About Advertising Contact
Symantec ThreatCon
Level 1: Normal
Threat level definition

Nessus, Part 3: Analysing Reports

Harry Anderson 2004-02-03

1.0 Introduction

This article, the last in the series about Nessus, will endeavor to explain a Nessus report and how to analyze it. Nessus is a vulnerability scanner, a program that looks for security bugs in software. The first article explained how to install Nessus and a basic overview of features. The second article gave general rules of thumb for various scanning situations. It is suggested that you review the first two articles before reading this one.

Understanding how and why vulnerabilities exist in software is important to being able to analyze the final scan report, and this article uses a number of examples to illustrate various important concepts. In some cases these may be older, more common examples instead of the "latest" day zero vulnerabilities. There are several reasons for this. First, I don't currently know of any awe inspiring "zero day" vulnerabilities to wow the reader with. Second, the stated examples typically have a large body of information available for the reader to do further research, and I will list a few sites providing a good start for such research. Third, although some of these examples may be older, they are still quite common and unfortunately most readers will still encounter them when scanning.

2.0 Report generation

In the last article we covered the scan process. Once the scan has been completed the results need to be analyzed. These results can be directly viewed within a report generated with the client, either Nessus GUI (Unix) or NessusWx (Windows), or by having the data exported for analysis in an external program. If viewed from within a Nessus client, the results are arranged in a hierarchical fashion by the host. In NessusWX the scan results can be obtained by selecting the session and choosing Session | Manage Results. The results can then be viewed using the VIEW command, or written into a plain text (.txt), HTML, or Adobe Acrobat (.pdf) report. In the Unix GUI the results are automatically displayed upon completion of the scan, as shown below in Figure 1. In NessusWX, sample results are shown in Figure 2.

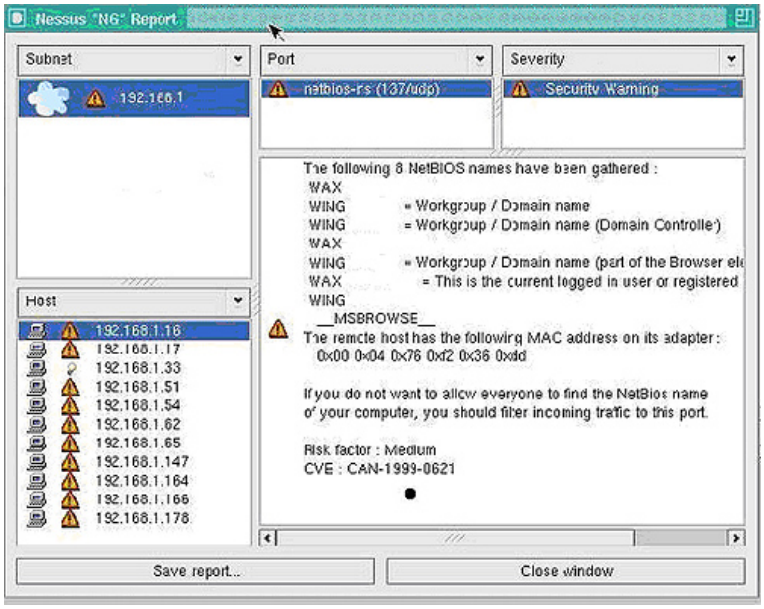

Figure 1: Results are automatically displayed in Nessus GUI.

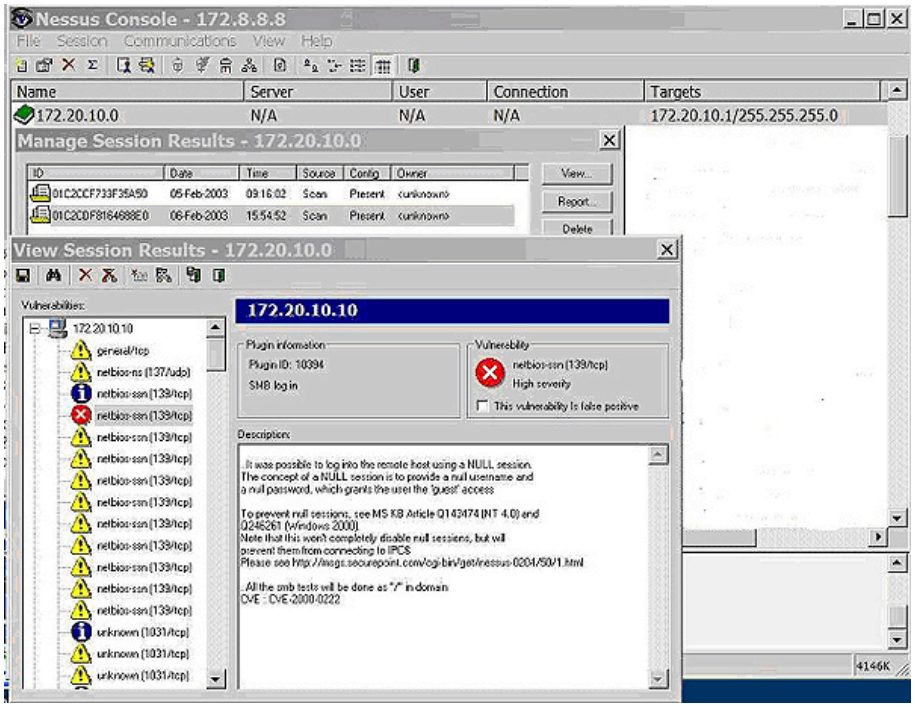

Figure2: Viewing results in NessusWX.

If a large number of hosts are being dealt with, the export functions can be very convenient. In NessusWX, results can be exported to CSV for importation to a spreadsheet, extended NSR, NSR and NBE for re-importation by any Nessus client, or into SQL files for importation into a SQL database (including MySQL). Each line item found will be exported as a single record. By importing the data into a spreadsheet or database, sorting and queries against various fields can be easily done. This gives one the ability to sort or group items by vulnerability, thus allowing one to deal with a single vulnerability at a time across many hosts. In NessusWX the scan results can be obtained by selecting the session and choosing Session | Export. In the Unix GUI, from the automatically displayed results screen, Save Report is chosen to export the data. The Unix GUI can save the data in a few more formats then NessusWX. The GUI saves in the depreciated NSR format (equivalent of the NSR format in NessusWX), as well as XML, HTML, NBE (equivalent to the extended NSR format in NessusWX), HTML, LaTeX, ASCII (similar to plain text in NessusWX) and HTML with Pies and Graphs. One difference to note between NessusWX and the Unix GUI is that NessusWX automatically saves the results when finished. The Unix GUI does not and the results will be lost upon exit if not saved.

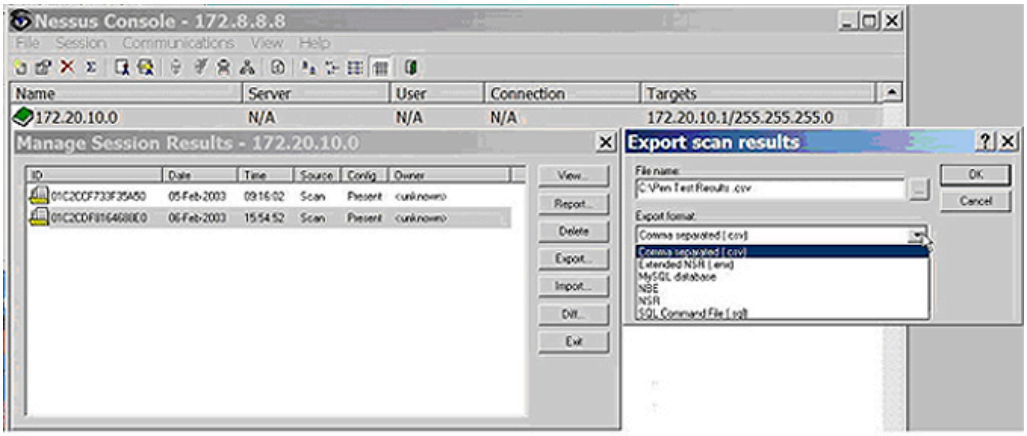

Figure 3: Image of exporting results in NessusWX.

3.0 Identifying false positives

Once the results are in a format you are prepared to deal with, the real work begins: analyzing the results and proposing solutions. In a perfect world this would be simple but due to the myriad combinations of software and configurations, remote black box vulnerability scanning is difficult and an inexact science.

One problem is that sometimes Nessus just gets it wrong. This is the case not very often but it does happen. If appropriate techniques are used (many of which are discussed in the second article of this series) the accuracy can be increased, but for the foreseeable future a human will still need to pass final judgment. Sometimes vulnerability reports can be confusing due to the many possible combinations of software and configurations involved. Although every possible way that Nessus may throw a false positive or confuse the reader can't be enumerated, many of the reasons why these things happen and the signs to look for can be discussed. Nessus

typically gives you a false positive due to one of two reasons: either the plug-in is only testing for a software version number or the results are unexpected but still valid.

Testing for a given piece of software's version number only may be desirable but can also create issues that must be reviewed. Nessus has a safe-check option which changes any dangerous (but safe-check enabled) plug-ins to just check for the version number. This is a good compromise for safety, but with the current state of software configuration management, the safe check option sometimes raises false positives. There are several causes for this. First, the software may be reporting the version number of a vulnerable version but a patch or workaround has been applied. This is frequently the case with various Linux systems. The major distributions or manufacturers such as Red Hat, IBM, etc., may patch the code and not change the version number.

Version number checking may also be misleading if the vulnerability is present but not accessible due to a workaround or best practice. A good example of a best practice bypassing a vulnerability is the IIS Unicode (MS00-078, BID 1806) vulnerability. This vulnerability will execute system commands through specially formatted URLs. By following a commonly applied best practice of placing the OS and application software on separate disk partitions, this vulnerability is rendered useless. The IIS software itself would still be a vulnerable version and ideally should be patched, but the system itself would not be vulnerable to the hole. Note that this example is not affected by the safe-check option in Nessus.

All safe-check enabled items bear a warning that they may give a false positive due to the fact that only version numbers are being checked. When this warning is displayed, often a bit of research is required to determine if the system is actually vulnerable. One approach is to check the manufacturer's release notes to see if they have fixed the problem in that version of software release. Possibly by researching the vulnerability it can be determined if the vulnerability is or isn't present on a particular OS version. In the end it simply may not be possible to conclusively determine if a system is vulnerable without access to the system, or by using a non-safe check enabled Nessus scan.

Sometimes false positives also occur due to unexpected but otherwise valid results being sent back. Nessus takes into account many possible responses but it is impossible to delineate all of them. A common example of this is the large number of web vulnerabilities (usually port 80) that are sometimes falsely reported when scanning a web-server. The issue here is that Nessus isn't receiving what it expects to receive. A plug-in testing a web vulnerability will typically expect to receive an "HTTP 404" error response if the requested page isn't found. However, although it is technically against the HTTP standard, it is becoming increasingly common for webmasters to use "custom 404 pages". These "custom 404 pages" aren't really "HTTP 404" error messages but typically a "pretty" page stating that the desired data was not found. Plug-in 10386, the "No 404 check" goes a long way to filter out these false positives by looking for common responses to page requests and correlating responses that are "custom 404s". Typically this works well. However if the web-server returns a custom page containing random or changing data, Nessus can not recognize these as a "custom 404". A telltale sign of such an issue is the large number of web-server vulnerabilities for both IIS and Apache (if the plug-ins for both web servers are being used). Verifying these is time-consuming but easy. Simply browse to the page reported to be a problem. You may have to reconstruct the problem URL from the vulnerability report. If a 404 type of custom page is displayed, it is a false positive. Otherwise the stated problem most likely does, in fact, exist.

Printers, UPSes and other hardware based devices occasionally flag a number of semi-false positives on any report. Stripped down versions of Unix/Linux and Apache are usually used for these devices and are often burned onto the device's read only memory (ROM). Of course these versions will still have whatever vulnerabilities that are later found in the software. However, since there is often little accessible memory and available resources, typically these vulnerabilities are unusable. Even if the vulnerability is exploitable, the risk may be low. Practically, how much damage can an intruder do compromising the typical printer? Printers can be identified by examining the OS discovery line item in the report.

4.0 Suspicious signs

Now that we know some of the reasons false positives may be generated, how do we recognize them? One suspicious sign is inconsistent results, especially with the reporting of a vulnerability for a software package that doesn't seem to reside on the target. Suppose a target is a identified as Windows based and yet an apparent Unix based vulnerability is found. This should raise suspicions, however the item should not be immediately written off. There could be three possible explanations: the item is false, the OS and software detection is wrong, or the problem software has been unexpectedly ported to this operating system.

While the OS and software detection in Nessus isn't perfect, it does quite well with common OSes. However, Nessus sometimes misses it when there is a variation from what is expected or when it finds an odd device. In addition, sometimes system administrators purposely change parameters specifically to confuse OS detection software. In today's computing environment more and more cross-platform development is being done, and many of today's software packages have been ported to a different OS or are used as part of other programs. Sometimes it is surprising how vulnerabilities can show up in ports across different platforms, so don't be hasty about writing off an item in Nessus' report without proper research. One example of such a surprising port is the recent system compromising OpenSSL vulnerability (BID 8732). Although usually referred to as a Unix based vulnerability, this software resides at such a low level in most packages it can also show up on Windows and Cisco environments as well. All that is required is a port of a program that uses the right version of OpenSSL.

5.0 Areas of potential confusion

False positives aren't the only issues to deal with in the report, as sometimes the correct results themselves are confusing. Confusion may stem from the result of similar vulnerability notifications. In one instance, plug-in 11424 advises against using the IIS WebDav component. This isn't a vulnerability but simply a suggested best practice. Plug-in 11412 tests for the much more famous and more dangerous, system compromising WebDav Overflow (MS03-007, BID 7116). Ironically this vulnerability has little to do with IIS and doesn't even require WebDav. WebDav was just the first of several delivery mechanisms discovered for the hole. These two items exist totally independent of each other, however in a quick reading they may be confused with each other as similar.

Another area of potential confusion is when the same named vulnerability produces different results in different environments. The Apache chunked encoding vulnerability (BID 5033) is a good example. Apache chunked encoding is actually two separate, highly related vulnerabilities. One is a DOS and one is a system compromise. The ancillary software on the target determines the result. If safe-checks are disabled, Nessus tests for this hole by sending enough junk to the server to trigger the problem. If the connection drops there is a problem. If the system compromise aspect is present, the server will start executing the uploaded data which overloaded the buffer. If the DOS aspect is present the web-server thread, which was overflowed, will crash. Nessus assumes that if the server stops responding that the vulnerability is present but it does not report if the DOS or system compromise aspect is active. If safe-checks are enabled, only the version of apache is checked.

One other area of confusion worth mentioning is the set of a few vulnerabilities to which no exploit, patch or workaround exists. These can be identified by the Nessus solution that states, "Contact your vendor for a patch." Although these vulnerabilities truly exist they are general in nature and the risk factor is low. Also, since these vulnerabilities are generally not acknowledged by the software manufacturer there is normally little that can be done to conclusively solve them

6.0 Informational items

Nessus also generates a number of informational items in its reports. Typically a report will include at least two General/TCP informational messages per host. One will document the current traceroute information to a host, and the second will make an educated guess as to the OS of the remote host. By comparing the target's fingerprint of various network-oriented parameters against a library of known values, this determination is accurate, especially for common OSes. For less common OSes usually it identifies at least the manufacturer correctly. If the fingerprint gathered does not exist in the library and the user knows the OS, the program requests the user to submit the signature and OS type to os-signatures@nessus.org. These submissions help to improve Nessus's OS detection facility for future use.

7.0 Evaluating risk

Evaluating the risk that a vulnerability might occur is another matter. Since the risk level typically determines the attention given to a problem, risk level is a important quality in the report. Nessus classifies each vulnerability as a Note, a Warning or a Hole. The risk is further identified in the plug-in summary as none, low, medium, High, Serious and Critical. Unfortunately, these classifications aren't well defined and have been subjectively applied over the years; therefore the definitions are somewhat inconsistent. There is a website available that lists some in depth statistics on Nessus's risk classifications.

If you are fixing the problems yourself, the built in classifications and risk level is probably sufficient. However if you are passing the results on for correction by others a better defined scheme would be helpful. Unfortunately, unlike many other industries there are no accepted computer security definitions for rating risk. The closest is the suggested definitions in the Open Source testing manual OSSTM (PDF document).

Another classification scheme that bears mention is the FBI/SAN's list of "The Twenty Most Critical Internet Security Vulnerabilities". This specification is valuable because it is fairly well known and can sometimes bring needed attention to a languishing problem. However, the list is fairly broad and it actually lists around 100 "Top" vulnerabilities and some of these are not as serious as ones which have been left off.

Most risk rating systems for vulnerabilities are based on risk relative to a target's operation and not the operation of the business. This simplifies the determining of risk but may be misleading to non-technical users since it is divorced from risk to business processes, which business types tend to better understand.

8.0 Finding a solution

Once there is a good list of found vulnerabilities, solving them is the ultimate goal. The person generating the audit report may also be the one in charge of remediation of problems, however this isn't always the case. Remediation is often passed off to others. Irregardless it is important for someone to know how to solve the problems that have been found. Some vulnerabilities are simple and obvious to solve, and the Nessus report will often include a link to a patch or a reference to a patch or workaround.

Sometimes a solution is a bit more elusive and doing some research becomes necessary. When available, Nessus provides some useful reference numbers. Chief among these are the Bugtraq ID (BID) and Common Vulnerability Exposure (CVE) numbers. The BID number is a reference number generated upon submission of a vulnerability to the Bugtraq security list. These are also the reference numbers in the searchable SecurityFocus vulnerability database. This database lists discussions on why various vulnerabilities work, lists published

exploits, vulnerable versions and solutions as well as a host of other data on specific vulnerabilities. This database is arguably the most complete collection of vulnerability data available.

The CVE number provides the key to the Common Vulnerability database run by Mitre. Although some vulnerability info is stored in the CVE database, its primary purpose is to differentiate and identify vulnerabilities. This is similar to a large vulnerability Rosetta stone, providing connection between various vulnerability databases. Although the CVE database is very useful, sometimes it is a bit frustrating. Many of the vulnerabilities are still in the Candidate stage, which means that their data isn't completely verified, even though they have been around for quite a while.

Once a solution has been found and applied, good practice dictates that the target be scanned again. Unfortunately sometimes patches do not update properly, workarounds do not work or open up other holes. Thus, a follow-up scan, although sometimes not possible, is certainly a good conclusion, verifying that the target is no longer vulnerable.

9.0 Conclusion

Analysis is always the tough part of any exercise. Pressing the buttons to start a scan is easy, but understanding the result is hard. Hopefully, now you understand some of the complexities of the scanning process and can start to be able to spot the significant problems out of the tangle of information Nessus can present. This the final article in the series of Nessus articles. Hopefully they have equipped you well for those long nights of scanning ahead. Happy scanning!

SecurityFocus accepts Infocus article submissions from members of the security community. Articles are published based on outstanding merit and level of technical detail. Full submission guidelines can be found at http://www.securityfocus.com/static/submissions.html.

PRINT EMAIL

Comments Mode: Threaded Go

Expand all | Post comment

ONLINE CLASSIFIEDS

Barracuda Spam and Spyware Firewall
Reclaim your Network. Stop Spam, Spyware and Viruses at the gateway. The leading solution.

Lock Down Windows Desktops - Without the Hassle
Remove admin rights and you're busy handling exceptions. Try a simpler approach. Free webinar with featured industry analyst.

Want to know your CIS security score?
The CIS has developed detailed IT security benchmarks which will help make your computer more secure. Click here to download the Belarc Advisor which will automatically show you how secure your system is compared to the CIS benchmark configurations.

Policy Management vs Vulnerability Scanning
Which is right for you? Vulnerability scanning products test for known vulnerabilities. Policy management products are pro-active by locking the doors in advance of a possible attack. Click to request our white paper.

Find IT Consulting Jobs
Access Pre-Qualified Projects from Top Businesses. Register Now!

Buy a link Now

Privacy Statement
Copyright 2005, SecurityFocus