Global Information Assurance Certification Paper

A MODEL FOR PEER VULNERABILITY ASSESSMENT

INTRODUCTION

Once a network is set up and running, it is critically important to persistently check the network and hosts to assure they are not vulnerable to attack. Once a system goes up, it becomes a target for a worldwide community of hackers – hackers of all skill levels. And due to the graphical interfaces and scripts available for hacking tools, it doesn't even take a great deal of skill to find and exploit vulnerabilities. Continuous assessment is necessary to maintain security. "Routine, independent reviews of security systems and procedures not only ensure an organization has adequate protections in place, but confirm that they are working as designed – and that the employees are using them effectively." [12]

So vigilance on the part of system administrators or independent testers is essential to maintaining security. Some methods for maintaining this vigilance are audits, penetration tests and self-assessment. Often system administrators perform the audits and tests. However, many times administrators may not have time, resources or skills to perform them.

One option is to purchase tools and services for testing. Often third party experts are brought in to perform the audits or tests. However, with the availability of freeware tools, there is opportunity for in-house expertise with these freeware tools to provide a high level of defense. In fact, "At the RSA Data Security Conference in San Francisco earlier this month, some of the most popular sessions focused on the free technologies that can help companies find holes in their network, keeping intruders and viruses out and sensitive data in. The upshot: While some software packages cost tens of thousands of dollars, you needn't think you have to break the bank to check your network security. " [1]

An additional way to help ensure security is to train those responsible for maintaining the security of the systems. It's been suggested that the only solution to the security dilemma is education, including lab-based exercises, practice, and attacking your own site looking for weaknesses. [9] Having worked in security for some time, it surprises me how much we as security professionals know about defense but how little we sometimes know about what hackers are doing. Most of us don't have time to explore hacking techniques; we have systems to run. And it seems that in classes or demonstrations, whenever the ease and flexibility of some of the attack tools are demonstrated, mouths drop open and interest is definitely piqued.

So while some situations clearly may require bringing in highly skilled resources to test systems, a sound basis for good security would be to develop in-house expertise in vulnerability testing by the system administrators, and to develop an effective method of performing testing.

The challenge for this effort, then, is to combine freeware tools with a methodology for using them that effectively promotes persistent security. This paper proposes a model

for ongoing assessment to be performed by the system administrators that includes testing and assessment in a non-threatening environment that provides added value of education for those performing the assessments. We will first examine existing methods of assessment, make the case for a peer assessment, explore the goals and benefits of a peer assessment, and outline a generic assessment model.

EXISTING METHODS

Auditing is generally a method for taking a set of standards or policies and evaluating a system against that standard. The result of auditing is typically a report enumerating areas where the existing systems did not meet expected standards. This report can often be extensive, voluminous and daunting to address. The source of the standard to which the audit is performed can vary from internal policies to best practices: a determination should be made as to the validity of the standards for the audit. The limitation here is that unless a vulnerability is specified in the standard, it may not be found.

Of all the techniques discussed here, penetration testing is probably the most glamorous. Penetration testing involves actually exploiting vulnerabilities: the only goal is to compromise security. [14] Penetration testing can be covert (the attacked does not know the test is to occur) or overt. The tests can also be "zero-knowledge" (where the tester knows nothing in advance about the tested systems) or "full-knowledge".

Penetration testing does not try to identify all vulnerabilities, but to prove a system can be compromised. It can be an excellent tool for proving the need for security. When managers see sensitive data compromised, the need for security often becomes clear. Also, penetration testing may be a tool for identifying security policies that are not being followed. While there are situations well suited for this type of technique, it is perhaps not the best choice for continuous, robust defense since it typically will not find all vulnerabilities.

Assessments can be defined as "an overt study to locate vulnerabilities". [14] Assessment differs from auditing in that there is not a specific standard against which systems are measured. The goal of assessment is to identify areas that need improvement. More than a checklist, the assessment addresses the extent of vulnerability and recommendations. The scope of the assessment can depend on which components you want assessed. This particular method will serve as a basis for our model. The initial example will not target a specific system or technology but serve as a general toolkit and a blueprint for future evolvement into more specific assessment guidelines.

A MODEL FOR PEER AUDIT

The proposed model involves regular peer assessments that incorporate elements of all of the above methodologies. It also incorporates an element of education for the professionals taking on the assessments.

The primary goal of the peer assessment is to, on an ongoing basis, identify and fix

vulnerabilities in systems. As a secondary goal, participants have a chance to learn more about their vulnerabilities. System administrators often are tasked with patching known vulnerabilities, but running an ".exe" patch doesn't exactly give you a wealth of information on how that vulnerability weakens your system. In this model, participants get a chance to play on both teams. As an attacker, they get a chance to learn the very techniques and tools that will they are trying to protect against. And as the attacked, they get a chance to see exactly what legitimate attacks might look like in their systems. This gives them a better understanding of what to look for in their intrusion detection activities.

This model also hopefully provides a more interesting environment in which to learn; participants get a chance to see first-hand what an attack looks like. And a little healthy competition could serve to make the process interesting, especially if there is no punishment associated with identifying vulnerabilities. As one GIAC security professional said, "If anything, these individuals will be overworked but believe me, they will love every minute of it. Where else would they have a chance to be the good guy, the bad guy, the expert technician and the auditor all rolled into one?" [6] In this collaboration framework, you get a chance to see both the attack and the result. You learn not only what attacks to expect but also what to look for. Like some penetration tests, it "provides a great opportunity to gain experience in a consequence-free exercise." [8]

An important part of this peer assessment model is the result reporting. The analyses of the results of the test are performed by the two teams. Results are discussed only among them, although extra recognition could be awarded the teams that report and publish helpful analysis and results, or "lessons learned", for future teams. However, an important element of this model is that *the only report that flows to upper management is that the assessment was completed successfully and when*. This is an important paradigm shift for most current assessment and reporting. The goal of this model is to foster healthy competition between teams performing the same function, the let the teams learn their craft, and discouraging the "seek infractions and punish" model of many audits.

Of course, there are inherent risks in running the type of assessment proposed here. The activities may result in captured passwords or sensitive data. Malicious or careless use of these types of tools can pose an enormous risk and cause harm. Additionally, the tools used may be fine and even fun in isolated test situations, but could have unexpected consequences on production environments. [4] For this reason the higher "exploitative" levels of this model should ideally utilize carefully scripted "prize" exploits where possible.

This seems an appropriate place to address the perception that actually exploiting vulnerabilities and hacking into systems is the realm of "hackers", and that these activities should not be in the realm of security professionals. The view is that that security professionals have no business trafficking in the tools and techniques that hackers use. It isn't clear where this perception comes from; perhaps out of the

brouhaha about hiring hackers, and there is some validity to the argument that the media may be instrumental in perpetuating this view. "Instead of stating how the hacks were easily prevented, they portray the criminals as geniuses, because it makes a better story and is easier to understand." [15] This may also be why there seems to be a general feeling that hackers are far more technically astute than the security professionals who are responsible for systems; a mistaken notion that hackers bring some special skills and experience that aren't available anywhere else. [7]

In fact it often does not take talent to perform many of the hacks – only a familiarity with the automated tools. And let's not forget that the hackers don't have to go to budget meetings or prepare and update monthly progress reports for a bazillion projects; instead they can focus on learning to hack. However, the perception seems to persist. "Sadly, even many security professionals don't realize that it takes very little skill to break into most computers." [15]

So while it may be true that penetration testing and vulnerability exploitation should be done by experienced professionals, should we worry about testers training in-house professionals in the skills and techniques needed for malicious attacks? [8] On the contrary, it is an assumption of this model that the more security professionals know about the tools being used against them, the better. And this model is a means to get them that knowledge.

So now we'll take a look at an outline for performing a peer assessment. While it is beyond the scope of this paper to lay out a detailed roadmap for specific assessments, we will go through the exercise of a preliminary general assessment for the most well known vulnerabilities.

THE PEER ASSESSMENT PROCESS

Periodically all system administrators would be required to participate in a peer assessment team. The teams would consist of an attacker team and an attacked team. The teams follow guidelines as to what tests/tools they will be using. The attack time frame will be known, but the exact time of the attack should be unknown by the attacked in order to allow them to try to identify the attack.

The model contains four distinct activities.

Initial scheduling and preparation During this phase the attacker and attacked are identified and a time frame for the assessment is defined. The time frame should include some preparation time for the attacked to implement any lacking modifications or fixes before the attack is launched.

Initial Attack During this phase the attacker uses the prepared guidelines to reach pre-defined goals. Meanwhile, the attacked should be monitoring systems to identify the attack.

Analysis During this phase the teams collaborate to analyze the attack tactics and

discuss the attacks. A determination is made as to what degree of success each level attained. Also discussed would be lessons learned, enhancements to the defenses against the attacks, and enhancements to the attacks that might be included in future assessments.

Optional Repeat of Attacks After fixes have been found and implemented, or enhancements have been made to the attack defenses, the same attacks could be repeated to determine is the proposed defenses were effective. Enhancements to the attack plans should be deferred to either a fifth re-attack phase or the next assessment.

Of course, care must be taken in setting up the assessment activities. Machines that don't belong to the company should never be scanned. Also, all addresses used should be double checked in order to avoid a typo that could result in scanning the wrong computer. [1]

How far should the assessment go? Stop at vulnerability identification or follow through to exploitation? If the activities are carefully planned, certain exploit "prizes" could be attained without disruption of production activities. With review, development and refinement of the assessment activities by an internal team, this should be a manageable risk.

THE ASSESSMENT ATTACK

We'll section the actual performance of the assessment attack into four levels. We're patterning this after two sources. The first source is a methodology discussed in an article on penetration testing [8]. This methodology proposed four steps: 1) discovery, 2) enumeration, 3) vulnerability mapping and 4) exploitation.

As some validation for this methodology (and just for fun), we also reference a second source: a hacker tutorial from a hacker "newbie" web site. [10] As a preview of things to come, this tutorial includes instructions for the following steps:

- *Use false identification – port your connection through someone else if possible, or work anonymously.*
- *Gather information about your target.*
- *"Case the joint". Capture web addresses, emails. Try to bounce emails. Try traceroute to identify servers at the site. Use Whois. "Give 'em the finger" to find out names of accounts.*
- *Port scan for open ports. Attempt stealth-scans if possible. Port scanning "… practically screams to the webmaster's of the victim site that they are in the middle of being hacked."*
- *Telnet to try to find out the server type. "You need to know the server type to have any hope of hacking the thing."*
- *Now try to get through the front door. Easy thing first: try root/root. It might work. Try known passwords. Try passwords that match user names. It's lame, but "about 1 in 20 people are stupid enough to have the same login name and*

password."

- *If this fails, try to get the password file. Sometimes you can get it using finger. If you get it, run a password cracker like Cracker Jack or John the Ripper on it.*
- *Get Linux and learn to use it.*

So we now have an outline to define our processes and a bit of validation that this is the type of attack process we can expect.

For each level of our preliminary attack guide we'll identify the activities, some goals or "prizes" for both the attacker and attacked for the level, and a suggestion of tools that could be used for the level activities. While it is beyond the scope of this paper to provide instructions on how to run the tools, there should be no problem finding information about them on the web as they are among the favorite freeware tools. References for learning more about some of them are also provided at the end of this paper.

A word about tools . . . selection of tools depends on the goals of the assessment. Also, no distinction is made in this paper between tools for Unix and tools for Windows systems. When reviewing tools and developing a specific guideline, care will need to be taken that the right tools are selected.

Level One – Discovery

Peer assessment conducted by company personnel will not be, by their very nature, zero-knowledge attacks. However, this does not mean that the discovery exercise isn't an important part of the assessment process. It can also serve to help baseline the attacked systems.

A part of what is typically known as "footprinting", the purpose at this level is to gain as much information as possible about the system. Typically footprinting includes discovering "… IP addresses, operating system types, port numbers and services, phone numbers either to dialups to internal networks or just a voice mail system, and even some times you will find logins on a website while they may not be for the system your looking for …. Routing tables are also good because they will let you know what other IP ranges you need to scan and what routers control information between internal and external systems. E-mail addresses and a simple link to another site can all aid you in exploiting the target" [13]. These tasks are found in both Level One and Level Two, but for our purposes we distinguish between pure discovery of systems and the later gathering of information about those systems. In this first level of intrusiveness you want to develop a map of addresses worth looking into during the next round.

Initial activities include **Ping**ing to discover active IP addresses, although Ping sweeps may reveal very little about systems since they may be going across a firewall or router… reply packets won't even contain the MAC address of the target systems, but the router's MAC address instead. [16]

After pinging, we can use tools such as **Whois**, **SAM Spade** or **finger** to learn more

about those addresses. An **email bounce** may give us additional information (sending an email to the domain name under a bogus email name that will bounce back to you). **Finger** may also give us some account names that could come in useful later on. We'll want to make a note of any email addresses or web addresses for the site that can be exploited later. We might also use **traceroute** to determine other servers at the target.

Level One Prizes:
Attacker: lists of servers, owners, emails, web addresses discovered with the Level One tools
Attacked: evidence that the attacker was in discovery phase: logs of pings or use of finger service.

Level Two – Enumeration
Compiling a list of live systems is the beginning. Now we want to know the services that are running and the operating systems in use. [16] In this phase we begin to target interesting addresses and attempt to find out what operating systems and what open ports and services we can find.

Using **nmap** will allow us to scan the networks to find which servers are active and which services they offer. Nmap is a very popular tool and is well recognized as a great tool for port scanning and operating system identification.

We should learn to perform several types of scans, and while using them learn quite a lot about the TCP behavior. Some of the types of scans we would want to perform include:

- Vanilla TCP connect() scanning,
- TCP SYN (half open) scanning,
- TCP FIN (stealth) scanning,
- TCP ftp proxy (bounce attack) scanning,
- SYN/FIN scanning using fragments
- UDP recvfrom() scanning,
- UDP raw ICMP port unreachable scanning,
- UDP recvfrom() scanning,
- ICMP scanning (ping-sweep), and
- Reverse-ident scanning.

See the Fyodor text "The Art of Port Scanning" [5] for details on these types of scans.

Additional useful tools for scanning are Foundstone's **SuperScan** and **Fscan**.

Winfingerprint can be used to query information on Microsoft machines by using Windows null sessions without any logon credentials. We might also be able to use **winfo.exe** to create a null session and dump user accounts.

Level Two Prizes:

Attacker: Lists of open ports and services running on them. Lists of live servers behind firewalls. Port scans that went undetected.
Attacked: Logs of port scans. Blocked scans. Absence of services that are not necessary but could have provided information.

Level Three – Vulnerability Identification

Once we have identified systems and have also identified operating systems and processes on ports, we want to look for exploitable vulnerabilities on those target systems.

One essential tool for this level is **Nessus**. This will be a critical tool for this phase of the assessment. Nessus is an incredibly versatile and extremely efficient application that not only identifies vulnerabilities that could be exploited, but tells how to prevent them. [3]

For evaluating web security, **Whisker** is a Perl program that scans Web server to see if they are running any of 200+ interactive Web programs with known security vulnerabilities.

Nmap can also be used in this phase since the very existence of some services may reflect an exploitable vulnerability (e.g., telnet, ftp).

Level Three Prizes:
Attacker: Reported existing vulnerabilities
Attacked: Lack of reported vulnerabilities, logs of vulnerability scans

Level Four – Exploitation

The main goal in many attacks is to gain access to a host machine via an exploitable vulnerability. However, actually exploiting the exposed vulnerabilities will be more problematic and more difficult than finding it. "The difference between identifying potential vulnerabilities and gaining interactive remote access to hosts requires a quantum leap in skill level." [8] You may have noticed that in our newbie hacker tutorial, the instructions pretty much stopped at this point with "get Unix and learn it." A determination should be made for each assessment whether to move into this level.

Of course, having said that, it should be noted that some assessment tools, specifically Nessus, have the capability to implement attacks with the click of a button. Therefore, if you are assessing production systems at this level, you should carefully control activities in this phase. Ideally, a set of specific benign actions should be developed that can prove the access was gained (e.g., placing a specific file or graphic in a specific place) yet not disrupt services. The initial suggestion is to only move into this level with known, tested benign activities.

We can again use the **Nessus** tool to proceed in this level with actual exploit implementation. Other techniques include gaining access to shared resources by

gaining **NetBIOS** access. **Netcat** is another tool we would employ at this stage. Netcat is a tool to allow raw socket connections to ports, over which data could be sent and received. One type of attack using netcat would be to locate an IIS server that is down (or cause one to go down) and then to use netcat to bind cmd.exe to port 80 and then telnet to the web server over port 80 [11].

Actually grabbing a password file and cracking the passwords with tools like **John the Ripper** would constitute a prize exploit.

Exploits can also attack specific applications on the host machines. Some of the top techniques used to attack web applications include [2]:

- Cookie poisoning
- Hidden-field manipulation
- Parameter tampering (changing information in a site's URL parameter)
- Buffer overflow
- Cross-site scripting
- Backdoor and debug options
- Forceful browsing (subverting the application flow)
- Stealth commanding (running unauthorized code on the host)
- Third-party misconfiguration
- Known vulnerabilities

Level Three Prizes:
Attacker: purloined passwords, placing benign files on target, proof of access to unauthorized resources (files, etc)
Attacked: Lack of successful exploits. Maintenance of critical systems with no unauthorized access.

NEXT STEPS

It should be apparent by now that there could be very specific guidelines developed for peer vulnerability assessment of specific systems or applications, perhaps with some very interesting and entertaining benign prize exploits defined. This paper has proposed a demonstration of the model with a very elementary approach to finding vulnerabilities. But specific guides could be developed for specific types of assessments. Hopefully this model will lead to further development of these guidelines in specific areas.

SUMMARY

Assessments are a good tool for ongoing security. This model for peer assessments provides a non-threatening, interesting format for security professionals to be allowed the opportunity to learn both sides of an attack and to practice detection and defense. In addition, the analysis phase gives participants the opportunity to update and enhance the assessment guidelines themselves, thus helping to keep them up to date and relevant to the local systems. Major differences between this model and existing methodologies are that:

- Participants alternately play the role of attacker and attacked, learning both the defenses against attacks and the tools and techniques of the attacks
- Analysis of the assessment is performed in a non-threatening competitive environment
- Participants can enhance the guidelines for assessment based on lessons learned and/or new knowledge, thus keeping the guidelines updated

Future enhancements to the guideline provided here would include more detailed attack instructions and development of guidelines specific to certain system architectures, applications and technologies.

REFERENCES

[1] Abreu, Elinor. "No Budget? That's No Excuse for Not Testing Your Network Security". April 25, 2001. The Industry Standard. URL: http://www.thestandard.com/article/0,1902,23991,00.html (17 December, 2001)

[2] Bar-Gad, Izhar. "Identifying the 10 Most Common Application-Level Hacker Attacks". September 17, 2001. Yahoo! India Technology. URL: http://in.tech.yahoo.com/010917/22/14vxo.html (17 December, 2001)

[3] Christensen, Paul. "An Introduction to Nessus". May 7, 2001. LinuxSecurity.com URL: http://www.linuxsecurity.com/feature_stories/feature_story-86.html (17 December, 2001)

[4] De Beaupre, Adrien. "Know yourself: Vulnerability Assessments". June 21, 2001. SANS Institute Information Security Reading Room. URL: http://www.sans.org/infosecFAQ/audit/know.htm (17 December, 2001)

[5] Fyodor. "The Art of Port Scanning". September 6, 1997. Insecure.org URL: http://www.insecure.org/nmap/nmap_doc.html (17, December 2001)

[6] Herman, Ben. "Routine External and Internal 'Hacking', An Important Part of Information Assurance". April 19, 2001. SANS Institute Information Security Reading Room. URL: http://www.sans.org/infosecFAQ/attack/routine.htm (17 December, 2001)

[7] Koerner, Brendan I. "Showdown at Hacker Gulch". June, 2001. Business 2.0 URL: http://www.business2.com/articles/mag/0,1640,14779,FF.html (17 December, 2001)

[8] Kurtz, George & Chris Prosise. "Penetration Testing Exposed". September, 2000. Information Security Magazine. URL: http://www.infosecuritymag.com/articles/september00/features3.shtml (17 December, 2001)

[9] McClure, Stuart. "Digital Battlefield". 2001. Foundstone URL: http://www.foundstone.com/cgi-bin/display.cgi/?Content_ID=180 (17 December, 2001)

[10] Overlord. "Beginners 'Step By Step' Security Guide, V0.1.32". June 1998. The Newbies Area Hacking Tutorial. URL: http://www.thenewbiesarea.com/tutorial.shtml (17 December, 2001)

[11] Shipley, Greg. "Tools From the Underground". May 29, 2000. Network Computing URL: http://www.networkcomputing.com/1110/1110ws1.html (17 December, 2001)

[12] Swanson, Dan. "Secure Strategies". October, 2000. Information Security Magazine. URL: http://www.infosecuritymag.com/articles/october00/features3.shtml (17 December, 2001)

[13] Tag. "Footprinting FAQ v0.1". 2000. Packetstorm. URL: http://packetstorm.decepticons.org/papers/general/Footprinting-faq-v0_1.txt (17 December, 2001)

[14] Winkler, Ira. "Audits, Assessments & Tests (Oh, My). July, 2000 Information Security Magazine URL: http://www.infosecuritymag.com/articles/july00/features4.shtml (17 December, 2001)

[15] Winkler, Ira. "Security Strategies for E-Companies: A Crisis of Confidence". February, 2000. Information Security Magazine URL: http://www.infosecuritymag.com/articles/february00/columns_logoff.shtml (17 December, 2001)

[16] Unix Insider. "Tapping on the Walls". November 17, 2000. ITworld.com URL: http://www.itworld.com/Comp/1423/swol-1117-buildingblocks/ (17 December, 2001)

Other Sources of Information

Bahadur, Gary. "Freeware Security Web Tools". October, 2001. Unix Review .com URL: http://www.unixreview.com/documents/s=1442/urm0110o/ (17 December, 2001)

Goering, Trevor. "Building a Security Toolkit". June 25, 2001. SANS Institute Information Security Reading Room. URL: http://www.sans.org/infosecFAQ/audit/toolkit.htm (17 December, 2001)

Kimber, Lee. "Tricks of the Cracker's Trade". January 31, 2000. ZDNet UK . URL: http://www.zdnet.co.uk/itweek/analysis/2000/04/client/01.html (17 December, 2001)

Lujambio, Danilo. "Learning with nmap" June 29, 2001. LinuxFocus.org URL: http://linuxfocus.org/English/July2001/article170.shtml (17 December, 2001)

StartX. "Tutorials – NetBIOS". Newchan URL: http://www.remisesba.com.ar/Newchan/Tutorials/NetBIOS.asp (17 December, 2001)

[17] Zenomorph. “Fingerprinting Port 80 Attacks: A Look Into Web Server, and Web Application Attack Signatures”. November, 2001. www.cgisecurity.com URL: http://www.cgisecurity.com/papers/fingerprint-port80.txt (17 December, 2001)