Global Information Assurance Certification Paper

Vulnerability Analysis and the Elimination of False-Positives

By Mark Muniz
August 5, 2002

GSEC Practical Requirement v1.4

Table of Contents:

Introduction

It happens on any given day in neighborhoods across the country. The blaring and oftentimes annoying sound of the car alarm raising the decibel level to an unneeded pitch. They are triggered by anything and everything imaginable. Some are sensitive and some are not. In many cases the alarm is raised by criminals, large trucks and even planes traveling overhead, but does anyone really pay attention to them? Or are they just another noise that reminds us of the city we live in. The car alarm is a prime 1example of a false positive. [1] In IT security false positives are normally associated with vulnerability scans, intrusion detection systems and even virus scanners. For the purpose of this article false positives are the reporting of information that the vulnerability scanner believes to be present on your network when in fact it is not. Much like the car alarm that nobody pays attention to in every city, false positives are found in every tool that network security administrators use from shareware to commercial products.

Over the next few pages I will attempt to explain the idea of exactly what the role of the security analyst is in relation to the elimination of false positives and why this role is a key facet in the protection of company data. Why should false positives be identified and how does one find them in the first place so that we know what is truly vulnerable? What are false positives, and how do they relate to vulnerability analysis? How can companies ensure that they are getting the best value for their dollar while maintaining the confidentiality of their networks and ensuring that their data and valuable intellectual property is secure? These are questions that I hope to answer over the course of this article. The security analyst plays a vital role in protecting these networks which at first glance seem to be ripe for the hacking community. The time taken by the analyst in verifying the existence of the findings on the vulnerability is an excellent start to defense in depth.

Tools of the Trade

There are a number of useful tools available to the network security administrator, both commercially and from the internet. Even the tools created by the hacking community can help in finding and assessing the validity of the secure nature of the organizations network. Some of the best tools used by the IT security professionals are free. Consider the table below. According to a study by the SANS Institute in May of 2000 the top 5 most popular network vulnerability scanners included Nessus which is a freely available scanner. According to the Nessus website the goal of Nessus is "… to provide to the internet community a free, powerful, up-to-date and easy to use remote security scanner." [6]

The Five Most Popular Network Scanning Tools		
Product	Vendor	Rating*
Internet Scanner	Internet Security Systems	3.36
Nessus (free)	Nessus	3.25
CyberCop Scanner	PGP Security/Network Associates	3.09
NetRecon	Symantec (formerly Axent)	2.88
Cisco Secure Scanner (formerly NetSonar)	Cisco	2.54
May 2000, 4 point scale. 4: it far exceeded expectations, strongly recommended; 3: it met expectations, recommended; 2: it works; 1: we have stopped using it or will stop using it shortly, definitely not recommended.		

101 Security Solutions: Intrusion Detections and Vulnerability Testing Tools: What Works?
SANS, May 2000; [7]

Other tools are just as beneficial as Nessus, telnet, netcat, and nmap are a few of the tools that can greatly aid the security analyst in the identification of security exposures. Perl is also another tool which can be used to write scripts and help in testing the validity of scan data. However, for our purposes the emphasis is on scanning the network itself, the results that are given, manually verifying those results and eliminating the false positives found.

Vulnerability Scanning

Vulnerability scanners are used by many IT security professionals. One of the top rated tools according to SANS is Nessus. Vulnerability scanners like Nessus and Internet Security Scanner allow network administrators to keep a watchful eye on the network. There are also many online vulnerability scanners that allow the administrator to input an IP address and will scan your system for vulnerabilities. Though scanning for vulnerabilities in your network is definitely a good idea, the network administrator also needs to understand that ALL scanners are not created equal, but ALL scanners do report false positives of some type. For example even the best scanner cannot fully test for the presence of a buffer overflow without actually exploiting the vulnerability itself. It is also worth noting that scanners are also known to have false negatives. A false negative happens when vulnerabilities exists but are not detected by the scanner.

Within these findings some scanners reporting false positives are actually fooled by the network they are scanning. For example while attending a class on exploits and hacking techniques I used a scanner to scan the network and was given a report that said the system I was scanning was a windows machine when in fact it was actually a Red Hat Linux 6.2 install. Some machines can be made to look like something it is not. This is in the hope that it will fool the intruder into thinking that it is a different machine than it is. In recent years a whole new area has emerged from this technique. Honeypots are based on the technique of

making a machine look like something it isn't.

As stated earlier there are a number of online vulnerability assessment services that have come of age over the past year. According to an article from Network World Fusion, the leading vendors in this new arena are:

1. Qualys
2. Vigilante
3. Foundstone, and
4. McAfee

The online scanners allow network administrators to schedule a scan and have a report sent to their inbox. [8] This also tends to cut down on cost which might be a factor for some companies. How do these scans affect the false positive aspect? For instance the Foundstone scan will show you what is actually on the system before it will report it to you. This is both good and bad. If the idea of security is to see what vulnerabilities are actually present on the system being scanned then this is a good and logical approach; however, if a buffer overflow is present on the system and the scanner cannot readily detect that this is indeed a buffer overflow then this paves the way for hackers to exploit the system. Although this may eliminate some of the false positives from being in the report, this can have the reverse affect and will fall into the "false negative" category. Another aspect of the Foundstone scan is the sacrifice of detailed scan data not being reported. A good network administrator would probably want to know the detailed information of the company network. Some of the detailed information left out of the Foundstone report is general services info. [8] This can be good for brevity, but also not a great idea. If an intruder finds that telnet is running he knows that telnet allows for the submission of usernames and passwords to be passed in clear text.

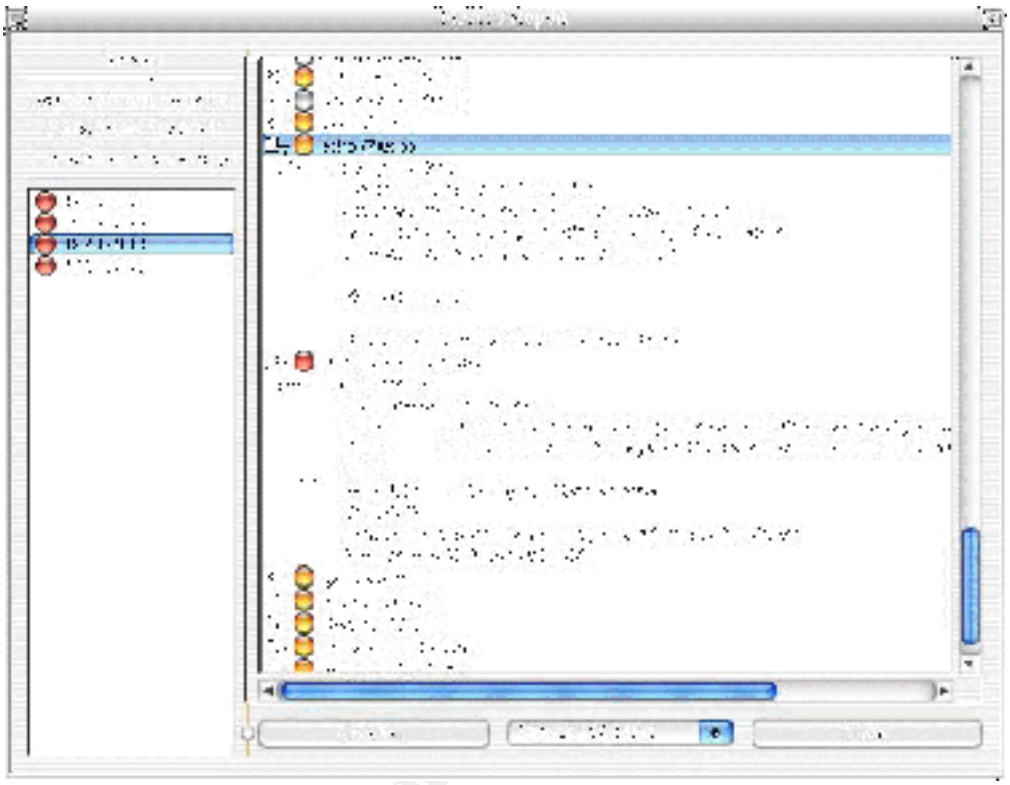
Other online scanners use some of the many open source tools that network administrators have come to love: nmap, telnet, stunnel, netcat and other tools are combined with custom built scanning tools that allow for detailed analysis of the network being scanned. Still other online scanning companies take commercial scanners and use them to do the online scans that are requested.

Analyzing the Data

Once the vulnerability scan has taken place the data should be analyzed. This alone is worth its weight in gold and is a step that should not be taken lightly. As stated earlier ALL scanners are subject to false positives. The verification of the data coming back from the scanner will likely take considerably more time than the actual scan.

Looking through the report for false positives and verifying the validity of the scan report takes on a detective mentality to some extent. When verifying

vulnerabilities many network administrators use an array of tools ranging from nmap to perl. One of the most basic of tools to use for verifying the results of the vulnerability scan is 'telnet'. For example the screenshot below



Nessus.org [6]

give an example of a typical Nessus report. Suppose that in looking at the report you know that there is an HTTP server running on port 3001.

```
[markm@localhost markm]$ telnet 10.0.0.1 3001
Trying 10.0.0.1...
Connected to localhost (10.0.0.1).
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Tue, 30 Jul 2002 21:45:00 GMT
Server: Microsoft-IIS/5.0
Content-Type: text/html
Content-Length: 57565

Connection closed by foreign host.
[markm@localhost markm]$
```

By trying to telnet to that particular port you can more accurately try to determine what service is running on that port. In this example we are given the server

banner of:

```
Server: Microsoft-IIS/5.0
```

Although this shows the HTTP server to be Internet Information Server 5.0, it would be no problem for the system admin to change the server banners to read something different like Netscape Enterprise Server, Apache or another server entirely.

The nmap tool is another good tool for use in verifying what comes back from scan reports. By using an nmap scan this allows the network administrator to see what is running on the servers in question. Below is the output from an nmap scan:

```
[markm@localhost markm]$ nmap -v -v some.server.com

Starting nmap V. 2.54BETA34 ( www.insecure.org/nmap/ )
No tcp,udp, or ICMP scantype specified, assuming vanilla tcp
connect() scan. Use -sP if you really don't want to portscan (and
just want to see what hosts are up).
Machine 10.0.0.1 MIGHT actually be listening on probe port 80
Host some.server.com (10.0.0.1) appears to be up ... good.
Initiating Connect() Scan against some.server.com (10.0.0.1)
Adding open port 443/tcp
Adding open port 515/tcp
Adding open port 80/tcp
Adding open port 25/tcp
Adding open port 22/tcp
Adding open port 3306/tcp
The Connect() Scan took 1 second to scan 1556 ports.
Interesting ports on some.server.com (10.0.0.1):
(The 1550 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
80/tcp     open        http
443/tcp    open        https
515/tcp    open        printer
3306/tcp   open        mysql

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
[markm@ localhost markm]$
```

Nmap is an excellent tool for checking the accuracy of the scan data. As stated above it is also used in some instances as the primary port scanning tool of vulnerability scanners. The definition of nmap is stated in the following:

NMAP is a multifaceted utility used to scan a range of IP addresses, identify active systems, determine which ports on those systems are

open, and identify the respective operating systems. Like all security tools it can be used defensively, by a network manager, to identify weaknesses that need to be corrected, or offensively, by an attacker, probing for vulnerabilities to exploit. [9]

The Perl language is also another extremely useful tool for network administrators, especially when coupled with netcat. When a scanner finds what it believes to be a buffer overflow it cannot fully test the vulnerability. The only real way to tell if there is a vulnerability present without intrusively exploiting the server is to look through the logs and/or core files for evidence that the service and/or the OS showed a break or an interruption in service during the time of the scan. This particular verification technique requires a bit of legwork and working with the system administrator. Many times perl scripts can help in this part of the analysis process by actually overflowing the buffer. Warning though – this technique can and will take down the service and/or the server itself. When overflowing the buffer you are in effect attacking the buffer but not sending malicious code. For example:

The IIS HTTP buffer overflow vulnerability in IIS 4.0 allowed remote attackers to cause a denial of service via a malformed request for files with .HTR, .IDC, or .STM extensions. Although positive verification could only be verified by looking for breaks in service at the server level we can tell if the buffer shows the characteristic of the buffer overflow by feeding that buffer with a perl script and netcat.

```
perl -e 'print "GET /" . "A" x 1136 . ".htr HTTP/1.0\r\n\r\n"' |
netcat target 80
```

This script will feed the IIS buffer a string of 1,136 A's. Either of two things will happen. If the buffer shows the characteristic of the overflow then it will immediately drop the connection.

```
[markm@ localhost markm]$ perl -e 'print "GET /" . "A" x 1136 .
".htr HTTP/1.0\r\n\r\n"' | netcat target 80
[markm@ localhost markm]$
```

If anything else comes back from the execution of the script and then disconnects you back to the command prompt – then you can more accurately consider this to be a false positive. However, it is still a good idea to check for service interruptions on the server itself. You can also test other services other than HTTP (port 80) with this same script.

```
markm@ localhost markm]$ perl -e 'print "GET /" . "A" x 1136 . ".htr
HTTP/1.0\r\n\r\n"' | stunnel -c -r target:443 -f
```

This particular line will test the same vulnerability but will allow you to use port 443 which is normally https.

The Effects of False Positives on Security

False positives can heavily affect the validity of the vulnerability scanning. It can also keep your system administrators and network administrators working around the clock at times. Without the elimination of false positives, the security reports would become just like the car alarm in our earlier example and be just another annoyance for the network administrator. Vulnerability scanners have the potential to show both false positives and false negatives. A report that has no analysis and verification done cannot and should not be looked upon as being absolute. False positives are one thing, but what about those false negatives? The vulnerabilities that the report doesn't show, can be even more detrimental then the findings that are shown.

Verification of false positives can sometimes be a slow and painstaking process, especially when new vulnerabilities and advisories come out on what seems like a daily basis, but the overall effect it has on keeping your network secure is much vital. Over time, historical data can be collected and can allow the network administrator to see and find problems more quickly. When one begins to analyze the vulnerability reports, it is not always a cut and dried situation. Many times the analyst will have to look at a finding more than once or twice and then look at it again in a totally different way just to make sure that what he finds is true or not. Take for instance the perl and netcat examples on the buffer overflows. When the perl script is run, you are feeding the buffer a specific amount of characters. Many times the analyst will use the same script with different values to see what if any differences may appear.

```
With 1,136 A's:
[markm@ localhost markm]$ perl -e 'print "GET /" . "A" x 1136 .
".htr HTTP/1.0\r\n\r\n"' | netcat target 80

With 100 A's:
[markm@ localhost markm]$ perl -e 'print "GET /" . "A" x 100 . ".htr
HTTP/1.0\r\n\r\n"' | netcat target 80
```

Sometimes this simple change can make the difference in determining if the finding is legitimate or not.

Hacking Statistics

Over the past several years the number of incidents and attack attempts has grown rapidly. According to CERT/CC statistics, in 1988 there were a total of 6 incidents reported, while in 2001 there were a total of 52,658 reported.

"A computer security incident, [...], is any adverse event whereby some aspect of computer security could be threatened: loss of data confidentiality, disruption of data or system integrity, or disruption or denial of availability." [4]

Number of incidents reported

1988-1989

Year	1988	1989
Incidents	6	132

1990-1999

Year	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999*
Incidents	252	406	773	1,334	2,340	2,412	2,573	2,134	3,734	9,859

2000-2002

Year	2000	2001	Q1-Q2,2002
Incidents	21,756	52,658	43,136

Total incidents reported (1988-Q2,2002): **143,505**

CERT/CC Statistics 1988-2002 [3]

In relation to vulnerability analysis, the number of vulnerabilities reported has also grown much like the number of incidents. In 1995 the number of vulnerabilities reported was approximately 171 while in 2001 the number went to 2,437. At first this does not appear to be as bad, but when you think that with each vulnerability there are likely many hundreds of hackers who use these vulnerabilities to create the incidents.

With this rise in hacking that we have a flood of tools and scanners available for the network administrators to use in verifying that what is shown on vulnerability reports is true or not.

Vulnerabilities reported

1995-1999

Year	1995	1996	1997	1998	1999*
Vulnerabilities	171	345	311	262	417

2000-2002

Year	2000	2001	Q1-Q2,2002
Vulnerabilities	1,090	2,437	2,148

Total vulnerabilities reported (1995-Q2,2002): **7,181**

CERT/CC Statistics 1988-2002 [3]

According to the Honeynet.org site, it takes on average about 72 hours for a Red

Hat Linux system to be scanned and attacked. With the shortest time being approximately 15 minutes for the same system to be scanned, probed and compromised. [2] For a default installation of Windows 98 the time was around 24 hours before the system was compromised. [2] With this in mind it is no wonder that vulnerability analysis coupled with the elimination of false positives plays a crucial role in determining how these systems were scanned and eventually compromised

At first glance, the statistics are rather bleak; however, it is clear that the information security community needs to scan and search for holes within the data along with doing careful analysis of the findings that occur. Like the car alarm example posed earlier the security professional who gets the alarm but ignores that alarm without looking into it is asking for trouble. For the hacker this is like opening the doors and saying "Here I am - come on in!" No one tool or even a multitude of tools can thoroughly scan your system and eliminate the need to eliminate false positives. There are too many variables involved. Just changing the server banner can lead one to think that the system scanned is a windows system when in reality it is a Sun box. With careful analysis and good verification processes and procedures in place, the findings from the vulnerability scans will yield useful information on what is really there.

Protecting the whole network

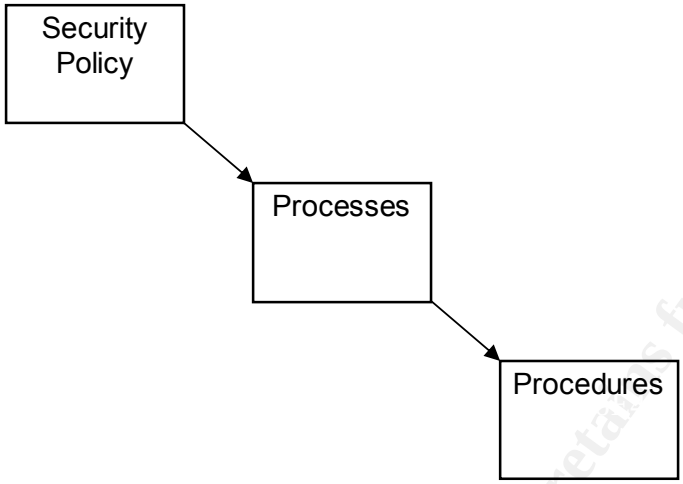
The security of any organization let alone the elimination of false positives begins first and foremost with the basics.

```
"Deny all, Allow as needed"!
```

This sentence should be the first line in the organization's security policy. In the fight against the "Internet Threat" an organization must have a blueprint to follow. For example, a security policy is the starting point, but along with the policy the organization should have a documented set of processes and procedures in place to further explain what needs to take place in order for the organization to have a secure network.

Security Policy - First Things First

If no policy exists then the second line should read: "Create a security policy". A good security policy is a key element in achieving an effective security barrier to your organizations networks. [5] The security policy can help define what type of tools are needed and used to find vulnerable systems on the network. The policy itself will not explain the process or the procedures you need to follow in order to eliminate false positives, but it will explain what will and will not be allowed on the organizations network. In essence there is a drill down effect starting with the policy then the processes and finally the procedures that will be taken to protect the data.



Relationship of security policy to processes and procedures

A good example of what the security policy does is to specifically define what parts of the network will be available to whom. If the accounting department does not need access to data in the engineering department then why give it to them? This type of specifics will aid when the scan data shows irregularities. It will also aid the network administrator in determining what type of software will be needed on different systems within different departments.

Conclusion

After spending some time and analyzing the reports given by today's vulnerability scanners it becomes apparent that not all scanners are created equal. You can't take the report by itself without some degree of uncertainty in the findings that show on the report. Nessus, ISS, Cybercop and others may be good, but there is no equal to actually looking through the report and verifying the validity of the findings. The first time one actually sits down and goes through the report to verify the findings you will think that there is no end in sight and there is no way that you will ever be able to accurately verify all of them. Identifying and removing false positives from a scan report is not rocket science, but it does take quite a bit more time than just pushing a button and running the scan. Eliminating false positives is not "rocket science", but with patience and a bit of time it is a great help in keeping the hackers out of your network. Unlike the car alarm that seems to have little or no effect on those around it, we need to fully understand the idea behind false positives and how to eliminate them in order to protect not only the company assets, but many times our job as well.

References

1 ITworld.com, "The Effect of Crying Wolf", IDG.net, October 25, 2001; http://www.itworld.com/nl/unix_sec/10252001/

2 The Honeynet Project, "Know Your Enemy: Statistics", Honeynet.org, July 22, 2001; http://project.honeynet.org/papers/stats/

3 "CERT/CC Statistics 1988-2002", CERT.org, April 5, 2002; http://www.cert.org/stats/cert_stats.html

4 John P. Wack (NIST, US). - US National Institute of Standards and Technology. - Gaithersburg, Md. - NIST Special Publication 800-3. - November 1991. - 17 ref. - 39 p.

5 Sans Security Essentials, "Basic Security Policy", General Security Essentials - Day II, November 1, 2000; http://www.sans.org

6 The Nessus Project, August 5, 2002; http://www.nessus.org

7 101 Security Solutions, “Intrusion Detection and Vulnerability Testing Tools: What Works?”, 101com.com, July 9, 2002; http://www.101com.com/solutions/security/article.asp?articleid=569

8 Network World Fusion, ”Vulnerability Assessment Services on the Rise”, nwfusion.com, July 17, 2002; http://www.nwfusion.com/reviews/2002/0204bgside.html

9 SANS Reading Room, “An Introduction to NMAP”, rr.sans.org, October 25, 2001; http://rr.sans.org/audit/nmap2.php