

---

# Introduction to Network Penetration Testing

James Shewmaker  
jims@bluenotch.com

Intro to Penetration Testing - © 2009 James Shewmaker

Welcome to Penetration Testing Methodology. Today we will talk about the methods and concepts of Penetration Testing. We will also discuss several case studies as scenarios.

This course is designed for the technical audience that has some general experience with the systems being tested. After we cover some of the techniques, we will get a taste for penetration testing and assessments. By the end of this material, you will be familiar with why and how to conduct a penetration test.

## Outline

---

- Definitions and Concepts
- Key Tools
- Targets and Scenarios

In order to understand the fundamental issues with pentesting we will start with the most common definitions and concepts. The tools section will show us the most popular tools used in pentesting. Without targets, penetration would be random testing, so we will discuss different kinds of targets and look at some sample situations.

Penetration testing is pointless without documenting findings and fixing issues, so we will also cover what to do at the end of a test.

## What is a Penetration Test?

- A penetration test (pentest) is a systematic probing of a system
  - A system could be any combination of applications, hosts, or networks
  - Emphasis on how deep one can get into the system
  - Sometimes confused with Audits or Assessments

Intro to Penetration Testing - © 2009 James Shewmaker

3

A penetration test's sole purpose is to see the extent of how deep into a system one can get. A system might include an application or a group of desktops. A system may also be a logical system that might include all things IT in a building or even how the IT staff responds to an incident.

Because such a system is usually complex, it is very difficult to conduct a thorough test without precisely detailing every bit of information seen and how it was acquired. Not all of this information will be necessary in the final report, but some vulnerabilities are not immediately obvious. Since a system may also be quite large, it is essential to highlight the critical aspects of each vulnerability.

Easier to reach vulnerabilities, often referred to as “low hanging fruit,” may be critical just because they are facing the public. Business critical system components would also be items to highlight in the evaluation report.

What distinguishes a penetration test from any standard test is that a penetration test is designed to see how deep past security layers one might be able to get. Most good penetration testers will employ a great deal of creativity to test all areas of a system.

## Testing Areas

- What areas can we test?
  - Response / Work flow / Policy
  - Physical
  - Logical
    - Network
    - Host
    - Application

There are several areas we can test in any system. Areas with strong human components are incident response, workflow, and policies. This means we can manipulate them on a social level. This could be anything from user tendencies to manipulating the workflow of a process, such as forging an email from the system administrator that instructs the user to run a command.

Physical access to a system also should be tested. Say someone could cycle power on a host, insert a bootable CD and boot to it. This usually allows the hard drive to be read, which could bypass any password on the harddrive.

When people think of a penetration test, they usually think of testing a network, host, application, or some combination of them. Usually this entails a very technical approach with specialized tools.

## Why are We Testing Anyway?

- How do you **KNOW** your network and systems are secure?
  - Your knowledge is only as good as your last test
  - Your last test is only as good as your weakest link
    - Tools
    - Experience
    - Execution

There are two types of computers that do work, those that are vulnerable and those that will be. One of the inherent problems with testing any system is that the system changes when stimulated or naturally by itself. A test can only evaluate a system during the course of that test. Once the test is over, the system may be changed by the owners, users, or its environment. Now that the system has been modified it may behave differently and could reveal new information or vulnerabilities.

A test is only as good as the tools, experience, and execution used. Since the system changes as it is used, it must be consistently re-evaluated. Setting up a policy and schedule for tests in the future is necessary to keep a good grasp of where the vulnerabilities in a system are.

## Authoritative Permission

- Permission must be
  - From the proper authority
  - Very specific in detail
- Unfortunate example
  - Oregon vs. Randal Schwartz, 1995
    - Mr. Schwartz used a back-door and cracked passwords in the course of his work without explicit permission

Authoritative permission is extremely important and should never be overlooked. The permission must precisely give the authority to test within this scope. This permission must also be from somebody who has the authority to give such permission as well.

The classic example of the permission issue is Oregon vs. Randal Schwartz. Mr. Schwartz used a back-door program and cracked passwords. This case has become the cautionary tale for any programmer or administrator. A penetration test MUST have proper and complete permission specifically authorizing the test on the system. Anything less than written permission is a very bad legal incident waiting to happen.

## Pentest Targets / Scope

- Scope is WHAT to test
- Highlight points of Interest or Value
  - Hosts
    - Service
    - Application
  - Network
    - Internal devices
    - Perimeter devices
- Where is the low hanging fruit?
- How deep can we go?

To be useful ultimately in defense, a penetration test must have designated targets. These targets could include the employment records, e-commerce database, or a public web server. There are usually interesting items besides the targets that provide key information or access, such as a firewall or syslog server.

One of the most valuable things penetration testers can find is a small chink in the armor that can lead to a target. These “low hanging fruit” are usually easy to reach and manipulate. The low hanging fruit is most likely what a casual attacker would find in reality. That is not to say that more complicated attacks are not also in reality; they are less frequent. Most casual attacks (such as a random network scan) may lead to the easy targets, while a targeted attack is likely to continue when there is a lack of easy targets.

Remember that the point of penetration testing is to see how deep we can get into the system. Avoid the assumption that an attacker wouldn't think to leverage something; you need to test ALL avenues and attack vectors. Often getting past one hurdle, new information could lead to a deeper attack. We refer to such an indirect move as a lateral move since it does not progress directly toward the target.

## Pentest Goals

- Determine target discoverability
- Assess state of Incident Response
  - Technical skill assessment
  - Policy and procedure practice
- Document unknown or orphan resources

One of the most common goals of a pentest is to find out which hosts are public facing. A pentest could uncover a misconfigured firewall in this case.

Besides the technical pentest, an organization could want to test it's incident handling skills and policy. So the penetration test's goals would include measuring the incident handling response and how closely company procedures were followed.

The most dangerous items to find in pentest are misconfigured or unknown resources. Rogue access points are a commonly found unknown resource in today's networks. These unknown resources are not only a liability because of a lack of coordinated management but they also demonstrate that either a policy or procedure was not followed correctly.

## Rules of Engagement

- Establish the HOW of the pentest
- Decide when to begin and end
  - Set a Date and Time window
  - Don't start at the first opportunity
- Build the testing team
- Establish rules for emergency 24x7

Organization is essential to a successful pentest. Before the test starts, be sure that working backups exist and any sensitive or off-limit systems are duly noted. Boundaries also should include when the test will begin and end, general rules, and any special circumstances. One of the most painful things during a pentest is to assume that a broken service was caused by test when it was already broken. Having a documented plan, details notes of tester's actions, and a backup taken before the test can save copious amounts of time and misplaced blame.

Clearly document the roles of everyone involved. If there is a defence team, establish who they are, what their responsibilities are, and to what extent they can interact with the test. Sometimes placing a knowledgeable third party in a position as a referee can help the testing stay on practical and useful ground. The offensive team needs to know everything for their role as well.

All parties involved needs to have these boundaries documented and understood prior to starting any testing. Any sensitive issues absolutely must be addressed in advance. Precautions useful in an emergency are also an important item to establish before starting.

## Key Definitions in Pentesting

- Attack Vector – A path to deliver a payload
- Leverage – using a component to better position or exploit another component
- Privilege Escalation – leveraging a low privilege account to a higher privilege account
- Remote Vulnerability – exploiting from an outside source
- Local Vulnerability – exploiting on the system itself
- Red Team / Tiger Team – offensive team
- Blue Team – defensive team

The terms attack vector and avenue of attack are sometimes used interchangeably. Avenue is normally used when describing an approach or technique of an attack while Vector implies a more precise action than a technique. Leverage is one of the key concepts in penetration: using one component against another. Remote vulnerabilities are the scourge of the Internet, especially when involving worms. Local vulnerabilities are usually viewed as less of a threat because their must have been some form of trust in place to be on the system.

Since penetration testing can dive deeply into technical arenas, the people involved in the test should be well aware of any application specific terms that apply to their environment during the test.

## Leverage In-Depth

- Penetration testing is mostly about discovery using leverage
  - We tend to see avenues of attack that can be represented in the OSI network model
  - We will skew the OSI model slightly to fit into our Penetration Methodology
- 7) Application ( public or private availability, input )
  - 6) Presentation ( encryption, checksum/checkpoint )
  - 5) Session ( an exchange, a specific instance of TCP traffic )
  - 4) Transport ( packaging )
  - 3) Network ( switching/routing- How to get from one host to another )
  - 2) Data Link ( local - How to get from one host to another )
  - 1) Physical ( the CAT5 cable, the USB thumb drive, etc. )

The most powerful weapon during a penetration test is leverage. If you can manipulate any function against something in the system you can control that system somewhat. By leveraging and manipulating one component you may be able to force the application to show weak points or make a weak point more vulnerable.

There are different areas where manipulation than occur. An obvious area is at any point where the application accepts input. By altering the input, an application could change its behavior or even break. Breaking an application often provides an error message. This error message could reveal new information that gives the tester an edge on finding new information about the application or a new vulnerability.

The physical area is often a weak point. For example, say a casual user can force your server to reboot by flipping a circuit breaker, he could have a system on the network pretend to be the server while the power is out and the users will try to connect to his malicious server with their passwords. Now with these new passwords this malicious user may be able to find new areas to explore or break into. In this example, the physical placement of the circuit panel was leveraged against the LAN system.

## Example Avenue of Attack

Layer	Attack Type Example
User	Social Engineering
Application	Input Fuzzing
Presentation	Abstraction Assumptions
Session	Cookie Stealing
Transport	Transport
Network	Man in the Middle
Data Link	ARP spoofing
Physical	Console Access

Our model needs an adjustment, however; we need to add the user. Social Engineering is simply manipulating a system on the human level. A perfect example is pretending to be a travelling executive that needs his password changed by helpdesk personnel. Ultimately, the user layer is usually the weakest link in our new model.

Attacks often involve more than one layer. Tricking a user into inputting bad data into an application could hide the tracks of an attacker. With the constant proliferation of the Internet and Web Services, leveraging an interface between two programs to the point where you can manipulate input between them is a significant problem. Without a user to see the interaction between the two programs, how long until somebody properly responds to the problem?

## Leverage In-Depth (continued)

- Things that may be leveraged
  - public access
  - private/authorized access
  - resources
  - environment assumptions
  - trust relationships
  - standards and assumptions

There are many things that could be leveraged in our model and between different layers of our model. Listing them here would be futile, almost as much as depending on it. Blind assumptions are usually a big portion of security failures. So as a penetration tester, you should strive to test all possible scenarios, even scenarios that people might say, “Yes, I see you broke the system, but in reality that would never happen.” Be careful to examine every detail but also how these small things might affect each other.

For example, visualize a boat containing 50 people standing. If just one person leans to the left, the natural buoyancy and people themselves can shift their weight so that boat will not tip over. But if everybody leans to the left far enough and at the right time, the boat could tip over. Even if it doesn't quite tip over, maybe a much smaller wave could cause the boat to tip over in this condition. Maybe your boat is big enough to survive the test without tipping, maybe it is not, but that knowledge is only as good as your last test . . . so you at least had to have tested it once to safely assure itself it will be safe.

## Techniques and Tricks

- Social Engineering
  - Manipulating the “user space”
- Physical attacks
  - Stop short of beating the system administrator
  - If you have physical access and can force a reboot, you usually have complete access to the system, for example:
    - Boot backtrack to clear the Windows Administrator password
    - Boot Linux into single-user mode and change the root password
- Combination attacks
  - Tap the keyboard with a key logging device, tell the administrator “I broke in, go check!” and when he does, you have his password . . .

Some of these areas are important enough to cover in more detail. Social engineering and physical attacks should never be forgotten. These require special consideration since the focus of the attack is limited by only the creativity of the attacker, and an attack tool is a luxury—not a necessity.

Leveraging one area against another in the same system can really magnify a weakness. This is true for practical tests as well as theoretical attacks.

This is even more evident when facing a Denial of Service attack. A DOS attack can be difficult to protect against. The system is obviously active for a reason, and when disabling a portion or the system itself, an attacker can deny the owner of a system its use.

How could one leverage disabling a system against another system? Have you thought about flooding an authentication server to keep it busy enough that it cannot keep up with the authentication requests? In this state, one could spoof the address of the authentication server to

receive user names and passwords. Or consider a packet shaping device: if this system is involved in a DOS attack, what does that mean to the services it is designed to adjust?

## Techniques and Tricks (continued)

- Literally any bit of information may be valuable as Recon
  - You may find systems that are “expendable” to the owner but may contain something you can use
  - You may find read-only access to logs that you can watch (valuable when you try to create errors on purpose)
- Look for trust relationships
  - Get access to one, and you have some sort of access to the other
- Look for indirect paths
  - Perhaps the SQL server is not public, but is used by the web server
  - Maybe the SQL server is also used for internal database
- Constantly adding to your collection of intelligence

A good penetration tester will take note of any information ran across during a test. While some information cannot be leveraged directly or indirectly, it may lead to new information.

The same tools used by a system administrator for troubleshooting and monitoring are valuable in offense as well. By watching the normal state of a system change, a tester may get an indication of where a weakness might appear under certain conditions. A race condition is a situation where a process is vulnerable to a timed attack.

Say a Joe process is designed to drink coffee. Normally, Joe walks the short distance from the office to the break room to make and poor his coffee. If after Joe makes the coffee then a Janitor stalls Joe by temporarily blocking his way, Jane might be able to taint his coffee before he gets there to pour it. Now that this race condition has been exploited and Joe drinks his poisoned coffee, he must immediately leave for the hospital, and either Jane goes through Joe's office and steels all his blue pens. Jane has taken advantage of Joe's delay to fuzz his coffee input and break the Joe process in a way that Jane can use to steal more blue pens.

## Techniques Summary

---

- Explore everything possible, from the largest hosts or applications down to every avenue of input.
  - Look for more holes
  - Look for more applications
  - Look for more clues
- Penetration is focused on “deep,” but do not forget “wide”
- Use Leverage wherever you can find it
- Take careful notes
- Think about assumptions and context, you may be able to manipulate the environment to open up new doors
- If you run into a dead end, start from what you do know and explore
- Lower hanging fruit is usually picked first, so start there
- **DO NOT EXCEED THE TEST BOUNDARIES**

## Process of Discoverability

- Each test should be repeatable
- Different kinds of progression
  - Going from what you know directly to what you do not know (serial)
  - Educated guessing using similar examples seen so far (parallel)
  - Guessing based on gut feeling or even random idea (brute force and usually pointless)

Without careful tracking of which tests have completed, a penetration tester can not be sure that nothing was left out. If a set of tests is deployed in a systematic manner the results will be much more accurate.

The differences between progression types is easy to see with an example. Say you are looking at a set of letters to determine their pattern. The first set of letter is ABCD. It is easy to recognize how to proceed from point A to B to C to D. This is a serial or linear progression, so we would be wise to consider EFGH as the next logical pattern.

Say the pattern is ABEFIJ. This is a little bit trickier unless you realize that the pattern is a vowel followed by the letter after it in the alphabet. By reasoning that the vowels are in order as well, we would anticipate the next step to be OPUVYX. This pattern also seems to be sequential, so our guess is also a serial one.

Now a third pattern is ZYXW. We could consider a serial pattern that continues towards the front of the alphabet in order, but with the knowledge of the previous patterns we can draw similarities in the patterns. We can consider that any new pattern is likely has an alphabetic order, does not include numbers, and never is more than six characters. These attributes we have

determined draw from their similarities but are not sequential (the three patterns do not follow each other). These similarities involve noticing parallel attributes of the patterns.

A tester needs to be very careful not to make bad assumptions, and this is where the documentation shows its value. By documenting the patterns and our own thought progression we can later re-evaluate them if we uncover a fourth pattern. We also know precisely where we stopped and can continue probing deeper into the patterns if needed, noting precisely where to resume the search.

## Typical Penetration Phases

- Perform Reconnaissance
  - Initial mapping and information gathering, focus on observation
- Port / Vulnerability Scan
  - Probe applications for potential leverage
- Manipulate and Exploit
  - Manipulate vulnerabilities and flaws for benefit
- Bootstrap the Penetration
  - Start the process over again from this new vantage point: Recon, Probe, and Exploit new avenues or objects

A typical penetration test can be divided into four phases: reconnaissance, probing, exploiting, and retesting with the new information and leverage. Recon and probing are similar except probing is more intrusive than merely observing. Probing naturally leads to exploiting; sometimes a vulnerability might be immediately known. This is one of the many reasons you must have appropriate permission for penetration testing, you may uncover dangerous things without knowing it until it scrolls across your screen.

A test progresses from vulnerability testing to penetration testing once the tester tries to go deeper into the system. Bootstrapping the process from a new location where a tester might have new conditions is a powerful concept. This new location may have access to non-public services or other resources. There may be new clues revealed to the tester where he could choose to back out a layer and try new tests. Although backing out of a deeper penetration point seems counterproductive, the new attack may reveal new problems and allow for a deeper penetration.

## Recon Tools

- Off site information gathering
  - Google / whois / Maltego / DNS
- Network mapping
  - nmap / nessus
- Host fingerprinting
  - queso / p0f
- Service probing
  - netcat / webscarab
- Custom scripts to harvest info

With the proliferation of the World Wide Web, a tester can leverage public servers for a penetration test. Using a whois form from a registrar such as [www.networksolutions.com](http://www.networksolutions.com) can get some contact information of a site. Often phone numbers, names of administrators, addresses of the site can be found. This information could be valuable seeds for starting a Social Engineering attack to get more private information about a target.

DNS requests on a company's domain name may reveal quite a bit of information as well. DNS traffic could give away information during this process. Suppose an attacker checks a DNS record with a DNS server under control of a defender. The defender might notice the query is preliminary to an attack if it is probing for all machines in a domain. Also, do not forget that an attacker might be able to watch his DNS servers in the event that defender tries to look up the attacker's IP address, informing the attacker that the defender is watching.

The lines between network mapping and fingerprinting has blurred in recent years. Nmap has been extended to include decent service version identity. There are still two kinds of maps that are made, network and services. The network map usually points out the hosts and their

relationships to each other, while the service map includes which services and versions are running on a host.

Probing with a generic network tool like netcat can let the tester read from a service port and send data to the service port. The webscarab tool is great for observing the traffic sent between a host and an HTTP server, also allowing for editing the input.

## Network Monitoring Tools

- Discovery new of assets to leverage
- Identification of testing breakage
  - wireshark
  - dsniff
  - snort

Network monitoring tools can be used for recording traffic, filtering traffic, and dissecting protocols. The most prolific tool is tcpdump. Ethereal is a common graphical sniffer useful for protocol dissection. Most network monitoring tools use a pcap library to interface with the network card and place it in promiscuous mode (listening to all traffic instead of just traffic sent to the host). Newer tools will allow for ARP trickery, allowing the interface to manipulate its MAC address so a switch or other machines on the network segment will think it is a different host. Ettercap is such a tool. Snort is a specialized sniffing tool that is commonly used to trigger alerts on specified traffic conditions.

An attacker can use these tools for reconnaissance. After observing normal traffic, the attacker may be able to plan an attack that will not set off many alarms. An attacker is also likely to uncover vulnerabilities when watching traffic, such as capturing passwords or privileged information.

## Perimeter Tools

- Any packet tool
  - hping (packet crafting)
  - firewalk (firewall testing Mapping out a firewall)
- Check other perimeter devices
  - Perhaps an IDS that make assumptions about fragmentation?
- Don't forget to think outside the building
  - Modem pool
  - Wireless

There are many tools useful for testing a perimeter. Some are simply packet generators designed to send creative packets to see how a perimeter device might respond. Some are designed to map an access list on the perimeter device. An intrusion detection system is likely to have at least one sensor on the perimeter, and maybe there is some way to get information out of the IDS or even leveraging the IDS in the attack.

The perimeter is NOT just the public facing firewall or router of a network. Please do not forget about the potential for wireless communication devices. These devices can easily be accidentally activated on the LAN behind the fixed network connections. As other communication devices converge into appliances, there may be devices on the perimeter without the owner's knowledge. Mopiers and other multi-function devices combine scanning, faxing, and copying. These devices have the potential of having some access over their phone line, as well as being plugged into the LAN for printing or scanning use. With the proliferation of PDA and phone hybrids, you may have a cell phone connected to the LAN via a PC that could be attacked. Also, do not discount portables that may be plugged into a partner network. It would be valuable to coordinate a penetration test with partners to evaluate the potential damage from portable computer activity.

## Application Tools

- HTTP proxies
  - Use webScarab to modify HTTP traffic
- Fuzzers
  - Recon but with consequences
    - Try to break the application to learn more about it
    - All attempts are likely logged
  - Designed to test input parameters
    - Type of input (alphanumeric, numbers, or object)
    - Size of input (underflow or overflow)
- Manually probe responses from application (netcat)
- Custom programs or scripts
  - Context and Environment leverage
    - Possible race condition
    - Weak path or file system permissions

In addition to watching normal HTTP traffic from the host using webScarab, a pentester can modify HTML fields before they are actually sent back to the server. Combined with input fuzzing, a tester can try many different combinations of input to break the application. This is especially useful to test whether all input validation is performed on the client side (such as JavaScript).

By manipulating the input with a fuzzer or manually, an attacker is most likely going to leave some sort of evidence via network traffic and in the log files of the application or server. Where recon transitions to probing with input is where activity can be observed by a defender.

## Exploit Tools

- Test for vulnerability
- Leverage vulnerability
- Favorites
  - Hydra (password guessing)
  - Metasploit (modular exploits)
  - Custom (ie: [www.milw0rm.com](http://www.milw0rm.com))

Normally to discover a vulnerability, a tester must actually exploit it. Because of the vast differences in systems and applications, especially in Linux and custom websites, a potential vulnerability may not immediately be obvious. Some of these vulnerabilities may exist on all systems, but the chances of there being some slight difference is great. For this reason, actually exploiting the vulnerability during pentesting is extremely valuable. A reasonable effort to avoid destroying a system during the exploit is the best practice. Otherwise, a tester would not be able to continue the test.

One of the most widely used tools in the exploit engine category is the tool called Metasploit. This is a very powerful tool that contains modular exploits and payloads. It is easily extensible and can be used to test known issues. By automating most of the extreme details in publicly known exploits, a tester can focus on the custom applications and services. While Metasploit is an open source tool, there exist other commercial solutions like CORE Security's Impact Professional which have powerful interfaces.

Other tools such as hydra are designed to automate repetitive tasks such as easy to guess passwords. In many situations writing custom scripts to automate these tasks is a huge time saver for the tester.

## Potential Targets

---

- Network
  - Perimeter devices
  - Internal nodes
- Hosts
  - Public facing
  - Private leveraging
- Applications
  - Escalated Access
  - Valuable Data

Intro to Penetration Testing - © 2009 James Shewmaker 24

So far, we have covered various types of systems and some of their components. Some of those components are usually the targets of a penetration test. It is important to remember to focus on penetrating the targets and not to get sidetracked with testing things that do not directly relate to the target. Intermediate hosts and systems can be valuable during a penetration test, but the point is to reach the target and not to find every possible vulnerability on the system.

## Penetration Testing Styles

- Styles
  - Black Box (Scenario A)
    - Begin with a clean slate and no insider knowledge
    - Simulates random target approach
  - Crystal Box (Scenario B)
    - Some previous knowledge
    - Specific targets
- Approaches
  - Internal (usually not Black Box)
  - External (completely outside the firewall)

There are two major styles penetration tests can be conducted in, black box and white box.

Black box testing tends to reveal more obvious problems to a casual user or attacker because the inner workings of a system are not available. A white box test is designed to identify particular issues in a system, with all source code and internal knowledge. Crystal box testing is valuable to evaluate these internal issues.

Starting a penetration test from the inside or outside can help limit the scope of the test by focusing on that half of the system. Black Box testing usually requires a tester to pretend they know less of the system than they usually do, so it is essential to log each detail and how they find it.

# Black Box Testing

- Starting with nothing
  - Reconnaissance
    - What shows up with a network sweep?
    - Anything interesting?
      - Hosts
      - Applications
    - Stop and think about how one might find this target in the first place (Google maybe?)
  - Going deeper and see what else you can find

The initial reconnaissance phase of a black box penetration test is not that different from a generic penetration test, since usually recon is already provided in a white box test. The nature of the black box approach should be kept in mind while testing: start with obvious targets such as on-line stores, email systems, intranet gateways. These higher profile systems would be the most likely targets in an actual attack and tend to be more critical in business functions anyway.

Avoid not testing anything but primary targets. In today's world, there exists a random element in attacks such as terrorism. Everything should be tested, but a focus on the likely targets is not uncommon.

## Crystal box Testing

- Tends to involve specific targets
- Easier to define scope than Black Box
- Must be even more careful to NOT make bad assumptions

White box testing usually has more of a quality assurance vibe than black box testing. White box testing usually has more specific targets and goals. Using white box testing only for penetration tests can be very dangerous. Security in general is usually weakened severely by making assumptions. Assuming that only people with internal knowledge can exploit a vulnerability is a typical bad assumption to make. What happens if an employee leaves the company or a partner has different standards? Could privileged information about the system design make it a larger risk?

## Aftermath - Now What?

- Sometimes you have to fix something testing may have broke
- The test is pointless without careful and precise documentation
- Documentation is pointless if it is not available
- Use the test results to plan corrective action
  - Specific patching and configuration
  - Plan for future patches and tests

During the course of a penetration test, it is common to have applications and entire systems break. Although the risk is usually unavoidable it can be minimized with proper planning and organization. Besides the potential problems that are directly caused when testing, the results are of no value unless the owners of the systems are informed of the issues discovered.

As part of the test results, a resolution to the issues should be documented. This will provide a course of action to follow. Ideally document a time line to implement fixes and retest. If there are vulnerabilities in a business process or incident handling, those should also be addressed.

## Scenario A - Introduction

---

- You are a consultant
- Given a company name, find out everything you can
- Highlight points of interest along the way
  - Potential risks
  - Working controls
- No network sniffing
- No breaking of applications
- Avoid disturbing production data

Scenario A is an example of a small white box penetration test. The objective is to determine the company exposure to the public. This test should not hinder normal business operations, just get a good representation of the potential and real vulnerabilities that are available to the public.

## Scenario A – Stage 0

---

**GET FINAL,  
AUTHORITATIVE,  
WRITTEN, AND  
SIGNED PERMISSION!**

Did I mention you need to have legitimate and definite authority for penetration testing? :)

## Scenario A – Stage 1

- Exhaust off site recon opportunities
  - Anyone for dumpster diving?
  - Start with free utility services
    - Google
      - using the company name to find indexed sites
      - using the "site:mydomain.com" feature to narrow down the results
      - using the "filetype:xls" to search for data
        - » Could be private
        - » Could be public data but valuable as recon

Since the penetration test's goal is to determine the public facing issues without disturbing the business, we will want to use all possible search engines and other public utilities.

We can start with domain registration checks with WHOIS and DNS lookups. For the purpose of this example, let us say that the only thing we find is bogus contact information and one name server. We query the name server for different types of DNS records but only find one for the web server, so we continue with reconnaissance of the web server itself.

Try clicking on each link on the public site. Look for obvious information that should not be in the public view, taking note of all errors and bad links encountered. Often bad links will give an indication of structure that may be used in parallel pattern detection. Look at the /robots.txt file in the root of the webserver to see which directories you are telling the search engines to ignore.

Let us continue with getting a feel for the company and checking for obvious problems. We will use Google's index and search engine to comb the public site for internal document extensions such as xls, doc, zip, and mdb. These are Windows type extensions but don't forget to try non-Windows ones such as tar and gz. Look at the /robots.txt file in the root of the webserver to see which directories you are telling the search engines to ignore. Try each one of these directories.

If you reach an input area, note it and move on, we will test the input later. Try other Google searches like finding what sites link to your site.

## Scenario A – Stage 1 Continued

- uptime.netcraft.com
    - Netcraft may have the host type, web server version, and uptime
  - samspace.org
    - Can check for various things
      - » DNS records
  - isc.sans.org
    - Can check an IP for reported info
    - Sometimes every little bit of info helps
- Note and evaluate the information so far
- Where can go from what we know to the next level?

After we exhaust our favorite content search engines, we can proceed with other search engines. Do not forget to try cached copies of the pages on the site via Google and [www.archive.org](http://www.archive.org). If things were accidentally placed on the server but then deleted, they might still be available in an online cache! Some on-line engines such as [samspace.org](http://samspace.org) can query other engines for your for different purposes. Do NOT forget to check the IP address of the website in all of the engines you use in addition to the name itself.

To confuse a detection of our recon, we could have used a proxy server to hide our source, but that is not really beneficial to our test in this scenario.

Go through the information collected and see if there is anything missing. Have you examined all that public can see without committing user input to the system?

## Scenario A – Stage 2

- DNS will be valuable
  - Does whois tell us that the DNS is hosted elsewhere?
  - Can we do a name transfer or figure out how to enumerate records?
- Email addresses of administrators in whois
- Maybe their website or email is on a shared hosting box, how can we tell? Is there any info we can leverage?
- Remember to keep recon as limited as possible to postpone detection

Let us start to evaluate the information we have so far. Can we glean any information about the hosting provider via the DNS or IP address? Is there maybe another co-located website we might be able to leverage (or notify of our test!)?

Is there any other environment information we should be looking at before we continue? Now that we have paused our testing long enough to consider other recon opportunities we can proceed with probing. If we get new information we will revisit this position. By limiting our recon to non-probing behavior, we may have avoided detection. More importantly, we definitely have not polluted any production database or log files since we have limited our input to casual browsing.

## Scenario A – Stage 3

- Skip Network Monitoring, outside the boundaries for this test
- Check out the perimeter
  - Start a traceroute to the public IP addresses you have so far
  - Note any host or service that shows up, ones that definitely do not exist, and ones that are unknown
  - If you find new services or hosts, conduct initial recon on these new items before proceeding to Stage 4

Since we have a target of our public web site and are testing externally, we really cannot (and definitely should not) perform sniffing of traffic. Getting consent from every public party visiting the website could be possible, but it frankly is not worth the potential problems and is outside the scope of this test.

Is there any other environment information we should be looking at before we continue? Now that we have paused our testing long enough to consider other recon opportunities we can proceed with probing. If we get new information we will revisit this position. By limiting our recon to non-probing behavior, we may have avoided detection. More importantly, we definitely have not polluted any production database or log files since we have limited our input to casual browsing.

## Scenario A – Stage 4

- Check the reachable hosts and devices with specific tools
  - Craft packets with hping to elicit responses from known and unknown devices
  - Manipulate reachable applications, try to break them manually or with a fuzzer; try to generate error messages
  - Look for new clues that may reveal another private host or other resource you cannot see directly

Now we will begin probing the system in our test. Using standard tools as well as more creative ones, check for open services and reactions to creatively crafted packets. Try to get a layout of the services reachable on the system with nmap or other port and service mapping tool.

One could use the nessus tool here as well to check for known vulnerabilities. We will focus on trying to manipulate the input of our web applications with a typical browser and webscarab. Do NOT forget to try adjusting the hidden form variables in HTML. Look for clues to input validation in javascript and try to send invalid input webscarab. Note the errors you receive. Keep in mind you may find new information about another host. If you do find such information, you will want to go back and perform additional recon on any new host or site.

While sending invalid input, remember to test for invalid types of input as well as input that is incorrectly sized. Send input that is both small and very large. As with everything else, note each error message.

As you test the input of the web application, try to probe deeper. For example, if you receive any internal errors such as “file not found,” see if with input you can manipulate which file it actually looks for. If you can pull up the system's password file, then this is a serious vulnerability

indeed! Remember each time you get past a layer or area, take a brief look to see what additional info is available.

## Scenario A – Overall Results

---

- Documentation
  - Process of discovery
  - Tree diagram representing where and how deep the discovery went
  - List of publicly reachable devices and applications
  - List of test conditions and generated errors
  - List of known exploitable conditions

The results for this penetration test should show what information and services a public user was able to reach casually. The report should also contain a list of the errors and any precise fixes that can easily be implemented. If a vulnerability is more complex, an addendum to the report should contain the context of the problem and the solution. Separating vendor vulnerabilities and custom vulnerabilities is usually a good idea in the report.

A tree diagram or other map of the publicly facing information is a great way to illustrate the results of the penetration test. Also include general recommendations for future testing and any crucial issues encountered.

## Pentesting Hands-on Results

---

- Initial reconnaissance
  - Service Map of host using nmap
- Network monitoring
  - Switched network, no benefit
- Perimeter tests
  - Host is target, perimeter testing is no different than host testing
- Application tests
  - Testing reveals vulnerabilities
- Exploit
  - Leverage vulnerability with recon
- Bootstrap the test to the next level

## Scenario B – Introduction

- Defcon 2005 Capture the Flag prequalification
- Emphasis on penetration and vulnerability discovery
  - Kickoff: an email with an http link to begin the contest, and a username and password for later use
  - There are eight (8) flags representing valuable data
  - That is all you have to go on!

Now that everyone has a penetration test of the same system under their belt, we will cover the critical points in depth and relate it to the actual competition. Keep in mind this event was designed to find people skilled enough to compete in one of the most hostile networks.

While we progress through the testing, the instructor will demonstrate key vulnerabilities. Due to time constraints, the instructor will not be testing every single potential vulnerability, but will cover the significant ones in a way to best represent a good penetration test.

## Scenario B – Stage 0

---

- HTTP link reveals several things
  - Hostname dujour.kenshoto.com
  - Running webserver on port 80
  - Some content
- Where does recon stop and preliminary testing start?
  - Sometimes you don't know what you're testing until you break it!

Our preliminary look at our instruction page shows us a link to a webserver. There is little content on the server. If we disconnect from the class and search the Internet for cached copies and ownership info, we also find little. So we must proceed with the information we have, which looks like it will involve probing the site and its applications as the next step.

## Scenario B – Stage 0 (continued)

- Recon
  - Host info
    - nmap/netcraft/whois/arin
  - Host services
    - Map them with nmap/nessus/amap
    - We see that there is an HTTP service, we can try exploring
      - Default directories/files
      - View source for clues in comments or other links

We get some information displayed when looking for other services on the host. Most obvious is the HTTP service we already knew about. Since we are very short on publicly available files on this site, maybe we should try some default, similar, or random files.

Remember we also want to keep in mind any similarities, and see if we can make any logical guesses as to what other files may exist. If we see a login.php page, might we expect a logout.php page to exist?

If we can distinguish which operating system or httpd daemon is actually running, we may know or be able to find default or sample files. This is one of the times where we disconnect with our new info (operating system and application versions) and gather a little more background info.

## Scenario B – Stage 0 Results

---

- Nmap says

- 22/tcp open ssh OpenSSH 3.8.1p1 (protocol 2.0)
- 80/tcp open http Apache httpd 2.0.53 ((FreeBSD))
- 6969/tcp open acmsoda?
- 19150/tcp open unknown

Searching for these application versions on-line does not appear to give us any super low hanging fruit. So the system appears to be reasonably current and up to date.

## Scenario B – Stage 1

- <http://dujour.kenshoto.com/cgi-bin/stage1>
- Learn more about the application by viewing the source, exploring, and breaking the web application
- HTML form has a hidden form field named "message"
- <http://dujour.kenshoto.com/cgi-bin/stage1?message=aa>
  - File Not Found error!
  - We also learn that the application doesn't discern between POST and GET variables

So we start getting into examining the first application we were pointed to more closely. By viewing the HTML source, we can see a hidden form field. How does the application behave when we change that field. What kinds of values can we change it to? What kinds of errors do we get?

## Scenario B – Stage 1 Results

- Manipulate the CGI link from Stage 0
  - `http://dujour.kenshoto.com/cgi-bin/stage1?message=/etc/passwd`
  - Download `/etc/passwd` and `/etc/passwd.shadow`
  - Combine and crack with john and a decent dictionary
- Meanwhile more recon by leveraging Stage 1
  - Other key system files (`/etc/hosts.allow`)
  - Other application files (`/usr/local/etc/apache/httpd.conf`)
  - Log files (`/var/log/messages`)
  - User files (`/root/.history`)
  - Temporary files (`/tmp/*`)
- Discover username/password
  - `breakme/apple1` and `root/fred`

So if you blinked and missed the vulnerability, we are able to manipulate an application to let us download any file on the system the web server's user can read. That is one huge hole!

Now, we have a whole file system to examine for new information and vulnerabilities. For the sake of time, we will start with the password file. Let us download that, take a medium sized dictionary and run john the ripper on the password file. While that is running, we will continue to look through the system and note the key areas.

Bonus: Having access to reading the log files is a phenomenal value. We can see errors in the log file if they are suppressed by the httpd server. We can also watch the logs for other testers getting to the next stage!

Super Bonus: Who has control of the log files? Can you trust what you find?

## Scenario B – Stage 2

- ssh breakme@dujour.kenshoto.com
  - Returns binary data and closes connection
  - Appears to be base64 encoded
  - No other new leads
- Can continue this route or look for lower hanging fruit
  - Let's map the services on the host
  - Don't forget to watch log files with the CGI form
    - Note typical behaviour
    - Note error messages due to your probes

Well we have a lot of information to go through, but since we are penetration testing we should try deeper before we try wider approaches. Let us use the account that we cracked and just try it to see if we can get in. We cannot get to a usable shell with ssh, but we do get something odd. Some odd characters and then getting disconnected is what we find.

Based on personal experience, I have seen such a condition where something was automated to connect, run a program, then disconnect. Let us take a closer look at the data we received.

We need a tool to record this data. Using netcat itself is not the greatest idea itself because of the ssh encryption. For this example, I will use ethereal to record the traffic as I connect with an ssh client. By following the packets, I can extract the data from the session and save it to a file for closer examination.

## Scenario B – Stage 2 Results

- Reverse Engineering the Binary reveals clues
  - Useful tools include gdb, strings, and metasploit
- Binary appears to communicate on port 6969
- Vulnerable to overflow

Now this course is not one on Reverse Engineering binaries, but we can get some info if we want the digital equivalent of poking the data with a stick.

The data is actually an ELF binary that we can decompile to find a buffer overflow. We can cause this program to overflow to get it to run a program for us. One of the easier ways to do this is with the help of metasploit's payload. All we have to do is whip up a small exploit in metasploit to take advantage of their built in payloads. What we expect to be able to eventually see is a way to execute a program on the server before this application disconnects. Maybe we can create a user, or prevent it from kicking us off.

## Scenario B – Stage 3

- Play with the binary with netcat (using username/password from Stage 0)

```
$ nc -vv dujour.kenshoto.com 6969 Warning:
inverse host lookup failed for
206.131.226.59: Unknown host
dujour.kenshoto.com [206.131.226.59] 6969
open AUTH:team13:tUqXasJuxM
OK
```

- Back to trying to break it
  - Try bad username/passwords
  - Try invalid input after authentication
  - Step through the binary with a debugger
    - Helpful tools are gdm and ktrace
  - Find an overflow to leverage

This can be a very time consuming process, especially for the beginner. You would be wise to read the ever popular Phrack 49 “Smashing The Stack For Fun and Profit” for this situation.

## Scenario B – Stage 3 Results

- Eventually overflow the binary to open up a remote shell
- Attempt to leverage this new shell
  - Create another account
  - Enable ssh access
  - Hide evidence of penetration?
  - Create misleading evidence?

Eventually we can send an overflow and shellcode to create a listening shell with netcat. We can connect to this port, though we do not have a TTY assigned, so we cannot answer any prompts. Short of a command that requires an answer to a prompt, we can now execute as the user running the exploit!

Now, before we get any deeper, we have reached another significant milestone. This is one of those places where it pays to take a quick look around and note any new info we see. We can run a “ps” command and look at processes running. We can look at directory permissions. We can also create files.

After we take a chance to look around, let us get back to seeing if we can go deeper in our current attack. We do have a shell of some sorts, but we do not have a TTY so we cannot use “passwd” or “vi” interactively. What we can do is edit files with “sed” or a custom script if we wanted to. First let us use sed to change the shell of our exploited account so we can actually log in with sshd. That way our traffic is encrypted and the other testers will not be able to cheat by watching our plaintext traffic across the wire.

## Scenario B – Stage 4

- We have access to the machine, so now what?
  - Privilege Escalation
  - Data discovery
- Possible avenues
  - Perform forensics on the box for clues to valuable data access
  - Perform monitoring on the box to reveal admin or other user access
  - Brute force additional access

Now that we are in a more comfortable shell, we can continue to look around at files. Let's look at the files in the /home directory! We see they are relatively well protected permission wise. By now we also should have noted that we have a root password. Our host will not let us become root from this user with “su” --unfortunately we are not in the wheel group. Maybe we can leverage that somehow.

It happens that except for the easy to guess password on the root account, it is fairly difficult for us to use it at this stage. For the sake of this walk through, consider that we spent an awful lot of time trying to find suid binaries, race conditions, and other local vulnerabilities to elevate our privileges. All we find are the suid binaries in some of the home directories that we cannot yet read.

## Scenario B – Overall Results

- Repeated the process, trying all existing accounts and system binaries
- This scenario had several types of overflows, including heap, stack, and format string overflows in local applications
- Report includes the extent of how deep we reached into the system

There are different types of vulnerabilities in these binaries that have exploitable conditions such as format strings, buffer, and heap overflows, but you get the idea. We successfully penetrated to a local level where one can read the filesystem remotely, login as a user, and run commands.

We could potentially watch for commands and files on the system to run, often an admin may mistype his password in the login field instead of the password field on the console, and it gets logged! Watching the running processes can reveal to us anything used on a command line sometimes also a password. We definitely have reached a point of privileged local information, where most files on the system can be read.

At this stage we document the initial vulnerabilities and each additional vulnerability we found. If we can draw up any solutions we will document that too, though often the solution involves a business decision or at least some more research.

## Conclusions

- Where do we go from here?
  - Increase penetration testing depth
  - Increase penetration testing scope
- How can I reliably test production networks and systems?
  - <sarcasm>Be sure to let the rest of us know when you find the perfect way</sarcasm>

Hopefully you desire more information on penetration techniques. SANS offers several courses designed to provide more details. With regards to penetration testing, the following SANS classes are directly applicable: I keep a schedule of both free and paid classes at <http://www.bluenotch.com/events/>.

Security 504: Incident Handling and Hacking Techniques

Security 542: Web Penetration Testing

Security 560: Network Penetration Testing and Ethical Hacking

Security 553: Metasploit for Penetration Testers

Security 610: Reverse Engineering Malware

Security 702: Exploit Development

# Summary

---

- Methodology Concepts
  - Precise and Systematic Testing
  - Leverage anything an attacker might have or obtain
- Tools
  - Like anything else, use what your comfortable with and what would be appropriate for the environment
- Targets and Scenarios
  - Get Permission and Prepare
- Documentation
  - Precisely document what you find: good, bad, or seemingly insignificant
- Aftermath
  - Put the knowledge gained from testing to good use

## For More Information

---

- Oregon vs. Randal Schwartz  
<http://www.lightlink.com/spacenka/fors/>
- The Hacker's Choice <http://www.thc.org>
- Metasploit <http://www.metasploit.org>
- Nmap <http://www.insecure.org>
- IP Address Allocation
  - <http://www.ietf.org/rfc/rfc1918.txt>
  - <http://www.iana.org/assignments/ipv4-address-space>
- Snort <http://www.snort.org>
- Fteter <http://fteter.sourceforge.net>
- Backtrack <http://www.remoteexploit.org>

## For More Information (2)

---

- Maltego <http://www.paterva.com>
- Impact <http://www.coresecurity.com>
- Exploits <http://www.milw0rm.com>
- More free material
  - <http://bluenotch.com/resources>
  - <http://sans.org/resources>
  - <http://www.phrack.org>