capture essential information to identify potential security issues, such as design flaws or missing security requirements.

One important value of threat modeling conducted during the architectural risk analysis is that potential vulnerabilities, such as design flaws, can be correlated with vulnerabilities found with other assessments in later phases of the SDLC such as coding, verification, and testing. One example of this correlation consists of identifying whether design flaws found earlier in the SDLC with threat modeling result in reduced security flaws found later (e.g. during coding) with manual secure code review. This type of vulnerability data correlation helps to identify and address root causes earlier in the SDLC with improved process efficiency and security defect management cost reduction.

When applying threat modeling to architectural risk analysis, it is important to adopt a methodology that is used consistently across the organization so that threat modeling assessments can be compared across applications that are designed and implemented by development units. Oftentimes though, application threat modeling is conducted as an informal assessment, especially in the unfortunate cases when the application architecture is not documented and the security analyst SME can

only assess the potential risks and impacts to the application by capturing them on a white board during an application development team meeting. This might be a useful exercise and effective in recommending improvements to the security of the applications and software in scope, but the effectiveness might be limited to the knowledge of the application brought to the table during the meeting as well as the experience of the application security SME in conducting this type of analysis.

When a formal threat modeling methodology is followed, the assessment can be conducted consistently by following the methodology step by step. Typically, the first step is to gather the necessary documentation to conduct the analysis. Once the initial design documentation of the application is gathered, it is possible to estimate the time necessary to conduct the assessment and plan the application threat modeling activities. For example, this is the time that is required for the review of the documentation such as business requirements, the review and the analysis of the design documents, architecture diagrams, use cases, sequence diagrams, and so on. Depending on how big the application is, this requires days or weeks to read the documents and set up meetings with the application stakeholders necessary to gather information and understand the application functionality and architecture more in depth.

As good approach toward the goal of improving the security of existing applications is to proceed gradually and plan a gradual, phased, and controlled adoption of application threat modeling among the different application development teams. Adopting a formalized, standard application threat modeling process might occur over several months or even years in large software development organizations. Companies that follow a standardized threat modeling process across the different teams within the organization are more likely to produce consistent and reliable results than companies that do not.

For increasing the changes of adoption of application threat modeling within an organization, it is important to leverage existing software engineering processes, such as security architecture design reviews. During these reviews, the security architect will review the architecture design documents and analyze the physical and logical architecture diagrams to identify the main components of the architecture, such as web and application servers and databases and to identify the data assets and the interactions with the data flows. Once the various assets and components of the application architecture have been identified, the next step is to enumerate possible threats to each component to determine how they can be affected. This step in the application threat modeling is often referred as architecture decomposition and involves decomposing the architecture into basic architectural elements such as assets, data flows and components (web server, application server, and database server), and considering how a predetermined list of threats such as STRIDE might affect these architectural elements. The outcome of this architecture security review might lead to identifying either errors or gaps in the design of security controls that are required to protect the application asset from potential threats. A change in design to address the security concerns has a substantial impact on the security of the application. For example, after reviewing the architecture of an existing application, the analyst finds that the communication channel between the application server and database storing authentication data is not encrypted to protect the data from information disclosure and

spoofing. Additionally, there is no mutual authentication that prevents repudiation of the connection request to the database from an untrusted client. The information sent is also not digitally signed to prevent tampering outside the transmission channel. The security architect might request to roll out these changes to the application prior to the next release to mitigate these risks.

A more comprehensive application threat modeling process might also include a preliminary risk analysis of the application, the threat agents, the threat libraries used to identify likelihood and impacts to the assets, attack tree analysis of the different channels and assets that can be attacked, correlation of threats to existing vulnerabilities identified in the application, determination of technical and business risk, determination of security measures and prioritization of these based on a risk strategy whose objective is to maximize protection by minimizing cost to the business.

For a threat modeling process to be executed consistently, a standard application threat modeling methodology needs, at a minimum, to define the steps, the required inputs, and the expected outputs (e.g. artifacts) that need to be produced during each step. This will include, for example, architecture design documents that describe at a high level the physical and logical architecture and the design of the security controls. Conducting such a comprehensive application threat model can be very time-consuming and resource-intensive, especially when it is performed on new applications for the first time. On the other hand, the bulk of the effort is only required for the first threat model since it will be used as a baseline and just requires updating when new changes to the application are introduced, such as when integrating the application with new application components or services, as well as when changing the existing application data and components. Nevertheless, adopting a comprehensive application threat modeling assessment can be considered the ultimate goal for an organization and an application security investment that will lead to savings in vulnerability defect management efforts.

From the defender perspective, protecting the application assets is usually not just the task of one measure and one control but a set of measures and controls applied at the different layers of the application architecture. Applying security design principles such as defense in depth, security by simplicity of the mechanism, security as the security of the weakest link and security as security by default configuration, for example, is among the best practices that most application security practitioners can follow. The same security design principles can be applied to assess the security of the architecture during the secure architecture design reviews and threat modeling, considering a set of possible threats to each layer of the application architecture and assuming that when one layer is compromised, another layer would need to mitigate the risks and the potential impacts. In analogy to a defense in-depth evaluation of security controls, for example, a threat analyst needs to evaluate the potential exposure to threats by adopting an attack in-depth approach to identify how attacks can be mitigated at the different layers of the application architecture and to determine how application layered controls can work together to provide the best defense in depth for the application assets.

From the perspective of an attacker, trying to gain access to the application data, for example, if an attack vector causes one security control to fail, such as by exploiting a

vulnerability of the client browser, the risk to the data asset protected by the application can still be mitigated by another control at the server layer, such as requiring the user to authenticate outside with out of band authentication. This type of threat analysis considers different threats to the different security controls and measures adopted by the application to protect the data assets.

Information Security Reviews

From the information security team's perspective, making sure that the company data assets are protected is one of the main information security requirements. For applications, software, and data assets, information security policies can be validated by asserting the security of the design against these requirements. Examples of such security by design assertions include validating the presence of security controls using a checklist and testing the application for vulnerabilities that need to be remediated to satisfy the information security requirements.

A security issue identified using the checklist requires further validation and can be assessed further during a meeting of the security analyst, application architect, and product teams. When the issue is identified, it can be addressed by adding a missing security requirement and/or a design change. When this type of validation of security requirements is applied to the application architecture reviews, it is meant to validate a negative requirement because of not having a security measure to comply with the organization's information security policy. If the checkmarks confirm that the security measure is in place, no further action is required. This type of assessment, even if useful as initial information gathering about the security of the application is not as useful as to take actions to mitigate risks when such risks are identified as a negative answer in the security checklist.

The scope of the checklist might include validating specific information security requirements, such as authentication and authorization access controls, password policies, user entitlements, protection of sensitive data, and auditing and logging. Such validation of security requirements unfortunately does not always translate to actionable security for the application. For example, validating the security of an application by conducting an interview with the application stakeholders might identify a gap but might not determine the level or risk when this gap is exposed to a specific threat.

With an information security assessment of the design of an application, the issue is identified as violation of information security policy and the fix consists of implementing a security control according to the specific requirements for security controls to protect the confidentiality, integrity, and availability of the data as these are documented in the information security requirements. The limitation of this type of assessments is that the findings of secure design review do not translate into actionable recommendation for mitigating the risk of the exploit of the security control gap by a specific threat. In an application threat modeling assessment, fixing a design issue in the application design comprise more specifically implementing a countermeasure that goes beyond fixing the vulnerability that exposing the asset. Since security issues in a threat model are typically identified by analyzing threats, trust boundaries, and

data flow diagrams and are typically actionable by development teams that have an understanding of the architecture design of the application.

When security issues are identified using a threat model that considers the impact of threats and vulnerabilities to application assets, it would take the assessment of the application against information security requirements a step further. Beyond the basic compliance requirements and audit risks, a threat model covers the risks posed by threats targeting the application assets. By scoping the application design for application threat-model review, information security team could analyze the exposure of the application to threats and determine the countermeasures that mitigate the exposure to these threats.

From the risk management perspective, the strategic objective of threat modeling is to minimize risks and the associated business impacts. One of the risk management benefits of application threat modeling is helping information security professionals to manage vulnerability risks by prioritizing the remediation of these vulnerabilities according to their severity. Information security requirements can be set to manage the risk of vulnerabilities found with threat modeling during design ahead of releasing the application into the production environment. An example of information security requirement for critical vulnerabilities is that all security critical design flaws identified with threat modeling would require a design change before allowing the project to move to the implementation-coding phase and a security test to validate that such design flaws are fixed before releasing the application into production.

After completing a threat model, information security teams can look at the security flaws identified during threat modeling and, using architecture design reviews, validate that the application's design changed to remediate the security design flaws such as through the introduction of new countermeasures. During the validation phase, the effectiveness of such countermeasures to protect the application from threats and attack vectors identified during threat modeling is tested by specific security test cases. When the countermeasures successfully pass the security tests, the business can be assured that the countermeasures work as expected to mitigate threats and attacks identified during the threat modeling exercise.

Vulnerability Management

Threat models are very helpful in assessing the risk and exposure of vulnerabilities to the application assets. Another important application security improvement consist of the fact that security issues identified and fixed prior to the implementation and pen tests could reduce the number of vulnerabilities that need to be fixed at later stage of the SDLC such as during the validation phase of the SDLC. This will result in cost savings for the defect management costs associated with vulnerability management. From the defect management costs perspective, for example, using application threat modeling to identify security issues during design is more cost-effective than identifying and fixing them during coding or validation-test SDLC phases. Security issues that have roots in design can be fixed during design and reduce the overall security defect management costs. According to a study from IBM System Sciences Institute, the cost of fixing a design flaw during design is 7 times cheaper than fixing with

coding and 100 times cheaper than fixing it during production with a new application patch. According to the same study, threat modeling is also the application security assessment activity that provides the highest Return of Security Investment (ROSI) in application security.

Typically, vulnerabilities with high severity risks take precedence for remediation compared to medium risk while low risk vulnerabilities can be remediated last. After the vulnerabilities have been prioritized for remediation based on their severity, the next step in risk mitigation is to determine the cost for fixing them. This depends on the type of vulnerability and the type of corrective action plan that needs to be implemented. The cost of fixing the vulnerability might differ depending on whether such vulnerability requires a design, code, or configuration change. In essence, the cost depends on the root cause of the vulnerability. For example, some vulnerability fixes might require a change in configuration of the application. These changes are cheaper to implement and roll out than changing the source code and/or redesigning the application.

By classifying each type of vulnerability by its origin and cause, it is possible to determine the necessary fixes. The origin of vulnerabilities is the type of information that tells us that phase of the SDLC life cycle; the security issue is most likely being introduced.

For example, a cross-site scripting (XSS) vulnerability most likely originates during the coding phase and is mostly caused by either not implementing output encoding or coding insecurely and missing the implementation of that requirement. *This type of information is actionable for managing vulnerability risks because it can be used by software developers to fix vulnerabilities in source code during the implementation phase.* An example of actionable information that leads developers toward remediating an XSS vulnerability includes encoding the outputs so that malicious JavaScript will not be executed on the browser.

In the case of SQL injection vulnerabilities, for example, the actionable recommendation for development teams consists of fixing the vulnerability with a source code change and construct SQL queries using prepared statements instead of concatenated strings. The classification of the vulnerability type helps development teams to develop the vulnerability fix. For example, vulnerabilities that are categorized by the type of security control that is affected (e.g. authentication, authorization, data protection, session management, data validation, auditing and logging, error and exception handling) direct the software development team to fix the security control in the application that is impacted by the vulnerability.

For organizations whose software is either developed internally by engineering teams or externally by vendors and only integrated and tested internally, applying threat modeling to the overall application as well as the components that are either integrated or developed by third parties/vendors provides several risk mitigation benefits. Firstly, for applications developed internally, applying threat modeling allows organizations to identify any potential design flaws and remediate them during the SDLC without introducing any risk in the production environment. Secondly, for vulnerabilities either found on external third-party components or exposed to external services and delivery channels, a threat model can provide an assessment of the

technical risks during design and require the third party to address these risks before integration with the application. This is, for example, the case of an online banking application that relies on both in-house and third-party financial transaction processing systems and that can be accessed via different service delivery channels such as web, IVRs, and mobile. Because of the interconnected nature of such applications and systems, a threat to one of these application interfaces or channels can exploit vulnerabilities in any of these integrated and interconnected systems and services.

In the case of enterprise software where several complex systems and services are integrated, it is very important to define the scope of the application threat modeling to make sure that all critical components are covered. In the case of an e-commerce application, an online retailer might integrate the functionality to the online catalog-warehouse, shopping cart, and shipping applications. A complete application threat model would need to cover not only threats inherent to the application, but also threats to external components and services that the e-commerce application depends on, such as the credit card, payment services, and shipping services.

Application Risk Assessments

From the risk mitigation perspective, application threat modeling helps reassess the application against new threats and risks. As threats evolve, the exposure to these threats needs to be reassessed. For doing so, it is necessary to revisit the application threat model whenever application risks need to be reassessed or whenever new changes are introduced in the application that might pose new risks. From the business perspective, the main concern is first to understand the potential business impacts, and second, to determine if potential business impacts justify spending to introduce a change in the application to implement a countermeasure. A threat model of the application can help the business make such informed risk decisions to mitigate the risks and, for example, to decide that when the risk exposure is low, the risk can be accepted and that when risk level is high, the vulnerability needs to be fixed to mitigate the risk.

From risk mitigation perspective, a sound risk management practice is to update the application threat model every six months and each time the application design changes. One main reason for updating the threat model is that application design changes might expose the application assets to new threats. By updating the threat model, it will be possible to determine if application changes expose the application assets to new potential risks that require designing and implementing new countermeasures. Independent of changing the application, if the threat landscape for the application has changed and new threat agents and/or attacks need to be considered because they were used to exploit similar applications, for example, the existing threat model might need to be revisited.

From the security governance perspective, it is therefore important to consider updating the threat model with a frequency that depends on the initial risk profile of the application. The specific frequency that the application threat model needs to be updated depends on different risk factors such as risk associated with the classification

TABLE 2.3 Criteria for Threat Modeling Scope

Scope for Application Threat Modeling
Internet facing applications
Business critical function/services (online banking, brokerage-trading)
High-risk transactions (e.g. money transfers)
Access and handling of confidential data
Access and handling of authentication data
Access and handling of business sensitive data (e.g. credit card accounts, bank accounts)
Access and handling of PII
User administration functions

of the asset that needs to be protected by the application and the criticality of the services provided by the application. An example of criteria to decide whether an application should be in scope for Application Threat Modeling is shown in Table 2.3.

By using the criteria for threat modeling scope shown in Table 2.3, applications that are Internet facing and rated mission critical might be required to have their threat model be revisited semiannually. The same process might also require a review of the security of the application whenever a certain type of application change is introduced, such as a change to the classification of the assets and a change of security controls that protect the data.

Examples of application changes that might require an update of the application threat model include any application changes that might impact the overall security of the application, such as any changes to the application architecture and components, changes to the data and the information classification, and changes to security controls that protect the confidentiality, integrity and availability of the data. An example of criteria that can be adopted to determine if application changes should be put in scope for application threat modeling is included in Table 2.4.

Besides time-based and application changes criteria, another event that should be a reason to reevaluate application threat modeling is to reevaluate the application's exposure to new threats following a proactive risk mitigation approach, or reactively in case a security incident causes a data breach or fraud whose causes are still unknown.

BUILDING SECURITY IN THE SOFTWARE DEVELOPMENT LIFE CYCLE

“The most critical applications are generally developed internally. Thus, companies should focus on training their developers in secure programming and establishing a secure development process.”

Gary McGraw, Chief Technology Officer, Cigital on PCI reference in Compliance Means getting your App Together, Darkreading security

TABLE 2.4 Criteria for Application Threat Modeling Updates

Scope of Changes that Warrants an Application Threat Modeling Update
Application design to introduce new functionality, processes
Application components, libraries
User interfaces
Data
Data classification
Network appliances, hosts, servers, backend data bases
Network configuration, topology
Integration with third-party components
Design of authentication
Design of role-based access controls
Design of encryption to protect data in transit and/or storage
Introduction of new data filtering, ingress and egress
Communication channels and data flows
Session management controls
Error and exception handling
Auditing and logging

A larger percentage of software used by businesses and organizations today is internally developed in support of business critical functions and processes. In the financial industry, for example, most of the online banking applications, including online trading applications, are developed by internal teams. Typically, internally developed software is subjected to several types of testing before release, such as integration testing and quality assurance testing. The internal software development has also to adhere to the coding standards and the various software engineering activities that are part of the SDLC used by the software development teams within the company. Internally developed applications are also required to be secure and prevent exposing vulnerabilities to potential threats, especially when operating in environments at risk such as the public Internet. For applications whose functionality is exposed through the public Internet, this requirement consists of testing the application for potential vulnerabilities before it is deployed into production. This type of vulnerability test consists of a web application penetration test performed during the validation phase of the SDLC. Typically, when new vulnerabilities that expose the application data and/or business transactions to high risks are identified, they need to be remediated prior to releasing the application into production. Securing applications by testing for vulnerabilities after releasing the application in production is not very efficient for several reasons: the engineering teams need to put off engineering resources to fix vulnerabilities late in the life cycle, eventually impacting the deadlines for the production release of the application. Another reason is cost. It is very expensive to fix vulnerabilities later in the SDLC since sometimes design and coding changes have to be reintroduced to fix the vulnerability and require redesign, recoding, and reintegrating and rebuilding the application so it can be retested to validate that the vulnerability has been remediated. A more efficient way to secure

software and applications is to identify and fix security issues earlier in the SDLC such as during requirements, design, and coding. This can be done by adding information security review activities to software engineering activities such as security requirement reviews, secure architecture reviews, and secure code reviews. Another important aspect to consider is securing software by default to engineer secure software by requirements, design, and coding, a process also referred as software security engineering.

Rationale for Integrating Threat Modeling into the SDLC

There are several good reasons to integrate threat modeling activities within the SDLC. Threat models help information security reviews of software projects and assess the risks posed by vulnerabilities identified during the various type security activities that are built into the SDLC such as design reviews, secure coding, and penetration tests. A threat model help risk managers to assess application security risks by correlating the analysis of threats to the vulnerabilities identified in various assessments and these vulnerabilities to application data and functional assets. A risk analysis that factors threats, vulnerabilities, and assets can also factor likelihoods and impacts to determine the risk level and to apply security measures to reduce this level of risk. A threat model of the application also helps software engineers to derive security requirements such as by adopting use and abuse case analysis as part of the formal application design. As secure by design activity, threat modeling helps identify potential design flaws in the architecture that might expose the assets to potential threats and to document requirements in the architecture design documents. A threat model can also be used to derive secure coding standards such as by considering the threats and the vulnerabilities in the source code that these threats might exploit.

For the information security organization to adopt threat modeling in the SDLC, there must be a strong rationale since both development teams and business sponsors will be impacted by execution of threat modeling and it is important to communicate the value to all the application stakeholders.

For any security initiative to be successful within an organization, it is important to consider both strategic and tactical goals of threat modeling. One important strategic long-term goal is, for example, identifying and remediating vulnerabilities earlier in the SDLC and possibly reducing the number of vulnerabilities caused by weak security requirements and design flaws that can be identified using threat modeling during the secure architecture reviews. Other important strategic goals of introducing threat modeling during requirements and design phases of the SDLC are actionable recommendations for fixing these security issues, assurance that requirements for threat-risk mitigations are documented and validated during design reviews, and secure coding and tests are used to sign off the application prior to release. From business perspective, an important strategic goal is to introduce savings in defect management costs during validation and tests since there will be fewer expected vulnerabilities identified at this stage of the SDLC.

From tactical security perspective, integrating threat modeling as activity in the SDLC needs to consider where and how it should be integrated. This might depend

on several factors, such as the type of SDLC the development team uses (e.g. Agile, Waterfall, RUP), in which phase of the SDLC threat modeling needs to be integrated, and how it should be integrated as part of other secure software engineering activities such as security requirement engineering, secure design reviews, and secure testing and validations. Another important aspect to consider is the integration of these security engineering activities within the project workflows to establish which software changes need to trigger revisiting or issuing a threat model, which technical and business documentations are required to conduct the threat modeling activity, and the expected outcome of the assessment (e.g. threat profile with threats, attacks and vulnerabilities, risk profile),

The main prerequisite for the integration of the threat modeling activity in the software development process is the alignment of software engineering and software security objectives, essentially this means taking into consideration both security and engineering team goals. For organizations that did not yet adopted a process to integrate security in the SDLC also referred as S-SDLC process, integrating threat modeling as part of information security reviews and secure software engineering activities implies a lot of work to do up front to make it happen.

Typically the integration of security within existing SDLC activities, such as documenting functional requirements, designing application architecture, coding, integrating components, and testing quality, requires following security requirements in the SDLC methodology and engineering and security teams cooperating to make it happen. Even before choosing a standard for the S-SDLC to implement, this being MS SDL, Cigital-Touchpoints, or OWASP-CLASP, it is important to assess how mature and capable the organization is in critical areas such as secure software engineering domains. This can be done by assessing the capabilities and maturity of the secure software engineering practices using a standard maturity-capability models yard-stick, such as Cigital-BSIMM or OWASP-SAMM (Software Assurance Maturity Model), that helps to compare these practices against other organizations in the same industry sectors.

Depending on the maturity and capability of the software development organization, the size of the software development organization, and the resources allocated in terms of hiring and training the security engineering workforce, software security process, technologies and tools, the implementation of an S-SDLC process and the integration of threat modeling with the S-SDLC activities might require years. In essence, this is a task that requires following a “Software in the SDLC Strategy” as well as planning and cooperation from security and engineering teams within the same organizations.

Assuming that a strategy for adding threat modeling as part of S-SDLC process, roll out threat modeling training, and develop and/or acquire threat modeling tools has been approved by budget decision makers (executive management) of both the information security and the engineering organizations, the next step is to plan a roadmap and execution for the strategy. The successful execution of an S-SDLC strategy depends on the business commitment to push it within the information security and the engineering teams. Typically, the main obstacle to overcome at the beginning are the common security requirement misconceptions, such as security engineering

and security reviews can be thought of later in the SDLC, added security controls to mitigate risk impact, application performance and usability. Adding threat modeling during requirements and architecture design reviews requires extra time and delays approvals in the project workflows. Security teams and development teams must therefore work together toward a common goal while understanding each other's agenda and limitations. Since responsibility of the application's security usually lies not with engineering teams but with security teams, it is important to work together as team and share the effort in terms of resources, processes, training, and planning to make software security a viable proposition for the company.

To plan the adoption of threat modeling with the organization, it is important first to understand the organization's capabilities and maturity in secure application development processes; understanding which application security processes have been followed by the organization and how well they are followed. The big distinction is between organizations that have adopted a "security built in" approach toward software development versus the one that has not.

Software development organizations that have just started adopting application security as one of their programs will probably execute a "bolt on" security on process and apply security to the application after development, such as by testing it for security toward the end of the SDLC and making changes to fix security issues just before release into production. Software development organizations whose application security capabilities have matured over time probably already adopted a "security built in" process and strive to execute several application security activities in different phases of the SDLC, such as architecture risk analysis during design, source code analysis/secure code reviews during coding, and security testing during validation. The implications of integrating threat modeling in these cases really depend on which security engineering model, bolt on or built in, the software development organization uses.

Applications that are developed following a security bolt on model are usually designed and implemented without necessarily performing any preliminary risk analysis or threat model. If an information security policy is enforced with a requirement to test an application for vulnerabilities prior to release, the only concern for security is to make sure that the application security requirements for testing are satisfied by putting the application in scope for a vulnerability assessment. As vulnerabilities can be introduced during design as well as coding of the application, security strongly relies on the vulnerability assessments.

Adherence with Information Security Processes Applications' compliance against information security policy depends on different factors, such as how the information security policies are enforced by the different information security processes rolled out by the organization, the information security review processes and tools used, and the amount of human resources dedicated to these assessments. Organizations that review the security of the data assets, including application assets developed internally, might already follow standard application and information security review processes for compliance and governance. A security in the SDLC process can first leverage information security review processes that already align with the

SDLC workflow activities and add additional security activities to be performed when security teams are engaged to review and assess the security of applications and application changes. For example, organizations that already have rolled out information security reviews during the design phase of the project can leverage this checkpoint to integrate also with security design review assessments, such as architectural risk analysis and application threat modeling. A security checkpoint to test the application for vulnerabilities prior to release in production can have a checkpoint for assigning the risk of the available threat model/risk profile.

In essence, before we start considering adhering threat modeling to the SDLC, we must consider adhering information security review processes to the SDLC. One of the evaluation criteria is to determine how well information security review processes allow security teams to validate compliance of development projects with information security requirements. The assessment of the project scope against information security requirements can also be leveraged to add additional security requirements that are derived by a threat modeling activity such as use and misuse cases. For example, by considering the potential threats, risk likelihood, and impacts of these threats to the application assets due to changes and new features introduced to the application, it would be possible to validate if new security requirements need to be documented in the new project to mitigate these risks. Later on during the design phase, modeling threats to the application is a critical activity to determine if there is any risk of these threats being realized by exploiting design flaws in the application architecture. A model of potential threats, the vulnerabilities that these threats might exploit, and the security controls that mitigate their risks can be used to derive a set of security test cases, such as use and abuse test cases or a battery of attack library tests.

Security test cases that are derived can later be validated via security tests to assess the security of the application against possible threats to the application and application assets. For example, information security policies define the security requirements to protect the confidentiality, integrity, and availability of company data based on the type of data classification (e.g. confidential, PII, restricted, sensitive, internal). The security requirement to protect the assets that are dictated by the information security policy applies also to the application security, as these can be considered an asset since the application handles the data by storing it for a user's later consumption. Based on the data classification, for example, the decision whether to encrypt the data is not made based upon threat modeling assessments, but upon documenting security requirements based on the application threat modeling and use and misuse cases and enforcing them through security governance and application security assessments, such as security design reviews/architectural risk analysis, manual secure code reviews, static source code analysis, manual security tests, and vulnerability assessments.

For a start, to determine the threats to the application and the business, we need to include a step that characterizes the business context in which the application functions, the type of data stored by the application, and the type of processing the application applies to the data. If the application business requirements for the application are already documented, the next step is to derive the security requirements for

the application. Application security requirements can be derived from the information security policies and the applicable technology standards depending on the type of the application and the environment in which the application is meant to operate. Deriving security requirements can only be done after capturing business and technical requirements for the application. The engineering team can follow to design and build an application with security controls that are effective in mitigating potential threats based on these security requirements. Using threat modeling to derive security requirements is often referred to as security requirement engineering by security practitioners.

Application security requirements can be documented in different forms: “positive” or “negative” requirements. Positive requirements highlight the expected outcome of the requirement as described in the expected functionality, the requirement could state that the user is required to be authenticated to the application; the authentication will validate the user’s credentials, username, and password. As a positive outcome, it allows the user to log into the application and access the application functions. In essence, a positive security requirement describes how a security control is expected to provide a security function, such as authentication and authorization.

Negative security requirements describe how the security control should react to unintended use of the control, such as when a malicious user tries to abuse the control functionality. An example of a negative requirement for a security control is to include a description of how the control would need to function to mitigate a threat. For example, the authentication will lock an account after more than six failed attempts to log in to prevent a malicious user from using brute force on the authentication.

In any application, a use case describes at high level the application functionality from the user perspective and helps derive the application’s functional requirements. A use case describes either visually or textually who are actors using the application (e.g. users or external systems), the user’s goals (e.g. query data), and the steps required to achieve such goals (e.g. log into the application, input data, process query, retrieve data).

In simple terms, to create a use case, the user of the application is considered the actor and the application function is considered the interaction of the actor with the application. The result of the user’s action is also described in the use case as the actor’s goal.

Typically, applications that are designed by following formal design methodologies such as the UML include use cases as artifacts for deriving the application functional requirements. Application use cases also describe the application’s functionality from the security perspective as these requirements prescribe the security controls of the application, such as user authentication, authorization, encryption, data filtering, user session management, and audit and logging.

A prerequisite of the application threat analysis is therefore being able to capture technical and business data about the application. At a minimum, the threat analyst needs to be able to look at the application documentation, such as business and functional requirement documents as well as design documents. Without the application functional requirements, use cases help describe the main functional characteristics of the application and to describe how the user interacts with the application and the

application data. From the business perspective, it is important that a threat analysts first and foremost understands the business context in which the application operates, such as the data assets that the application ought to protect, the different user roles of the application, the type of user interfaces, the type of data that is stored and transmitted, the type of functions supported, and the type of data processes supported (e.g. query data, send data, use service). It is therefore important to understand the business context of the application specifically from the perspective of the application value as the application functionality is considered an asset for the business as well as for the consumer of the application.

Considering the application as well as the monetary value associated to that data as an asset is a critical factor for the threat analysis, specifically for determining the business impact to the organization in case the data asset is either compromised or lost.

Besides considering the data as an asset, it is also important to consider the application itself as an asset for the type of service and functionality that is provided to the customer by using the application. When considering the application functions as an asset, from the defender perspective, it is important to determine which application functions represent the higher risk of being targeted by an attacker. For example, in a financial type of application such as online banking, any movement of money can be considered a high-risk transaction that needs to be protected from an attacker with additional authentication such as multifactor authentication. In any given application, functions that require higher privileges than user privileges such as administrator privileges to create and delete users of an application can also be considered of higher risk than normal user functions.

Since an attacker will seek to compromise the application data by abusing the application functionality with different types of attack vectors, a threat analysis that simulates these types of attacks is extremely useful to determine whether the application business logic can be abused. This type of threat analysis consists of analyzing how the application can be abused by an attacker by deriving the use and abuse cases of the application.

The goal of an abuse case is to describe the possible application abuses, such as the various malicious actions that are undertaken by a threat actor to try to steal confidential data from an application.

An abuse case describes the application abuses visually or textually through a sequence of steps that a malicious actor will follow to reach his goals. Since this type of analysis is also referred as use and misuse cases, it is important to qualify the difference between misuse and abuse. A misuse case describes the misuse of the application controls by incorrect use of the application, such as by error or an act of user negligence. An abuse case is meant to describe the application abuse, for example, when a malicious actor abuses the application with malicious intent for stealing data, steal money, and/or disrupt a service. For the sake of the terminology used in this book, we will refer to either misuse case or abuse case interchangeably since the same type of methodology will be used.

A use and misuse case represents the first artifact of threat analysis and can be used to derive nonfunctional requirements. Nonfunctional requirements help engineering teams build applications that are more resilient to malicious attacks since they are

designed to withstand abuse of normal functionality. In software requirement engineering, use and abuse cases describe the various steps that a threat agent will follow to reach his goals. These use and abuse cases are translated in negative requirements, such as “the application will prevent brute forcing attempts of the password by locking the user account after several failed attempts.” These negative requirements can be later formulated into testable functional security requirements, such as “the user account will lock after more than six failed log in attempts.”

Adherence to Software Engineering Process One critical value of modeling threats to application security is not only the capability to derive security requirements, but also to design applications whose security controls are “risk mitigation effective,” meaning implemented by following the principle of security by design.

Following a security by design principle means that security controls are designed by a default engineering process that keeps in consideration the threats to these security controls and the risks posed to the assets that are secured by these controls. From the risk mitigation perspective, in order to assess whether these application security controls are effective to mitigate threats, it is important to conduct first and foremost a security design review of the application architecture and analyze the security of different components, such as clients and the servers located in the different tiers of the architecture including web servers, application servers, databases, and backend servers. This type of analysis is also referred by security practitioners (e.g. security consultants and risk managers) as architectural risk analysis and consists of discovering vulnerabilities in the application architecture that might represent a risk to the application and the business.

Integrating security standards during the requirement phase of the SDLC is, therefore, an important proactive risk mitigation step that security teams should take for newly developed applications and for changes to applications. The security requirements might consider the security aspects of different domains, such as the application functionality, compliance with legal, regulatory and privacy laws, security compliance, and last but not least mitigation of threats against the application assets. Deriving application security requirements based upon input from these domains is not an easy task since it requires expertise to understand all these different domains and the impacts on the application.

An initial check on whether security requirements are implemented into design can be done through a security design review of the application whose objective is to review the application architecture and validate that security requirements are captured and followed in the design. In essence, the main objective here is to identify the design flaws. Once these design flaws are identified, it is important to understand the risk exposure to the application and the application assets: this can be done by analyzing which threats affect the application asset and how they can exploit these design flaws to cause a business impact, such data compromise, function compromise, and monetary loss.

As software security matures and is practiced within an organization, it is more efficient to develop secure software by building security into the SDLC. This is usually accomplished by integrating different security assessment activities in each phase

of the SDLC. In the case of secure coding, this means performing static source code analysis during coding. In threat modeling, this means not only performing architecture risk assessments during the design phase of the SDLC, but also using threat modeling to derive security requirements and use and abuse cases that can be later used to test the security of the application.

One of the main values of integrating threat modeling as part of the SDLC, besides improving process efficiency and reducing the costs of fixing vulnerabilities, is proactive risk mitigation. For example, as the threat scenarios evolve, the security requirements need to evolve too to control the risk posed by these evolving threats. Application software that is developed today can only be secure if it is being built based upon security requirements that are derived based on today's threat and attack scenarios as well as newly discovered vulnerabilities.

In essence, this means that the application security requirements need to be derived using a new threat model specifically derived for each application users, data, environment, and business functionality. By using application threat modeling to derive these security requirements, security teams can understand how the application can be attacked (by adopting the adversary perspective) and determine which assets may be targeted by the attackers to make sure that the security measures that are implemented can be effective in mitigating these attacks.

A threat modeling exercise conducted during the imitation phase of the SDLC, such as to derive the security requirements of an application, also represents a great opportunity for knowledge transfer from security teams to business teams. For example, when the results of the threat model are communicated to the business analysis in the form of security requirements, it can be an opportunity to discuss the potential risks that the application might be exposed to and to proactively mitigate risks instead of waiting for vulnerabilities to be identified when it is too late or very costly to be remediated.

The value of integrating threat modeling as part of the SDLC in terms of proactive risk mitigation can be one of the main selling points for adopting threat modeling within a given software development organization. Nevertheless, before any organizations starts to roll out threat modeling as part of the SDLC, they need to consider the potential impact to existing processes, people, and technologies and plan the amount of resources needed accordingly. Since it might not be feasible to apply threat modeling to each software product being either a web application or software service produced by the company because of the limited resources in place, it is worth it to apply a risk scope criteria to consider in scope for the assessment only the applications that are considered high risk for the organization and the customers. As the execution of the application threat modeling process matures within the organization, it might be allowed to extend to cover all applications and software products being developed. The ultimate goal is to execute threat modeling as a security assessment and security engineering activity that is consistently performed by different development and security teams within the organization using common threat modeling methodology, processes, and tools.

Adopting a Security in the SDLC Process

Integrating threat modeling with SDLC activities during requirements and design is a win-win for security, business, and engineering. To realize this, application, business, and development teams need to work closely together to create a culture of cooperation based upon knowledge transfer and understanding each other's constraints and perspectives. For development teams, security is often seen as a feature of the product whose requirements need to be met, often at the expense of usability in order to release the application in production. In essence, it is seen as a tax on the release of the application. A threat modeling exercise can add value to engineering teams only if it helps them to build software with less vulnerabilities, a minimum impact on project resources to fix these vulnerabilities, and without causing any delays on the production release.

In the case of engaging with consultants performing application threat modeling, initial considerations about the benefits of application threat modeling are very important and allow the sponsor of the application threat modeling initiative to get the most out of the engagement. In particular, it is important to capture these in the form of contractual requirements documenting these in the consulting agreements and Statements of Work (SOW) between the consulting company and the organization committing the application threat modeling engagement.

In the absence of a formal threat modeling process adopted by the organization, the reliance on the vendor's threat modeling process is often done on the case-by-case basis to satisfy a tactical need such as to perform a threat analysis of a highly critical application asset for the organization. This threat modeling exercise might have specific goals and requirements such as to determine the impacts of potential threats to the application architecture at a high level. The requirements for this threat modeling exercise might be documented "ad hoc" for the specific risk profile of the application in scope for the assessment. At high level, the main requirements for the scope and objectives of the threat model might be documented after a meeting between the application security team and the application development stakeholders. Since an "ad hoc" threat modeling exercise not yet follows a standard process, it usually not repeatable and is only as good as the knowledge and skills of the threat modeling analysts performing the assessment including the quality of existing technical design documentation and knowledge that is brought on by the application development teams (e.g. architects and software developers) during the design walk-throughs.

Such a threat modeling assessment can be very valuable to the organization since several critical security issues can be identified just based on the expertise of the application security SME and his capability to gather the necessary information to conduct the analysis. Even if conducted informally, such threat model exercise can be very useful prior to and during the design of the application architecture as a preliminary risk assessment. For example, when the architecture of the application is sketched at high level, it is possible translate the security requirements in the high level design of security controls to design the type of authentication required for users and for each server, the need of encryption of the data in transmission and storage, and other

security controls including authorization, data validation, session management, error handling, and auditing and logging.

In summary, for providing value to an organization, an application threat model should at least fulfill a few minimum requirements:

- Clearly define the scope and the objectives.

- Provide general documentation of the threat modeling methodology and process including SDLC checkpoints, assessment scope, application changes event triggers, prerequisites, and deliverables.

- Provide a threat profile of the application that allows business to make informed decisions based on information about threats, threat agents, attacks, vulnerabilities, assets, impacts, countermeasures, and residual risks.

- Be actionable for fixing issues and vulnerabilities by identifying origins and root causes.

- Provide a technical and business risk assessment of impact on assets based on technical and business impacts.

- Provide a set of different options for security measures that can be selected by the business for fixing vulnerabilities based on risk prioritizing and measuring effectiveness to mitigate risks.

If application threat modeling is one of the application security assessments performed by security teams or SMEs within the organization, it is important to document the scope of the engagement and the specific requirements for engaging the security team with the threat modeling service.

The goals of application threat modeling will also vary greatly depending on the type of organization. Organizations that produce commercial off-the-shelf software (COTS), for example, might use threat modeling to minimize both security and privacy risks for the client using the software as well as to reduce the costs of fixing security defects. Several ISV (Independent Software Vendors) have adopted threat modeling as part of the SDLC and significantly reduced the number of security bulletins to address vulnerabilities in newly delivered software.

In the case of organizations producing their own software and integrating third-party software in support of business services to customers (non-COTS products and non-ISV companies), the main objective of threat modeling is to reduce application security risks to acceptable minimum levels for the business by protecting both the business and the customers. Examples include organizations that use and develop their own enterprise-wide software such as financial organizations, government organizations, health care, telecommunications, production, logistics, online retailers, and sales-marketing companies.

After a vulnerability assessment of the application is complete and vulnerabilities have been identified, the next step is to decide which ones need to be remediated. One possible approach is to rate these vulnerabilities according to the likelihood of being exploited and the potential impact on the application assets. Assigning likelihood and impact to a vulnerability allows a security expert to determine the risk severity and

prioritize it for remediation based on a qualitative risk scale such as high, medium, and low.

The business perspective of security should be considered when new security features need to be implemented to mitigate vulnerabilities identified by an application security assessment. For the business team, security is not the usually the primary concern unless security impacts customer's usability of the application, the application's performance, and the ability to perform business functions and transactions. The main concerns, rather, include implementing new features and products that can generate revenue for the organization, time to market the new product/application/software, and commitment to the deadlines for the product release. From the business perspective, a change to the application due to a security requirement of the costs and time that needs to be spent fixing application vulnerabilities is a detriment to the business since it might delay time to market of the product, introduce security features at the expense of product usability, and incur additional costs for allocating resources to the design, development, and tests of security features and application changes to fix security issues.

Therefore, before an application threat modeling initiative is promoted to the organization, it is important to consider the business perspective and be prepared for push backs with sales pitches. One of the pitches for selling application threat modeling to the business, for example, is cost reduction by saving time and freeing up resources dedicated to fixing vulnerabilities identified with other assessments such as penetration tests. As a matter of fact, when security issues are identified and fixed during design, it requires less effort for the development teams to fix vulnerabilities during validation since a smaller number of vulnerabilities will be left to be fixed later during the validation phase, prior to delivering the application/product to customers. As the security assessments are moved toward the end of the SDLC such as by analyzing threats during the requirement phase to derive security requirements and by using threat models to identify security design flaws, it would be less costly for the business to build security into the application and products. By emphasizing the cost reductions in fixing vulnerabilities and the efficiencies that can be obtained by rolling out application threat modeling as a proactive application security assessment, it is easier for the promoters of application threat modeling to make their cases to the business stakeholders.

IDENTIFYING APPLICATION VULNERABILITIES AND DESIGN FLAWS

"Measure what is measurable, and make measurable what is not so."

Galileo Galilei

One important risk mitigation strategy for any software development organization is to identify and remediate potential vulnerabilities prior to releasing the application to the customers. Typically identifying potential vulnerabilities in software and applications involves the execution of both manual and automated security tests.

Automated security tests consist of running vulnerability scanners to scan the application as well as the source code. After security issues are identified with a scanning tool, they need to be validated manually to determine possible false positives. After security issues are validated, the next step is to assign a risk severity value to the vulnerability. A standard risk scoring method such as CVSS can be used for assigning a risk score. By assigning a risk severity to each of the identified vulnerabilities, it is possible to prioritize them for remediation and mitigate any potential risks of these vulnerabilities being exploited to attack the application and the application assets.

Effectiveness of Automated Security Testing

From the security team perspective, applications are considered secure and ready for release commercially in production when the security has been tested and vulnerabilities have been remediated. Nevertheless, when these security tests consist only of automated security tests, to the security team might have a false sense of security since other types of vulnerabilities might not be found yet with manual tests because of the limited vulnerability coverage of security tools.

The type of security issues that can be identified with automated testing, such as Static Application Security Tools (SAST) and Dynamic Application Security Tools (DAST), are called Low Hanging Fruits (LHF). The security testing using an SAST tool is considered static since it does not require for the application to run, but just to compile source code to conduct the taint analysis and data flow analysis. The security testing using a DAST tool can identify issues in the application and is dynamic since needs the application to execute in order to conduct the assessment.

An LHF type of vulnerability is a common macroscopic vulnerability and therefore within the reach of identification by an automated security tool. For example, a tool can test and identify a cross-site scripting type of vulnerability without any inside knowledge of how the application is designed and/or coded. Note that “not requiring particular knowledge of the application” to identify the issue does not mean the following:

- Some of the instances of the vulnerabilities are false negatives (not an issue) and require a security tester to validate them with his knowledge of the application.
- Several other instances of the same type of XSS vulnerabilities are missed by the tool, also referred as false positives (did not find that instance/issue).
- Several other instances of the same type of XSS vulnerabilities can only be found with a human-based and comprehensive testing, considering the combination of all possible attack vectors, some of which are not part of the tool arsenal, as an attacker will do in weeks of trial and error analysis.

Fixing the XSS vulnerabilities will just require following the general recommendations of the tool without understanding the application architecture and the security options available to mitigate the risk of the vulnerability (e.g. output encoding, input filtering at different layers of the architecture).

For these reasons, it is important to realize that security testing applications and software, by relying solely on automated security tools, are not guaranteed secure. Even with the state-of-the-art security tools available today, such as automated testing, static and dynamic security application testing tools, it is only possible to get instances of potential security issues. These security issues can only be considered vulnerabilities when they are manually validated. Manually validating issues reported by tools is aimed at identifying false positives. Security tools might also miss identifying and reporting issues, leaving the vulnerability in the application. These are called false negatives. The main reason for false negatives is security tool vulnerability coverage. The tool's vulnerability coverage consists of the ratio between the number of different types of vulnerabilities that can be identified by a tool and the overall possible type of vulnerabilities that can be identified in an application. Previous studies have shown that the vulnerability coverage of all security tools combined, static and dynamic, is less than 50%. Because of the limited vulnerability coverage of security tools, fixing all vulnerabilities identified with tools alone does not provide a guarantee of security. Therefore, considering an application and/or software secure because no issues are identified by a security tool is wrong. A more comprehensive test of security vulnerabilities in an application needs to augment automated static and dynamic testing with manual code reviews and manual pen testing. These security testing techniques are shown in Figure 2.2.

Manual penetration tests augment automated vulnerability scanning by identifying other instances of vulnerabilities in the application through conducting a more thorough vulnerability assessment of the application. These manual tests test for common vulnerabilities by following a security-testing guide such as the OWASP security-testing guide. Manual code reviews augment automated source code analysis through a manual review of the source code line by line for possible vulnerabilities.

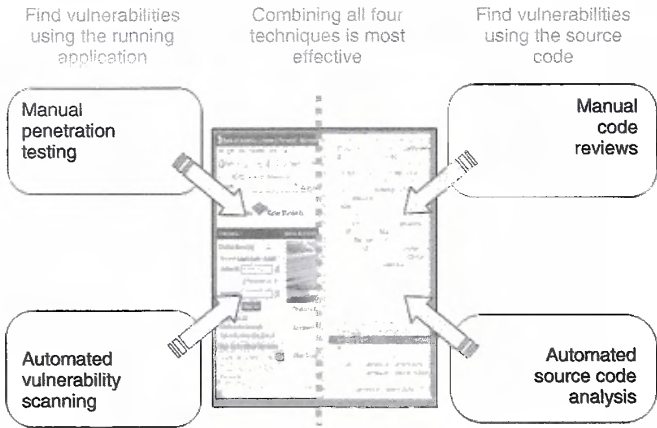


Figure 2.2 Manual and Automated Vulnerability Assessments

In essence, a vulnerability assessment, even when relying on automation/tools, requires human knowledge to assess the validity of the vulnerability findings. A comprehensive application security assessment needs to include manual and automated security testing as well to consider the exposure of these vulnerabilities to different types of attack vectors to determine probability and impact of these vulnerabilities.

Manual penetration testing and code reviews help identify a class of vulnerabilities that is usually not covered by security tools. These are omissions of security controls, flaws in design, and logical coding errors. Logical coding errors can be identified by a source code analyst with previous knowledge of the business logic of the application. Design flaws can be identified by a security architect armed with knowledge of the application security requirements.

Examples of design flaws that can be identified with manual source code reviews might include the following:

- Unencrypted passwords in storage such as when establishing a connection between application server and database.
- Unencrypted confidential data in storage and transit.
- Missing server side validation for potential malicious input.
- Unenforced authorization/role base access controls according to security policies and user access control matrix.
- Insecure exception handling in the source code.
- Not logging security events.
- Not implementing session logout and time outs.

To comprehensively test applications and software for these types of vulnerabilities, automated security tests must be augmented with manual tests, such as manual penetration testing, manual source code analysis, and manual security reviews of the application's design (architectural risk analysis and threat modeling).

Specifically, in the case of security design flaws, it is important to look at the possible causes in the design at the architectural application level so that they can be identified during design reviews. By looking at the possible causes of security design flaws, we can better understand where these design flaws might originate and what that type of vulnerability might represent for the application. Examples of security design flaws that are frequently identified during security design reviews include

- Not documenting a security requirement.
- Failing to design and/or implement a security control (e.g. authentication, authorization, encryption, auditing and logging).
- Leaving a gap in the design and/or implementation of a security control.
- Designing and/or implementing a security control that can be subverted and/or bypassed by a threat agent to conduct a business attack against the application.
- Not designing encryption to protect confidential data and exposing the data to information gathering threats.

- Authentication design flaws allowing a threat agent to either brute force or bypass authentication.
- Authorization design flaws allowing a threat agent to bypass authorizations (such as enforcing user entitlements based on client site parameters).
- Input validation flaws, such as relaying on client side data validations instead of server side validations, allowing a threat agent to bypass input validation to inject malicious data.
- Not logging security events, allowing threat agent attacks to be undetected and/or not conducting a forensic analysis of security incidents.

Since the chances of missing security requirements documentation leading to a security design flaw are very high, it is important to document security requirements as early as possible in the application/software development life cycle. By being prescriptive as to what the application/software is required to implement, it is possible to avoid several types of vulnerabilities. For example, not documenting security requirements, such as the requirement to implement authentication strength commensurate to risk of the transaction, can lead designing applications without MFA. Failing to document a requirement to encrypt authentication data, such as passwords in storage and in transit and restricting the use of hashes to standard hashing algorithms such as SHA-1 and SHA-256 and with salt, might lead to security design flaws that put all the application data assets at risk.

Not prescribing security requirements before coding might also lead to both design and coding flaws, such as not implementing a filter for malicious SQL injection commands before SQL queries are processed by the back end database and not requiring coders to use prepared SQL statements and store procedures to query data in the database.

Documenting the application security requirements is also important for software security assurance to assert these requirements with security assessments. By conducting a security design review against a set of security requirements, it is possible to identify any omissions in the design of security controls in compliance with the application security standards and information security policies.

Since security design flaws are usually not identified by automated tools, they require a subject matter expert/security architect to follow a methodology and to assess the application architecture by using an application security framework that considers both the potential threats to the application as well as the security controls that could mitigate these threats.

Identifying Security Design Flaws with Secure Design Reviews Security design flaws are in essence weaknesses and gaps in security controls that might originate from failure to document security requirements or in errors introduced during the architectural design of the application. The capability of a security team to identify all possible design flaws in an application depends on different factors, such as:

- Adoption of an architectural risk analysis/threat modeling methodology.

- Documented secure application standards.
- Documented design of the application to review.
- Secure code reviews of the application source code.
- Skill and experience of the security analysts.
- Cooperation of development teams to allow information gathering and sharing with the application security teams.

One of the prerequisites for security design reviews is gathering the technical documentation such as architecture design documents and technical and functional specifications of the application. This information gathering can also occur by conducting white board exercises with the lead application architects. To conduct the design reviews, it is necessary to refer to architecture design guidelines and standards, since these documents constitute the basis for validating the application security requirements.

The identification of potential security design flaws is facilitated by the understanding of the application logical and business context of the application more in depth. The business context can be understood by learning about the business objectives and requirements and the type of functionality that the application provides in support of the business objectives. Understanding the business and local context of the application is essential for determining potential business logic flaws that by might be exploited by a threat agent trying to attack the application. For example, assuming that the application executes online purchases by processing credit card data, it is important to understand the payment processing business logic and the controls including security controls such as authorizations and validation of payment data.

After a review of the business requirements, the threat modeling analysts focuses on the review of the technical and functional requirements and the design of the application architecture. One important aspect of understanding the application architecture is the decomposition of the architecture in its basic components such as user interfaces, servers, data assets, and data flows. This step is critical for threat modeling and secure architecture review process.

After the various components of the application architecture have been identified, the next step of the threat modeling exercise is to determine the exposure of potential threats targeting these components. First and foremost, it is important to consider the potential threat scenarios. Depending on the type of application (e.g. intranet or Internet), the type of service provided to the application's user (e.g. money transfers, payments, filing tax forms, health records, trading secrets on products), and the classification of the data being accessed (e.g. public, confidential, restricted, top secret-classified), as well as other factors that constitute the risk profile of the application, it might be targeted by certain threat agents (e.g. fraudsters, malicious users, or corporate and country sponsored spies) and attacks (e.g. social engineering, phishing, PC infection with malware through drive by download, or exploiting application vulnerabilities and weaknesses in security controls).

To conduct a manual review of the data flows, it is important to document sequence diagrams and data flow diagrams that describe the various interactions of the users

and application components with the data. Sequence diagrams, for example, show the data interactions between users and application components arranged in a time sequence. These diagrams are called sequential because data interactions with the various components are shown in time sequence. Sequence diagrams can be used to understand how the various data inputs and outputs are processed by application components and to determine the exposure of the data assets flowing among the different components of the application architecture.

Besides sequence diagrams, another graphical representation that can help a security architect understand how the data interacts with the components of the application architecture are Data Flow Diagrams (DFDs). DFDs help describe how the user's input is processed by the different processing components (e.g. web and application servers) of the application architecture. A simple DFD showing the flow of data credentials for both simple user and administrator and how these credentials are processed through the main components of the web application is shown in Figure 2.3.

In the DFD notation, each architectural component has a different graphical schematization:

- Input and output interfaces are schematized with squares.
- Application processes or functions with circles.
- Data storages with parallel lines.
- Trust boundaries with dashed lines.
- Input and output data flows across each of these components with arrows.

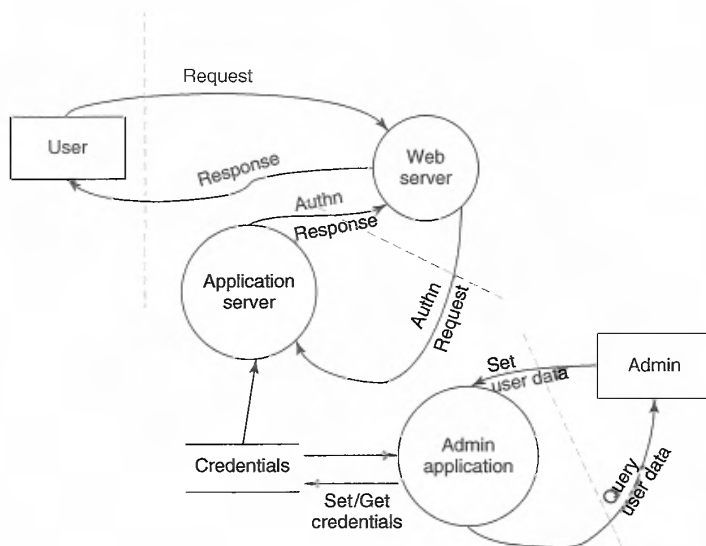


Figure 2.3 Example of Data Flow Diagram

At the architectural level, a threat analysis can be conducted by analyzing the exposure of potential vulnerabilities to different threats to the components of the application architecture identified in the data flows diagram.

The most important components to review for potential exposure of vulnerabilities to threats are the data assets, such as database storing confidential information and authentication data such as passwords. When these data assets are identified in a data flow diagram, it is possible to determine the data flows to query these data assets and how the data is potentially exposed while transmitted through the various components of the application architecture. When the potential exposure of an asset to a threat is identified, the next step is to analyze the risk of the vulnerability and the likelihood and the potential impact of that threat exploiting the vulnerability. The exposure of a threat by vulnerability might depend on the type of vulnerability, such as a security design flaw from a weak security control or not applying the security control all together.

Using Threat Libraries

An example of a threat library that can be used to identify the exposure to threats by potential vulnerabilities in application components is the STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege) threat library. STRIDE can be used against the various components of a data flow diagram such as client, servers, backend mainframes, and databases, and the data flows in between to identify potential exposure to STRIDE type of threats. This type of analysis is also referred as STRIDE per element and is part of the Microsoft TM methodology.

A STRIDE per element analysis can help identify the likelihood of threats based upon the exposure of that component of certain vulnerabilities:

Spoofing the user ID to impersonate a legitimate user by exploiting a session hijacking vulnerability to pose as a legitimate user.

Tampering with a file during transmission to change its contents because of no digital signature.

Repudiation of sending a file from an untrusted connection because of not implementing mutual authentication among servers.

Information disclosure of the contents of the file because of unencryption.

Denial of service on a web server because of not limiting the size of the HTTP requests processed by the web server.

Elevation of privilege to access other user's data because of failing to enforce a role base access control on the server to restrict access to the data based upon user's session and user's role.

By taking the attacker perspective, it is possible to determine the potential exposure of the application components to potential threats due to design flaws in the application architecture. Determining these design flaws might be facilitated by using

threat modeling as well as by the positive confirmation of the application architecture adhering to the security requirements and the application security standards.

From the threat modeling perspective, it is important to analyze how the data flows across the application components and validates the security controls at each component, starting from the various point of data entry through crossing boundaries to access the servers to reach the data assets. A trust boundary defines the boundary of trust that needs to be asserted as authentication and authorization checks for the data to move across a component, such as a web server, application server, or database server. By visualizing the different trust boundaries within an application, it is possible to determine whether basic security controls, such as user authentication and authorization, are enforced before the user is allowed to access the data under such a component's control. Another important aspect of the trust boundary is whether the connection among servers enforces nonrepudiation with mutual authentication (e.g. client to web before the data can flow across these tiers).

An important factor to consider for identifying design flaws is to characterize the goals of potential threat agents targeting the application assets. One possible goal might be to steal PII for identity theft. One possible threat scenario might include attacking the application to get the PII data by exploiting vulnerabilities in security controls to protect the confidentiality, integrity, and availability of such data. Therefore, a good place to start the threat analysis is to identify where PII data is stored, how it is accessed, and the security controls in place to protect the confidentiality, integrity, and availability.

Another approach to determine the vulnerability exposure to threats is to analyze at a design level how a vulnerability might be exploited at the different layers of the application architecture. Assuming, for example, that there is a known SQL injection vulnerability, we would like to determine how the risk of a threat exploiting this vulnerability can be mitigated by applying different security measures at the design level. These security measures might include for example:

- Input filtering in the web server using a Web Server API such as NSAPI or ISAPI.
- Input filtering in the application server using a servlet filter.
- SQL prepared statements for accessing the database, such as before serving the query to the database.

Because we can tie threats to vulnerabilities and vulnerabilities to the components of the application architecture, we can ultimately determine how to best apply defense in depth to mitigate the risk of the SQL injection vulnerability. For example, we could mitigate this risk by implementing countermeasure at all layers of the application architecture such as web server, application server, and data access components. The follow-up of this type of threat modeling/architectural risk analysis consists of documenting the design of input validation filter, such as NSAPI, to block SQL attack vectors as well as to enforce the secure coding requirement to use parameterized queries and/or store procedures for the application server components used to access the data assets. For complete assurance that security requirements for mitigating SQL

injection vulnerabilities are followed, validate that SQL injection vulnerabilities previously identified and fixed with a design and code change can be closed by retesting the application. In essence, these security measures can be designed and security tested thanks to threat modeling during design reviews. The different types of options to mitigate the vulnerabilities both at design and source code level provides engineering teams with several risk mitigation options to choose from. These security measure options can be applied at the different tiers of the application architecture (e.g. web server, application server, or database) in adherence to the defense in depth security principle.

Remediating Vulnerabilities and Design Flaws

Once application security design flaws are identified, the next step is to properly report any issues by highlighting the risk and the corrective actions for fixing them. The corrective actions are usually referred to by security professionals as recommendations for fixing the vulnerabilities and are part of vulnerability assessment reports.

In the case of issues identified during security design reviews of the application architecture, these recommendations should be included as part of the threat modeling and architectural risk analysis reports. Possible recommendations might be documenting a missing security requirement, changing the design to meet a missing security requirement, or fixing a design flaw. The information of how a security issue should be remediated is usually part of the recommendation for each issue and included in the vulnerability assessment report along with the type of the vulnerability and the risk severity. For a finding to be considered actionable toward remediation by the development team, is it important that the recommendation is clear and detailed enough so it can be documented in the corrective action plan. The recommendation would also need to include what the most probable cause of the vulnerability can be:

- Missing security requirements.
- Design flaw (e.g. gap in design of a security control).
- Coding error (e.g. security bug).
- Misconfiguration (e.g. misconfiguration of security control).

Classifying the root cause of a security issue can fall in any of these categories and help the development team address it with a design, source code, or configuration change. Classification of vulnerability root causes is also critical for managing vulnerabilities identified across different assessments such as security requirements reviews, application threat modeling/secure architecture reviews, source code analysis/secure code reviews, and application penetration testing/ethical hacking. Examples of vulnerability root causes might include gaps in requirements, errors in source code, flaws in design, and application misconfigurations.

A security issues categorization that includes root causes allows security professionals to quickly decide where to fix the vulnerability, such as by changing/adding security requirements, redesigning the application, fixing the security bug in the source

code, or changing the application configuration to adhere to secure configuration management procedures and guidelines.

The classification depends greatly on how the security issues are identified in the first place. In the case of security issues that are identified by testing the application inside out using testing tools such as automated source code analysis and manual code reviews, the causes of the vulnerabilities are typically either in the source code or the application binaries. The most probable causes of these vulnerabilities are insecure coding or insecure libraries that are integrated as part of the application builds. From a tactical perspective, fixing insecure coding would just require rewriting the code, removing the offending source code, and/or integrating with a nonvulnerable library. From a strategic perspective, the causes of insecure coding errors might be missing secure coding standards that developers can follow, not enforcing these standards, not providing adequate secure code training to software developers, not having dynamic application security testing tools, and not using secure components that are already validated for vulnerabilities.

In the case of security issues that are identified with outside-in type of testing, such as automatic dynamic security tests and manual penetration tests, there might be several origins, such as missing security requirements, design flaws, coding errors, integration and building with vulnerability libraries, or insecure application configurations and the servers where the application run before conducting these tests.

Therefore, when vulnerabilities are identified by testing the application from the outside in, it is not possible to determine “a priori” what the cause of the issue is since what is visible is the effect of the vulnerability and not the cause. To identify the cause of a vulnerability, we need to conduct a root cause analysis to correlate the non-visible causes of vulnerabilities within the application source code and/or design with their visible symptoms, the observable insecure behavior of the application. This type of analysis that identifies the non-visible causes of visible effects is generally referred as a root cause analysis and is exemplified in Figure 2.4.

To identify the root causes of vulnerabilities, it is important not to just stop at analyzing the symptoms but to look under the surface and understand what might cause them. To understand what the causes of vulnerabilities are, it is important to test the application both from outside out and inside in and to correlate the findings of the two types of assessments, a technique that is often referred to as gray box testing, a mix of black box and white box testing.

The black box testing goal is to test the application to identify vulnerabilities from the outside in by considering the application as a black box with no previous knowledge of how the application is designed and implemented. Examples of black box testing include manual penetration tests and using dynamic application security testing tools.

The white box testing goal is to test the application to identify vulnerabilities from the inside out by considering the application as a white box with knowledge of security requirements, design requirements, and source code. Examples of white box testing are source code analysis and using static application security testing tools.

Architectural risk analysis can also be considered a type of white box security testing for identifying potential design flaws (errors in the design of security controls)

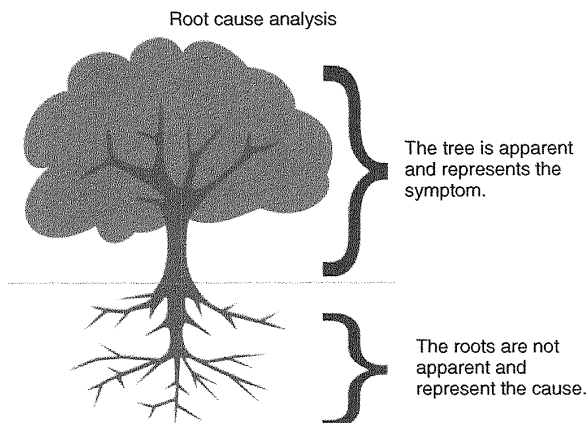


Figure 2.4 Root Causes versus Symptoms

and/or gaps (missing designs of security controls). Examples of the design flaws that can be identified include not implementing a security control, such as authentication, and storing sensitive information such as passwords in unencrypted form.

By combining the vulnerability assessments of architectural risk analysis, manual and automated source code analysis, and manual and automated dynamic testing, it is possible to cover different type of vulnerabilities and to correlate them to identify the causes of the vulnerabilities. For this reason, an overall application vulnerability report that aggregates the different types of vulnerabilities that are identified by other security assessment helps a risk manager determine the most effective risk mitigation measures to fix the root causes of vulnerabilities.

For example, security flaws identified with application threat modeling can be correlated to vulnerabilities found in secure code reviews and penetration tests to determine if these are the instances of the same root cause, such as a design flaw identified with threat modeling. A design flaw identified by threat modeling could be correlated to the same design flaws identified with a manual source code review. Reporting security issues by correlating them in one report helps risk managers focus on the corrective actions that address the root cause of the issue instead of the symptoms.

In general, to get the benefit of correlating vulnerabilities to identify the root causes, it is important to plan the threat modeling assessment ahead of manual secure code reviews. A threat model can be used to drive the secure code analysis toward the review of the source code of those components whose design flaws were previously identified in the threat model. The approach to use threat modeling for simplifying source code reviews is also referred as depth-first secure review instead of breadth-first for manual secure code reviews.

In general, the goal of vulnerability reporting is mitigation of the technical risks caused by vulnerabilities identified in the vulnerability assessments. The risk manager's main goal is to fix vulnerabilities depending on the level of risk based

on the factors of likelihood and potential impacts to the application assets. Any vulnerabilities of high risk are typically prioritized for mitigation over vulnerabilities with medium and low risks. Once vulnerabilities are prioritized for risk mitigation, the development team must determine how to fix them and document the design and code changes in a corrective action plan. For this reason, it is critical that the recommendation included in the vulnerability assessment is clear and detailed enough to allow the development team to implement the fix. Vulnerabilities that are marked as fixed when it is reassessed through a retest that is no longer present. In the case of automated vulnerability scanning, vulnerabilities should be considered fixed when it is no longer reported by the tool as a finding. In the case of vulnerabilities that are identified with manual testing such as penetration testing, the validation consists of validating that the security test case is negative. For issues that are identified in the source code through either a manual or automated code review, the test consists of either reviewing the source code manually to validate that the offending source code is cleared, or by retesting it with the static code analysis tool.

In the case of security design flaws that are identified through threat modeling and architectural risk analysis, the report needs to clearly articulate the risks posed by the design flaw as well as the recommendation to mitigate the risk. Since the fix of a design flaw typically involves a change in the design, it has to be articulated in terms of architectural changes of the application that can be understood by the application architects. The recommendation can also refer to secure architecture guidelines and standards that can be followed to implement the fix.

Ideally, the organization's security architecture guidelines should include a set of security requirements derived from compliance and regulations as well as recommended technologies and frameworks for designing application security controls. A reference to these guidelines should be part of the recommendations as applicable to each case of design flaws being reported.

Similar to vulnerabilities identified by other type of application assessments, such as source code analysis and penetration testing, design flaws identified through the threat modeling exercise can be classified based on their risk severity. The qualitative risk ranking model used for assigning the risk severity to vulnerabilities identified by other assessments such as penetration testing and source code analysis can also be used for ranking risk of design flaws identified with the threat modeling/architectural risk analysis. One way to rank the severity of a threat-vulnerability is to use DREAD factors (damage potential, reproducibility, exposure, affected users, and discoverability). A qualitative risk ranking of HIGH, MEDIUM, or LOW based on the scoring method, such as CVSS, as well as calculating risk factors, such as likelihood and exposure, also allows risk managers to prioritize fixing these design flaws according to their risk.

An important factor to consider for determining the risk mitigation strategy is the business impact of the design flaw. Design flaws that pose an impact to the business, such as monetary losses when exploited by fraudsters, help build the rationale for risk mitigation to the business that ultimately has to allocate the budget to either implement the fix or to acquire the security technology, process, and tool to mitigate

the risk. Reporting the business impact of the design flaw is therefore critical for empowering the business in making the informed risk mitigation decisions.

The recommendations issued in the threat modeling report need to consider the different levels of security knowledge and responsibility, including business, information and risk managers, and development teams. Ultimately, the real benefit of application threat modeling is that it empowers development, business, and security professionals to make informed decisions on how to manage application security risks throughout the software and application life cycles.

Another important factor to consider in vulnerability reporting is that the same application will be reassessed for vulnerabilities during its lifetime. This will include the vulnerability assessment in compliance with standard processes requiring security control effectiveness to be reassessed over certain time periods (e.g. every 6 or 12 months) or because of new application design and code changes.

As new types of assessments are rolled out for the organization, such as application threat modeling, it is important to be able to produce evidence to the business that threat modeling assessments are a factor of proactive risk mitigation and reduction of the overall risk to the business. One measure of proactive risk mitigation effectiveness is to correlate vulnerabilities that are identified earlier in the SDLC, such as during the design phase, due to adopting threat modeling as factor to a lower number of security defects identified later by other assessments such as source code analysis and penetration tests.

Another factor for proactive risk mitigation is classifying security issues by root causes, which is critical to correlate the vulnerabilities found by other assessment. A vulnerability classification that includes a categorization of the vulnerabilities by root causes, such as a design flaw in security controls including authentication, authorization, encryption, and session management for example, helps eliminate these issues before they are implemented and tested with source code analysis and penetration testing tools. The improved efficiency in managing the security issues attributed to threat modeling can also be shown in the vulnerability metrics when the number of vulnerabilities decreases over the lifetime of the different releases of the same application/software products. Typically, these positive trends in vulnerability and risk management are a factor of security activity, better training of the security team, and tools and technologies introduced to improve the security of the applications that are built and security tested.

The training provided to the security team and development workforce is probably one of the most critical factors in producing applications with less vulnerabilities to fix ahead of releasing the application in production. This is the case of applications that are designed and implemented by following security design principles and a security engineering process. Ultimately the threat modeling metrics, including the reports of the design flaws identified by threat modeling, need to take into account what is actionable to improve the security of the application both from the tactical risk perspective (e.g. fixing issues) as well as the strategic risk perspective (e.g. process improvements). From the tactical perspective, actionable security metrics include identifying the root causes of design flaws as well as pointed recommendations and

resources on how these can be fixed and prevented in the future. From the strategic perspective, actionable security metrics include recommendations for process improvements, including security engineering and security training. As the author of the book *Security Engineering* and renowned University of Cambridge professor Dr. Ross Anderson pointed out, “the goal of a good security engineering process is the understanding of the potential threats to a system, and then applying an appropriate mix of protective measures both technological and organizational to control them.”

From the tactical risk perspective, measuring the effectiveness of threat modeling might include producing successful trends that show that the organization is getting better at fixing issues. Remediation is now faster and cheaper than without using threat modeling. It is therefore important that the results of the threat modeling exercise provide enough information for the development teams to manage security defects more efficiently: a developer lead or an application architect would need to be able to take the results of the threat modeling and fix the design and/or coding to implement the fix for the security issue/vulnerability.

From the strategic risk perspective, measuring the effectiveness of threat modeling is possible when the organization has reached a certain degree of maturity so that a consistent security process, such as threat modeling, is being followed throughout the organization. Consistency in conducting the process is essential for producing consistent metrics so that design flaws are reported consistently by different security teams independently. This can be achieved when both security and application development teams follow the same methodology/process, take the required security training, and use the same technology and tools. The proactive risk mitigation improvements can be measured as positive trends in getting better at mitigating the application risks, being able to lower the overall risks to the business, and by providing visibility for the organization to become more efficient and effective in reducing the application security risks.

ANALYZING APPLICATION SECURITY RISKS

“Do not think of attack and defense as two separate things. An attack will be a defense and defense must be an attack.”

Kazuho Kudo

Analyzing Threats and Countermeasures

One of the main goals of information security is to protect the information assets such as company data from potential attacks. More specifically, this means protecting the data integrity, confidentiality, and availability.

From the business perspective, company data is considered an asset when data loss might have a negative legal and/or financial impact for the company. This valuable data includes PII, user credentials, and company secrets such as the company intellectual property. From the information security perspective, one of the main

security requirements is to protect the confidentiality, integrity, and availability of data assets as well as of the functions that process such data. Protection of confidentiality, integrity, and availability typically requires implementing security controls such as encryption, digital signatures, and authentication and authorization. From the defensive perspective, these security controls include both preventive and detective defenses to protect the company assets, data, and application functions from potential attacks against these assets. Implementing these security controls is often driven by the enforcement of information security policies that define which data assets and functions need to be protected and the protection levels that are required.

The need to protect company assets is driven by other factors besides compliance with information security policies. Protection is also driven by enabling business with customers through establishing trust with the consumers of these assets, such as clients and customers and the service providers of these assets. This establishment of trust between consumers and providers of data assets and services depends on different factors such as evidence that security controls are implemented, evidence of security reviews and vulnerability assessments, and security certification/assurance from third parties that measures are in place to protect the data.

Examples of security measures include preventive and detective controls whose function is to either prevent or detect attacks against the application. Some defensive measures are user authentication and authorization to restrict access to the application assets to authenticated and authorized users, encryption to protect the confidentiality of these assets when they are stored and transmitted, as well as auditing and logging to identify “a posteriori” who gained access to the application assets. Enforcing security policies to protect the application assets is typically based on a defensive perspective of security, not an attacker perspective.

From the attacker perspective, application defenses might constitute a deterrent to prevent attacks against both the application and the application assets. A deterrent security measure costs too much for an attacker to break. For example, an attacker must have specific knowledge to exploit certain type of vulnerabilities. Another important aspect of this is the cost for an attacker to run an exploit. When the cost of breaking a security control becomes affordable for an attacker to exploit, then the security control only gives you a false sense of security. An example of this is cryptography. In 1998, it took the Electronic Frontier Foundation \$250,000 to develop a machine to crack the DES encryption algorithm. Today a DES encryption can be broken by an affordable FPGA high-performance computer in a relatively short time (four or five days). DES is no longer considered a secure algorithm by US federal standards.

As the cost for acquiring tools to break into application security controls becomes cheaper, the deterrent effect of the security defenses becomes less important for an attacker. In some cases, the costs to conduct an attack compared to the possible rewards for an attacker, such as monetary gains of several million dollars, can justify the attacker’s cost in cybercrime tools. A further justification might come by the reasonable low risks of being caught by law enforcement because these attacks can be conducted in complete anonymity using malware dropping services, attack proxies, and command and control centers hosted in different countries. In some

cases, unmitigated vulnerabilities and design flaws in an application might represent an opportunity for an attacker to exploit security controls using relatively inexpensive tools. When this happens, the data breach might be undetected and the public disclosure of the vulnerability and data breach might represent a considerable reputational loss for the company.

Since the threat landscape has dramatically changed in the last 20 years, from exploiting vulnerabilities for fame and notoriety to targeted attacks for monetary gain such as fraud and identity theft, the challenge of who has responsibility to protect the application and the application asset has become much harder. The frequency and volume of data breaches occurring in the last five years represents a worrisome warning for companies today. The question is no longer IF but WHEN a data breach will occur.

Today, an application that is designed just to be compliant with information security policies cannot be considered secure enough to protect digital assets against motivated attackers. Just relying on application security because of compliance and audits only gives the defender a false sense of security. Securing data assets need to take into account both defensive and offensive perspectives of security, not just compliance and audits.

The defensive approach assumes that the application can defend against opportunistic and targeted attacks by focusing on preventive security controls and measures. Examples of defensive measures include following security standards and implementing security controls as required to protect the data and the application from unauthorized access, such as authentication and authorization, protection from malicious inputs by validating all inputs, protection from unauthorized access to sensitive data by encrypting data in storage and transit, and protection from exploits of session management vulnerabilities that can be used to hijack other user's sessions. Some defensive secure activities include designing applications by following secure architecture principles and adopting defensive coding following secure coding standards when developing source code for a given application/component.

The offensive approach assumes that the application can be targeted for opportunistic exploits of source code vulnerabilities, exploit of architectural design flaws, and functionality-business logic flaws. Targeted threats might use attack tools and techniques that are specifically crafted to exploit source code specific issues, specific business logic flaws for which the application is being designed and last but not least specific design flaws. Targeted attacks might also exploit common vulnerabilities but the attack vector might be specifically designed to exploit that common vulnerability type in a unique instance of source code error or design flaw.

From a defensive perspective, the fact that an application has been tested for common known vulnerabilities using generic attack vectors and the vulnerabilities are identified and fixed, is not a proof that the application security controls will not be bypassed and the data breached by an attacker. A simple example would be for an attacker to use social engineering and malware compromise on a host to steal the user's credentials to try and exploit the application business logic post-authentication. The offensive approach is the one that takes into consideration the fact that any layers of the application architecture, such as either the clients or any of the servers (e.g.

web server, application server, database servers, middleware, mainframes), can be compromised by using specific attack tools and attack techniques.

When considering the threat of targeted attacks, it is therefore important to consider all the factors of threat analysis that is the “who” the “what” and the “how” in essence who the threat agents are, what are the assets being targeted and how different attack vectors that can be used against a given application and its asset as a target.

By a general definition, a threat can be defined as a possible negative event whose occurrence might cause a negative impact to the application. The analysis of the potential threats targeting the application needs to take into consideration several characteristics of a threat, such as whether the threat is caused by an external or an internal agent, the type of threat agent (human and not), the likelihood of the threat to be realized (assuming possible points for attacking the application), and the degree of knowledge of attack techniques and tools required to conduct the attack.

For a threat analysis to be consistent and accurate, it is important to follow a threat analysis process step by step. Initially it is important to identify the possible threats and threat agents that could potentially target an application and the application assets. It is therefore important to identify the application’s assets, such as the data that is stored and processed by the application as well as the application functions. Once the application assets and functions are identified, the next step is to determine their probability of being targeted. For example, an internal only application whose data is not confidential is less likely to be the target of an attack from an external threat agent seeking to steal confidential information. To conduct the threat analysis in objective manner, it is important to analyze the threat likelihood as well as the potential impact using a threat library. With a threat library, it is possible to enumerate all possible threats and the potential targeted asset types and choose the ones whose characteristics of threat agents and targeted assets match the assets (data and business functions) of the application. By matching the characteristics of the application asset to the potential threats affecting them, it is possible to put together a threat profile for the application.

If a web application is already operational, the theoretical threat profile can also be augmented by considering historical data of security incidents, such as security incidents that have targeted the application or applications that have a similar inherent risk profile (e.g. similar data, functions, and exposure to threats). One example of a data source that can be used for building the threat profile for the application includes the history of data breaches and fraud committed against the same or similar applications.

After building a threat profile for the application, the next step is to analyze the attacks. Attacks are different from threats. A threat represents a potential negative event whose risk be analyzed by considering the likelihood and impact to a certain target. At a high level, a threat describes a potential negative event in terms of threat agents and targeted assets. An attack is the action undertaken by the threat agent to realize a threat. An attack can be described in terms of attack techniques and tools and can be analyzed in terms of the different steps that a threat agent takes to cause a negative impact to an asset. For example, the threat can be described as a distributed denial of service against an Internet application from a threat agent classified as an

activist group targeting similar applications in the past (online banking application). The attack involves using a compromised group of hosts to direct a high volume of network traffic and HTTP requests toward the website to exploit the servers processing resources and causing an impact on the availability of the website to customers.

Identifying how threats can be realized through the exploit of potential weaknesses in the application is one of the goals of an attack analysis. Vulnerabilities are considered valid independent of the threat and attack vector used to exploit them. The information provided by the threat analysis and the determination of the attacks is critical to determine how the vulnerabilities can be exploited, the assets that can be affected, and the potential impact on them. Vulnerabilities are not necessarily caused by flaws introduced during design, bugs during the coding of the application, misconfigurations of security controls, or missing a security test, but can be the cause of the lack of all these security controls and measures.

Identifying any gaps in the requirement of deploying certain security controls, such as missing authentication or input validation, represents a vulnerability for the asset that the security control is supposed to protect. A vulnerability such as security control gaps and/or control weaknesses represents an opportunity for conducting a certain attack. The type of attacks and attack techniques that exploit gaps and weaknesses in security controls have negative effects on the data assets and functions that these security controls are designed to protect.

The description of the attack includes the type of threat, the vulnerabilities that are exploited, and the asset impacted, and allows the qualification of the risk of the attack, such as the probability and the technical impact. In essence, attack analysis is a prerequisite for an assessment of the business risks faced by an application. For a security tester whose job is to identify vulnerabilities, exploiting the vulnerability is not required to assess the technical risk of the vulnerability. For a risk manager who needs to decide whether the vulnerability represents a risk to the business instead, qualifying how the application assets can be compromised by an attack such as an external or internal threat agent, the type of attack tools, and knowledge of attack techniques required to conduct the attacks is critical to make a decision on the overall risk of the vulnerability and decide what to do about it.

Some attack vectors that can be used for vulnerability exploits are used by security testers on applications. Often these attack vectors are embedded in security testing tools and are used to positively assert the exposure of the vulnerability by simulating the attack as a real attacker would do, but in a test environment and with test data to avoid any potential impact to either the application or the application assets. Identifying these vulnerabilities is limited to a black box analysis by using attack vectors and observing how the application reacts to them. A more comprehensive security test for vulnerabilities in an application also requires understanding the business logic of the application, how the application is designed, and the availability of source code to conduct a secure code review and identify potential security issues in the source code.

Often vulnerabilities originate prior to dynamic security testing. For example, vulnerabilities might be introduced as design errors and can be identified and fixed through reviewing the application architecture during the design phase prior to black box testing the application.

Other type of vulnerabilities might originate during the coding of the application as coding errors, or during integration with vulnerable libraries and components and during configuration. Vulnerabilities in the source code are not necessarily identified by applying attack vectors, but by examining applications for vulnerability conditions in the source code statically, without running the application. An example of static vulnerability identification techniques include static tainting, which is correlating a potential attack source to a place where the vulnerability can be exploited, referred to as the sink. The source of an attack in static tainting analysis does not qualify the type of threat agent that can be used for the attack, but the type of potential attack vectors that can be used. The threat agent can be internal or external and can be remotely controlled malicious software or a malicious user. Without specifically determining what the possible threat agents are and the attack vectors that these might use to the exploit the vulnerability identified in the source code, it is rather difficult to determine the risk of the vulnerability. This is where threat and attack analysis helps. By using a threat model that correlates the vulnerability with different threat agents and attack vectors, it is possible to determine the likelihood of the exploit and the technical risks of that vulnerability in terms of probability and asset impacted. Once the technical risk is calculated, the next step is to determine the business impact based on the asset value and make the final business decision of what to do about the risks. In case the risk is high and needs to be mitigated, the next step is to determine which and where security controls need to be applied. By using a threat model that includes a threat and attack analysis, it is possible to determine the type of threats and the different layers of the application architecture that can be attacked to exploit the vulnerability and negatively impact the asset. Through the threat and attack analysis, it is possible to determine which attack techniques the threat agents can use to cause a negative impact to the application assets and determine where and which security controls should be put in place to prevent and detect these attacks before any vulnerability is exploited.

Threat and attack analysis is a critical element of any threat modeling exercise and is essential to determine the risks and decide how to deal with the risks. Using information from the analysis and exercise, risk managers make informed risk decisions on whether to mitigate, transfer, or accept the risk. The effectiveness of such threat and attack analyses depends on the analyst's technical and business knowledge of the application and also on adopting a risk framework and a step-by-step process to conduct the analysis so that it can produce objective, consistent, and reliable results.

Analyzing Threat Agents and Threat Scenarios

The NIST special publication 800-30 Rev a, Risk Management Guide for Information Technology Systems, defines a threat as “the potential for a particular threat-source to successfully exercise a particular vulnerability.” Characterizing threats is essential for analyzing risks. A threat, either human or nonhuman, is a condition of a possible adverse event that might negatively impact an asset. A threat can be characterized by different factors, such as the type of a threat, the threat source (e.g. the threat agents), and the targets (e.g. application assets).

An important factor in characterizing a threat is the cause of a threat. We will refer to the cause as the threat agent. Each threat has one or more threat agents. Identifying a threat agent is critical for qualifying risk. Risk is defined as “the probability of a threat agent to exploit a vulnerability to cause a negative impact.”

A prerequisite to identify the possible threat agents targeting an application is adopting the threat agent taxonomy, which is the classification of possible threat agents. A comprehensive characterization of threat agents should include different types of sources for possible threats including human sources (people-based threats), nonhuman sources (e.g. malware), and mixed (e.g. a blend of human and nonhuman sources).

Examples of human-based threat agents targeting an information system might include the following:

- Political hacktivists.
- Cyber-criminals.
- Cyber terrorists.
- Disgruntled employees.
- Fraudsters.
- Cyber spies (Industrial and Government).
- Fraudsters.

This classification of human threat agents is important to understand the threat agent’s motives, their ultimate goals, and the attack techniques and cybercrime tools that are used to attain such goals.

Examples of cybercrime tools that threat agents use to conduct attacks include:

- Malware software such as Trojans, Viruses, Worms.
- Key loggers.
- Spyware.
- Cyber weapons.
- Botnets (client and servers).
- Hijacked hosts and processes.

Nonhuman threat agents might include threats that are not initiated by humans, also referred as “acts of God,” that can adversely affect an information system such as:

- a power-outrage at a data center hosting an application caused by nonhuman event such as a storm, or
- the destruction of the computer and data resources whose application relies on caused by a fire, earthquake, or tornado.

Once the threat analyst has decided which threat agent classification then to use, the next step is to analyze the probability of these threat agents targeting an application and the potential gains derived from attacking the application.

From the threat agent perspective, an application could become a target when an attack provides a return on investment to the attacker, such as a financial, political, or strategic reward.

The first question that a threat analyst ought to consider is which type of threat agents might attack the application based on the functionality and data assets that the application stores and processes. It is therefore essential to capture the application business context: the type of application functions, the type of application functionality that is provided to the users, and the type of assets that are stored and processed in order to provide such functionality. After capturing the business context and having identified the application assets, the next step is to identify the possible threats agents targeting the application.

For example, a website that provides online services for a given organization, private or public, that is considered hostile by a certain group of hackers, could be targeted by attacks to disrupt or to take down the website hosting the application, such as denial of service attacks.

Another example is an Internet web application of a financial institution that provides financial services, such as online banking, whose data processed, includes customers' confidential information such as bank accounts and credit card data. Fraudsters might attack the application to take over the customers' accounts for fraudulent transactions, such as moving money from the bank into fraudulent accounts, as well as to gain unauthorized access to the customers' confidential information.

Identifying the threat agents is critical to the definition of a threat scenario that has the application as a potential target and includes the threat agents, their possible motives, and the likely targets that include both the application functions and the application assets. Different threat scenarios can be derived depending on the business context of the application. Some basic examples of threat scenarios are:

A business critical web application that provides online banking services for customers. Threat agents that have targeted this type of application in the past include script kiddies, hackers and crackers, fraudsters, carders, and organized cybercriminal gangs and groups. The type of attacks include exploiting application vulnerabilities to cause denial of service, defacing web pages, and compromising customers' confidential information for identity theft and fraud.

A site that provides services to citizens to renew a driver's license or pay car registration fees. Past attacks include actors that steal confidential PII for identity theft such as names, addresses, date of birth, and driving license numbers. Other attacks pursued by these threat agents in the past include exploiting application vulnerabilities to cause denial of service, defacing web pages, as well as compromising customer's confidential information for identity theft and fraud.

A company website that provides R&D department users access to classified trading secrets and restricts access only to authorized employees. This type of application needs to consider insider and outsider threats for industrial espionage and unauthorized access to confidential and restricted information for financial and business gains.

An intranet workforce web application only accessible to company employees so they can access their salary statements and PII. This type of application needs to consider the risk posed by insider threats, such as disgruntled and terminated employees

abusing of their privileges to access some other employee's PII as well as damaging the company's reputation.

Deriving the application threat scenarios is a fundamental step for threat and attack analysis because it focuses on the characteristics of a specific threat landscape for the application based on the type of application, the assets, and the application functionality. Applications with similar functionalities and data assets stored and processed might have very similar threat scenarios and become the target of similar threat agents. By deriving a set of threat scenarios for each application, it is possible to focus on specific threats targeting the application, analyze the attacks, and prioritize the security measures to mitigate the risks. Since the threat agents' motivations, targets, attacks, and techniques evolve over time, it is important to keep the threat scenarios up to date by analyzing from different threat intelligence sources, new attack techniques being used, the types of vulnerabilities that are exploited, the reported security incidents, the types of functionalities that are exploited, and the types of data being breached.

Application Assets as the Threat Targets

After having captured the most likely threat scenarios for each application and keeping these threat scenarios up to date with threat intelligence information, the next step is to determine how the threat agents could attack the application and its components to realize their ultimate goals.

At the application architecture level, the focus of the threat analysis is to understand how different threats can target the several components that make the application architecture. In the case of a web application, the attacks that can be used to target the application assets might involve spoofing the confidential data in transit from the client to the server by exploiting flaws implementing a secure channel between the client and the server, tampering with the data to inject malware, hiding the source of the attack to prevent identification (e.g. by repudiating the source), or gaining unauthorized access to confidential information by exploiting weak authentication and so on.

In order to analyze the effects of threats consistently for each of the application components, Microsoft (MS) has derived a threat framework called STRIDE. The STRIDE threat framework is used in the MS threat modeling methodology to identify threats affecting each component of the application architecture. At the application level, identifying these threats points security professionals to security measures that need to be put in place to mitigate the risk posed by the threats. For example, since any data in transit can be attacked, security measures must be put in place to protect the application component from threats such as channel encryption to prevent spoofing, digital signatures to prevent data tampering, mutual authentication to prevent source repudiation, data encryption in transit to prevent disclosure, availability of the connection to prevent denial of service, and enforcement of role base access control while accessing the data to prevent elevation of privileges. By following the STRIDE methodology, a threat analyst can determine whether the application security controls are strong enough to protect the application components from these threats. If an

absence of a security control is identified, a design flaw whose potential exploitation from an attack vector could negatively impact the application and the data.

A threat framework can also provide mapping between the threats, the vulnerabilities that can be exploited by these threats, and the security measures that can be adopted to mitigate the risk posed by these threats.

Analyzing how a threat can exploit one or more vulnerabilities to realize the threat agent goals consists of identifying the different type of attacks that can be used to realize these threats. A threat framework can include a predefined set of threats and attack libraries that can be used to specifically attack the several components of the application.

Threats of sniffing/network eavesdropping, for example, might be realized through man-in-the-middle attacks. MiTM attacks work by compromising the client through socially engineering the victim to accept a spoofed certificate that allows the attacker to capture any data traffic in transit between the client and the server. Threats of information disclosure affecting the data in storage and in transit might exploit weak authentication to access the data and weak encryption to protect the confidentiality of the data. Information disclosure threats might exploit input validation vulnerabilities by attacking the application with buffer overflows, cross-site scripting, SQL, LDAP and XML injection attacks, format string attacks, HTTP redirection, and response splitting attacks.

When conducting a threat analysis of the threats affecting different components of the application architecture, using a threat modeling tool that incorporates threat frameworks, such as STRIDE, and automates threats generation based on the information of the application components can be very useful. A threat modeling tool might include a visualization tool to capture the architecture logical diagrams depicting the different components of the application architecture such as user interfaces, data interfaces among components, and the different components such as web servers, application servers, databases, mainframes, and middleware components. Such visualization tools can be used to decompose the application into the different components and help identify the threats that affect each component. The characterization of the application architecture in components also helps identify the data flows, data interfaces, and trust boundaries that need to enforce authentication and authorization controls for the application users.

A threat modeling tool, such as the MS threat modeling tool, drives the threat assessment from the perspective of protecting each element of the application architecture from possible exploit of vulnerabilities that might result on a compromise of the data as an asset whose confidentiality, integrity, and availability need to be protected. By considering the threats, the vulnerabilities, and the impact to the data assets, the threat modeling tool facilitates the threat analysis by taking into consideration both the attacker and the defender perspectives.

Analysis of Attacks and Exploits of Vulnerabilities

An attack can be defined as the realization of the goals of a threat. An attack describes how a potential negative event is realized by describing how the source might use

different types of attack tools and techniques that exploit previously known application vulnerabilities to cause a negative impact to the application assets that include both application functions and application data. An opportunistic attack might exploit common application weaknesses in the security controls, such as vulnerabilities in authentication and authorization, protection of confidential data, as well as gaps in detective controls such as auditing and logging and secure event monitoring.

A targeted attack will seek to identify first what defenses are in place by trying different attack techniques and by changing the attack vectors until the defenses can be bypassed. By comparison with opportunistic attacks, targeted attacks are more time-consuming for an attacker and require the attacker to adjust the attack to the defenses in place to try to bypass them and gain access to the application. Behind targeted attacks, there are typically motivated threat agents whose time and effort are justified by the financial gains and high return on investment of the time and resources spent attacking the application.

The extent to which a threat can be realized through an attack depends on different factors, such as the probability of success of the attack due to exploiting known vulnerabilities as well as the potential gains that might justify a threat agent acquiring knowledge of new attack techniques and investing in more powerful attack tools. It is important to look at these factors more in depth to analyze the possibility of an attack's success. Some factors that influence the success if conducting an attack are:

- Public information about the organization and press information that exposes it as potential target by motivated attackers.
- Public events that might facilitate the attacks such as company holidays or increased online shopping during holidays.
- Easiness of conducting the attack due to availability of attacking scripts and limited knowledge needed to run them.
- Knowledge and experience of the attacker/threat agents.
- Availability of information on how to conduct the attack (e.g. information shared among threat agents).
- Availability of tools for exploiting the application vulnerabilities.
- Availability of tools specifically designed to attack the application (e.g. banking malware).
- Reproducibility of the attack techniques to exploit a known vulnerability.
- Reliance on manual and automatic methods to conduct the attacks.
- Level of skills required to conduct the attack.
- Costs of tools and resources necessary for the attack, such as free attacking tools or tools that attackers can afford.
- Exposure of the application (e.g. attack surface) to the attacker.
- Exposure of the application vulnerabilities that can be either necessary or useful to conduct the attack.
- Public information about the application vulnerabilities and exploits.
- Not yet published vulnerabilities (e.g. zero-day).

- Public information about the application's inner workings such as source code, patents, technology stack and so on, which might facilitate discovering vulnerabilities and conducting the attack.

One methodology that helps analyze how a threat can be realized through different attacks is to use attack trees. An attack tree allow risk professionals to analyze different attacks by starting from the attacker's main goal and walking through the branches of the attack tree to determine the different ways the attacker's goal can be realized. Each branch of the attack tree represents possible opportunities for the attacks to be realized. Examples of these opportunities include the availability of vulnerabilities to be exploited, the availability of attacking tools, and knowledge of attack techniques that the attacker might have or need to acquire. In essence, each of the attack tree paths represents a possibility for an attacker to attack the target.

Once all attack paths are explored with an attack tree, it is also possible to determine which one among these attack paths an attacker would most likely pursue. The most likely attacks are the ones that have a high return on investment; the ones that minimize the cost in time and resources for the attacker goal while maximizing his gains. The top of the attack tree represents the attacker's main goal. By walking the attack tree from the root to the different nodes of the branches, it is possible to analyze all the different attack scenarios, explore the different attack paths, and validate the opportunities for an attacker to achieve his goals.

From the defensive perspective, understanding all possible avenues for an attacker to pursuit allows risk managers to identify the security measures that can be put in place to prevent these attacks. A very simple example is the condition of a vulnerability exploit necessary to pursue a certain attack path. Remediating that vulnerability will represent a lost opportunity for an attacker and will stop his attack at that node of the attack tree where the condition of the vulnerability is necessary to move further toward the attacker's goal.

The first step in analyzing an attack tree is to qualify what the possible goals of an attacker are. There are different types of attackers or threat agents, and each threat agent might have one or more goals to achieve by attacking the asset, application data, and application functions.

It is important to identify first and foremost which threat agents' goals would make them most likely to target the application. These agents pose the greatest risk to the application. If the goal of a threat agent is to commit fraud with credit card data, for example, one option for the threat agent is to attack either the credit card holder or a website that gives the credit card holder access to credit data information and transactions, such as requesting a credit card balance, linking the credit card to a bank account, and transferring money from that bank account to pay off the credit card.

Since attacking the credit card holder is much less expensive for an attacker than trying to attack the credit card web application, he will attack him first. An easy social engineering attacks directed toward a credit card holder is to use phishing e-mails with malicious links or malicious e-mail attachments that install malware on the credit card holder's PC to capture any website credentials with a key logger. Besides phishing, the attacker might try to use tools that exploit vulnerable web servers to install malicious

software that infects site visitors through drive-by-download techniques and exploit of common browser vulnerabilities.

In case the selected attacks fail, the attacker might decide to attack the website directly by opportunistically exploiting common vulnerabilities such as XSS and SQL injection. If unsuccessful, the attacker might try a set of blended attacks that use phishing and drive by download to install Trojans that perform man in the browser attacks, or use more sophisticated attacking tools that allow him to capture the credit card data while in transit between the client browser and the application with man-in-the-middle attacks.

All these attack paths that could be pursued by an attacker can be documented in an attack tree to visualize how an attacker could pursue his goals based upon opportunities (e.g. vulnerabilities, exposure to data and functionality) and preconditions (e.g. attacker tools, knowledge and experience of vulnerabilities and exploits). To better analyze attack scenarios, using attack trees allows risk managers to explore the “what if” scenarios and understand how the attack can be realized. By assigning each node of the tree factors that qualify the likelihood (e.g. how easy and exposed is the vulnerability that can be exploited), it is also possible to estimate the attacks that are most probable or improbable to occur. By assigning a cost for the exploitation of the vulnerabilities, it is possible to estimate the attack scenarios that require the least resources. The combination of these criteria helps determine the attacks that are most probable and less expensive to exploit. The attacks that are most critical for likelihood and impact are the ones that need prioritized remediation.

Attack scenarios that are preventively analyzed using attack trees are also very useful for determining the best way to respond to a security incident by deciding which security measures are most effective to stop an attack. Attack tree analysis can provide useful information for incident response procedures during the preparation and identification phases. When attack scenarios are analyzed with an attack tree, it is possible to trace the possible conditions that might lead to an attack and devise the appropriate countermeasures to contain and mitigate the attack and similar incidents.

The attack tree analysis provides a methodology for a threat modeler to analyze graphically or textually the different possible means or ways to realize the attacker’s man goal. Assuming that the attacker’s main goal is bank account compromise, for example, the attacker could attack the user credentials to log into the application via social engineering/phishing attack or by trying to brute force the authentication. By using attack trees, it is possible to identify the main goals for the attacker and the means to deliver such attacks by exploiting different vulnerabilities.

An attack tree that visualizes many attack scenarios can help visualize these attacks. The attacker can try to exploit input validation vulnerabilities by attacking the communication channel between the user and the application, such as in the case of MiTM.

Data protection vulnerabilities, for example, can be exploited to get the user’s credentials by sniffing them while in transit. The data flows between the user and the application help visualize the necessary controls to validate the data input at the entry points of the application tiers before being processed by the application components.

Vulnerabilities in authorization controls, for example, can be exploited by an attacker to elevate privileges through session hijacking. The threat model can visualize the several trust boundaries that can be bypassed or broken to gain access privileges to the application components and data assets.

For a threat model to be effective, all potential attack scenarios and attacker goals must be considered. Depending on the data and functional assets of the application, this will include any data assets, such as the customer's sensitive and confidential data, the user credentials, and the business functions-services offered to the customers through the application.

By identifying potential vulnerabilities in security controls and mapping these to the threat agents, attack vectors, and malware tools that can be used against the application, it is possible to determine the risk posed by the application vulnerabilities and prioritize them for remediation based on the risk that they pose. At this point of the threat analysis, a threat model can be used in conjunction with other vulnerability assessments, such as penetration testing and source code analysis, to assess the likelihood and business impact that exploiting these vulnerabilities might cause.

Identifying all possible threats and vulnerability scenarios can be a daunting task for a threat analyst, since in the real attack scenario an attacker might try a combination of several attacks that exploit all possible vulnerabilities in the application. From a defensive perspective, it is more important to focus on the most critical threats, such as the ones that are most likely to be realized and cause the most business impact.

In the case of attackers targeting a web application, for example, it is important to focus on characterizing the main threats and how these threats can be realized by the different attack vectors by exploiting vulnerabilities. An example of attacks and possible vulnerability exploits is provided in Table 2.5.

Mapping threat agents to attacks and attacks to vulnerabilities that these threat agents can exploit allows a threat analyst to get a better understanding of how potential threats to the application and the data can be realized through exploiting application vulnerabilities as well as weaknesses in security measures and controls at the high level.

Another factor to consider in attack analysis is identifying the attack surface. The attack surface represents the extent to which a target is exposed to potential attacks, such as through different entry points, either internal or external. An application that can be accessed through different interfaces, both web and mobile, has a larger attack surface than a web-only access.

Entry points and access levels are also important factors to consider. The data entry points identified with a data flow diagram and document on the level of authentication and authorization required to access the data for different users is a very critical factor to determine the exposure of a given application to attacks. Entry points can also be qualified as high risk when they are exposed to external users whose access to the application is allowed from both visitors and customers from untrusted client browsers and desktops. Entry points are considered medium risk when they are only accessible to internal users whose client browsers and desktops can be trusted, since they are subjected to more restrictive security controls such as patching, event monitoring, intrusion detection, and prevention systems.

TABLE 2.5 Mapping of Threats to Vulnerabilities

Attack	Possible Vulnerability Exploits
Social engineering – phishing a user to deliver malware (*)	Luring users to select links to sites that are vulnerable to cross-site scripting, cross-frame scripting, cross-site request forgery, HTTP response splitting, and invalidated redirection flaws.
Attack application entry points to exploit application vulnerabilities	Weak encryption in storage and transit, weak authentication, weak enforcement of user authorization, forceful browsing, design flaws in business rule enforcement, authorization flaws, lack of input validation, access to data using insecure references to objects and resources, malware injection, exploiting lack of input validation, remote file inclusion, file upload vulnerabilities, lack of mutual authentication between client and server, unrestricted access to resources/systems, and lack of enforcement of minimum privileges and user roles to access data.
Attack the application session (account take over) to cause fraud and monetary loss	Client PC and browser vulnerabilities leading to malware compromise enabling online session and account take over. Additionally, exploiting site session management and authorization flaws and lack of adequate machine tagging information, logging, monitoring, and fraud detection to monitor, detect, and alert for occurring fraud.

(*) Note: social engineering a victim can be used to collect sensitive information directly from the user by redirecting the victim to a malicious link or indirectly by installing malware with key-logging capability.

Once all avenues of attacks against an application have been explored, it is important to determine the path of least resistance for an attacker. The attacker’s path of least resistance is the one whose vulnerabilities are easier to exploit, or the attack paths that exploit gaps in security measures.

Any vulnerability or security control gap that is part of an attack path of least resistance needs to be prioritized for remediation since they represent the highest probability of exploit in an attack. By prioritizing vulnerabilities based on the likelihood of the exploit, risk managers can ultimately make better decisions on how to mitigate risks.

Analysis and Management of Application Risks

The risk exposure of the various data assets of an application depends on different factors, such as the availability of a user interface with data entry points that can be exploited, pre-authentication application functionality that can be attacked, unenforced data protection, and unprotected access to data. Any vulnerability in the application that allows an attacker to break into the application and gain unauthorized

access to application assets represents an opportunity for exploit and could lead to a negative impact for the organization whose responsibility is to protect that data. It is important to look at each opportunity an attacker has to attack the application as a potential risk. The mitigation of the impact of this risk consists of removing the opportunity by protecting that application functionality and the data with a security control, removing the vulnerability, removing the functionality and the data, or making it difficult and expensive for an attacker to attack the application and the data by strengthening the security controls around it (e.g. adding MFA).

It is important to analyze the different layers of defense in an application and to analyze the different data interfaces and user interfaces and the possibility that these might be attacked by a threat agent. Assuming, for example, that an external threat agent's goal is to gather confidential data stored in an application database, he will consider different possible attacks. If the application is found to be vulnerable to SQL injection, the attacker would try to exploit this vulnerability first. If it does not appear that an application can be attacked pre-authentication with SQL injection vulnerabilities, the attacker will try to exploit other vulnerabilities, such as XSS to run scripts on the browser and steal user session cookies as well as run key loggers and proxies to entice the use to accept spoofed SSL certificates to spoof the channel with MiTM attacks.

The likelihood of the attack's success to exploit post-authentication vulnerabilities is usually much lower than pre-authentication. Nevertheless, there are other vulnerabilities that can be exploited by attacking the client directly, such as browser vulnerabilities and social engineering attacks such as orchestrated phishing campaigns whose objective is to infect the client host and steal web application credentials to log into the application. Since multiple application layers can be attacked by a threat agent, data asset protection from potential impacts ought to employ multiple layers of protection according to "defense in depth" principles.

A threat modeling exercise that visualizes the different layers of the application architecture by decomposing it to architectural tiers that include front end interfaces, middle tier web servers, application servers, and backend tiers that include the data assets stored in databases and main frames, allows the threat modeler to validate if defense in depth principles are followed when designing the application architecture. After trust boundaries are identified and authentication and nonrepudiation controls are assessed at each tier of the application architecture, the next step in the threat analysis is to determine how the other components of the architecture, such as the data connectivity among servers and the processes themselves, are exposed to potential threats. The simplest way to look at threats impacting data flows and processes is to consider threats that impact data and process security attributes, such as the data confidentiality, integrity, and availability.

Once the likelihood and impact to the confidentiality, integrity, and availability of the data are evaluated, it is possible to qualify the risk as a factor of probability and impact. The evaluation depends on the technical risk model used to calculate the risk. One technical risk model that can be used to rank the technical risk of vulnerabilities is the industry standard MITRE CVSS model. This model makes it possible to rank the risks of vulnerabilities by assessing several intrinsic and nonintrinsic factors (e.g.

change of risk over time or environment factors); the basic factors of exploitability and impact.

According to CVSS, the intrinsic risk of vulnerability can be assessed for the following factors of exploitability:

- The vulnerability could be exploited remotely.
- Any pre-authentication condition required to attack.
- The complexity of the attack to exploit the vulnerability.

The following factors can be used to assess the impact of a given vulnerability:

- Exposure to confidential data.
- Possibility of damaging the data integrity.
- Possibility of impacting the data availability.

Analyzing the application architecture and exposure of the architectural components to potential threats can also be used to determine the CVSS factors of exploitability (likelihood) and impact and rate any vulnerability that is identified in a threat model. The information captured in a threat model, for example, can help determine whether vulnerabilities in an application are exposed to attack from a remote threat agent and if this threat agent has access to the vulnerabilities pre- or post-authentication.

The impact of the vulnerability can also be assessed by identifying any exposure of the confidentiality, integrity, and availability of the data due to presence of the vulnerability. For example, with a threat model that includes a data flow analysis, it is possible to analyze the flow of data from a threat agent attacking an external interface to the data storage component that can be impacted. If the data storage element stores confidential data, the vulnerability might have an impact on the data confidentiality and integrity.

By analyzing the data flow, it is also possible to determine if the vulnerability exposes the confidentiality and integrity of the data in transit. For example, assume that sensitive data can be transmitted between the user's browser and the web server. If the data is not protected with encryption during transmission, data confidentiality and integrity is exposed to spoofing, tampering, and information disclosure threats by a remote agent. Furthermore, if the connectivity between the web server and the application server is not authenticated and protected, attacks against authentication data might be exposed to repudiation threats post-authentication.

In case a vulnerability is found post-authentication, the probability of a threat agent exploiting it is diminished because the threat agent has to exploit weaknesses in the authentication first in order to gain access to the vulnerability and trying to exploit it. In the absence of a pre-authentication vulnerability, it is more difficult but not impossible for the threat agent to exploit it.

The easier attack for the threat agent at this point would be to try social engineering user to select a malicious link or malicious attachment in an e-mail whose end result

would install malware on the client, such as a key logger to capture online credentials. Since the application users are among the weakest links in any security control, the threat agent would likely attack them first. This type of attack would also be facilitated by vulnerabilities that can be used to attack the client, such as XSS, XFS, and CSRF.

The least likely attack, but not as difficult, would be brute forcing the password to access a web application. In any case, the presence of another vulnerability that can be exploited pre-authentication would increase the probability of any post-authentication vulnerability. For example, a post-authentication SQL injection vulnerability can be rated as a MEDIUM risk when it is only exploitable post-authentication. However, in the presence of vulnerabilities that have collateral damage, such as weak authentication, the exploitability risk of a post-authentication vulnerability, such as SQL injection, should be increased from MEDIUM to HIGH. Examples of weak authentication vulnerabilities include using weak passwords policies (length of the password is too short or the format is nonalphanumeric), flaws in the design of the authentication such as no rules to lock the user account for failed login attempts, too informative error messages when validating usernames and passwords, and usernames cached in memory and cookies or temporary files.

A threat model can also provide information to rate the complexity of an attack and help quantify the attack complexity risk factor. Typically, attack complexity analysis is best conducted using attack trees. Through attack tree analysis, it is possible to analyze the different avenues of attacks and exploits of vulnerabilities and also when an attack might be easy to exploit, such as when vulnerabilities are known and the type of tools and knowledge of the attack techniques are not difficult to obtain.

Using the CVSS model to determine the risk severity of the vulnerability, it is possible to assess the technical impact that vulnerabilities might cause to the confidentiality, integrity and availability of the data. The information provided by a threat model helps assess the factors of probability, such as exposure of the vulnerability to a threat agent, and impact: whether the vulnerability might be exploited to impact a data asset whose confidentiality, integrity, and availability need to be protected. Furthermore, when the technical risks due to vulnerabilities are qualified as HIGH, MEDIUM, and LOW risks, the next step is to associate the potential business impacts of such vulnerability exploits. A standard methodology to quantify the business risks is the quantitative risk analysis. This analysis calculates the value of the asset that can be lost in a single event, such as a security incident, whose cause is the exploit of a vulnerability factored by the frequency of this security incident measures on the yearly basis.

Based on an exploited vulnerability's economic impact to the organization, it is possible to make informed risk decisions, whether to accept the risk, mitigate the risk by applying a countermeasure, or transfer the risk to an external entity. Since the risk of the vulnerability exploit is monetized, it is possible to make decisions on whether to mitigate the risks by factoring the cost of the security measures and the potential economic impacts in case the vulnerability could be exploited. An investment in security measures, such as the remediation costs for fixing the vulnerability, is justifiable when the cost of the security measures is a fraction of the costs that the business will

incur if the vulnerability were exploited. That cost is calculated based on quantitative risk analysis.

It is common sense, for example, not to spend on security measures per year to mitigate the vulnerability more than the possible economic loss that the vulnerability is estimated to produce in a given year. The challenge of this approach is not how much to spend, but to produce a reliable estimate of how much the organization should value an asset in case an incident causes the loss of the asset. The value of the asset needs to be estimated by considering both direct costs and indirect costs that are difficult to estimate. Direct costs associated with an asset either lost or compromised in a security incident vary. Typically, security incident direct costs include the cost for responding to and recovering from the security incident (incident response costs), the costs for fixing the vulnerability that caused the incident, and the costs for replacing the asset that is lost (account numbers). Indirect costs are associated with tangible expenses, such as legal costs in case third parties take legal actions because of the incident, regulatory fines and audit costs, loss of future revenues due to loss of existing customers, clients, and investors as they react to the incident, as well as intangible loss, such as company reputation. In case the security measures cost too much, the business could consider transferring the risk to a third party that is willing to provide service based on taking a liability of a potential loss as well as by signing cyber insurance with a legal entity willing to insure the organization for the potential losses.