Pentesting

ECE478 Network Security

Dean Johnson & Emily Dunham

June 13, 2014

What's pentesting?

Penetration testing is the occupation of those security professionals who attempt to gain access to a system using only the tools and knowledge available to a malicious attacker, but with the system's owner's consent [1].

Pentesting includes automatic vulnerability scanning, manual testing, and a variety of other techniques that mimic those which would be used by malicious attackers.

Types of tests[2]

Targeted testing is a form of "white box" testing in which testers work with the client's internal teams who maintain the product being tested. This can help simulate the level of knowledge and access available to an attacker such as a disgruntled employee.

External testing explores how far an external attacker could get into a client's systems by performing an attack from outside the company's network and systems, trying to break in. External testing usually involves giving the tester some special knowledge about the systems, which might not be available to all malicious attackers.

Internal testing seeks ways that a disgruntled employee could escalate the priveliges of standard system access in order to harm the customer's systems.

In **blind testing**, the tester is given minimal information about the network being tested. They will know the company's name, which can be used to explore tech blogs, social networking sites, press releases, and any other disclosures of information – however, the tester knows no more than a malicious attacker from outside the company would.

For all of the above types of tests, it's assumed that all of the relevant personnel inside the client company will be aware that the tests are being performed. However, in **double blind testing**, the client's teams who run the systems being tested are not informed that the test is taking place. Double blind is the only form of testing which can accurately predict the client's response to a real attack, since in all other forms of testing people are warned about the test and will be more vigilant during the testing period.

Pentesting Clients

Any company or organization who could be harmed by a malicious attack on their computer systems can benefit from penetration testing. Some of the most common pentesting clients include banks, e-commerce sites, governments, corporations, and militaries.

Additionally, any company that builds or sells products that make security claims, such as firewall programs or operating system code, can save itself the embarrassment of some public vulnerability disclosures by having its product pentested before release. Open source projects related to network security, such as the router firmware project Tomato, generally get some level of security testing from ordinary attacks against their users' deployments.

How can pentesting be done legally?

The two key differences between pentesters and malicious attackers are that the pentester attacks a system only with its owner's consent, and the malicious attacker exploits discovered vulnerabilities in a way that harms or disregards the wellbeing of the system's owner.

A legal pentest starts out different from a malicious attack in that the tester gets the system's owner's permission first. The pentester can optionally be given additional information about the system, to help guide their attacks by avoiding time that would otherwise be wasted in gaining those basic facts.

Acceptable Use Policies

As an example, OSU's AUP contains the wording:

> Entry into a system, including the network system, by individuals not specifically authorized (by group or personally) or attempts to circumvent the protective mechanisms of any University system are prohibited. (http://oregonstate.edu/senate/agen/2006/aupcurrent.pdf)

Most acceptable use policies prohibit malicious attacks while permitting authorized pentesting using a clause about **authorized use**.

Additionally, cloud hosting providers such as Amazon often have policies about doing pentesting from servers they host, because pentesting attacks will necessarily match the same profiles as illegal attacks[3].

> Permission is required for all penetration tests. To request permission, you must be logged into the AWS portal...

Jobs in the field

Getting into the pen-testing / computer security industry is not too difficult. It used to be that many 'hackers', as portrayed by the media anyways, were black-hat, and often got into the industry by breaking systems without permission. However, in recent years their have been many improvements to our current systems of reporting bugs. One such improvement has been bug bounties.

Bug bounty programs are set up by many well known companies, such as Facebook, Google, Yahoo, and many more. These programs allow security researchers to try and find security vulnerabilities in these companies systems and report them for a financial reward. Before, people may have tried to find vulnerabilities in systems for these companies without permission, and instead of being rewarded, legal action was often taken against the hacker (even if the hacker had good intentions).

Another way of getting into the computer security industry is getting certifications, to prove you know what you're doing. However, getting real clients yourself, or getting involved with a good consulting company usually requires more than just certification; experience is the most valuable.

Tools & Techniques

Metasploit

Metasploit is a collection of tools, the most popular of which is the free and open source Metasploit Framework. The framework breaks down the task of executing foreign code on a remote system into 4 simple steps, each of which is configurable based on your use case.

First, select an exploit. An exploit is simply a piece of code that takes advantage of a bug in the target system. The Metasploit Framework comes with over 900 exploits for Windows, Unix, Linux, and Mac systems. The framework can optionally check whether the target system's version and specific configuration are vulnerable to the selected exploit. All the exploit does is allow you to run an arbitrary piece of code - choosing what that code does is the next step.

Next, select a payload. The payload is the code that gets run on the target system once the exploit succeeds. The payload could be as simple as a wall message or fork bomb, or as complex as a VNC server.

At this point, the combination of exploit and payload can be easily detected by intrusion prevention systems. The third step of using metasploit is to select an appropriate encoding technique to make the exploit and payload look like benign data, or be ignored by the target's security systems entirely.

Finally, execute the exploit. If you've chosen the exploit and payload correctly, the payload code will run on the target system. To minimize guesswork and wasted time, a pentester needs to know some basic information about the operating system and network configuration of the target system.

NMAP

Nmap is another tool in every pen-tester's arsenal. At it's most primitive levels, it's a port scanner. However, it has so many more features than just port scanning. By using nmap, you can find which computers are on a network, what operating system (and version) those computers are running, the programs (and their versions) running on those computers, and ports that are open on those computers as well. Using nmap, it makes it easy to do a system-wide assessment of the security on a network, and helps you lock down systems that are vulnerable.

A basic example of using nmap may look like this:

```
nmap -v -A \textit{host(s)}
```

What this does is run nmap with the verbose and application flags turned on, against a given host. Verbose means it will return extra information for each of the scripts it runs, and -A ensures it returns applications running on the target host. Since we enabled the -v option, it also returns version numbers for each of those applications, so we can look for unpatched vulnerabilities available for exploit on the server.

Nessus

Nessus is a (now) proprietary piece of software. For versions 2.2.11 and below, they were GPL licenced, and the source code was available online. Nessus is sort of like a type of swiss army knife of network security tools. Is has a very unique pluggable architecture which allows you to take tools like nmap, Hydra (a weak password checker) and Nitko (a cgi/.scripts checker) and integrate them all together. Nesses also has some features unique to itself, like checking server misconfigurations, denial of service attacks and PCI DSS (Payment Card Industry Data Security Standard) Audits.

Social engineering

There's an adage among systems administrators which states that the only truly secure server is one that's powered off and unplugged from the network. Unfortunately, this is not a popular security strategy among companies who want their servers to actually serve data to the correct people. The problem with this idea is that the server itself has no way of knowing who the correct people actually are.

All of the pentesting techniques outlined so far have focused on tricking systems into executing code that they shouldn't, or finding out how to. Social engineering, on the other hand, involves getting data from authorized users that allows an attacker to impersonate them.

Social engineering can gain account credentials or other information necessary for more technical attacks.

Pretexting is a social engineering attack that involves impersonating someone to whom the target thinks it's appropriate to provide the desired information or access. For instance, a worker with a safety vest with the power company's logo and a clipboard of forms can often pass unimpeded into a datacenter, where someone in a business suit or jeans and T-shirt would be questioned or restricted from access.

However, in some situations it's better not to stand out from the rest of the target's employees when seeking physical access. Someone dressed like the other workers can often gain entry to restricted areas by **tailgating** an authorized person, following them through an ordinarily locked door. Tailgating is more effective when the attacker has a plausible reason to not open the door themself, such as carrying an armful of books or boxes.

One psychological effect that's often combined into social engineering attacks is **quid pro quo**, latin for "something for something". This is especially effective when negotiating for access or information: If the target feels that the attacker has gone out of their way to do something for them, he or she will feel obligated to repay the percieved favor.

Phishing and **spear phishing** attacks are a form of pretexting in which the attacker sends an email posing as an organization to whom the target is expected to disclose sensitive information. Phishing includes the familiar "Your paypal account is suspended, click here and enter your password to reactivate it" type of spam that we've all learned to ignore. Spear phishing uses phishing messages that are customized for a particular individual. For instance if an attacker doesn't know the target's bank password but has gotten their account number through other means, it sounds far more legitimate to say "Fraudulent activity detected on this account number, log in to view details" since the target is likely to assume that only their bank would know that their email goes with their account.

Finally, **baiting** attacks don't require any direct interaction between the attacker and the target, but instead leverage common traits of human behavior. A baiting attack involves leaving an item such as a USB stick lying around where the target is likely to pick it up, and then simply waiting until the target plugs it into a computer to see who it belongs to and what it contains. Infected USB sticks are believed to be the vector which introduced the stuxnet worm to Iran's nuclear centrifuges.

Vulnerabilities

SQL injection

The most common type of exploited vulnerability according to [6] is SQL Injection. SQL Injection is a technique that relies on formatting your input to a form that is sent to a database in such a way that gives you either unauthorized permissions, or a view of more data than intended. For example, if you have a login form, and know there's an account named admin, you can specially format an input string to bypass checking the password to the account if the SQL is not properly escaped.

We do this by making an educated guess at what the SQL query may look like, then formatting it to do something other an intended. For example, a typical login form probably has a SQL query that looks something like this:

```
SELECT * FROM users WHERE username='\%s' AND password='\%s';
```

If we then pass in a username that looks like "admin'; –", we effectively make the statement look like this:

```
SELECT * FROM users WHERE username='admin'; -- AND password='\%s';
```

Since "–" escapes the rest of the statement, we then don't need a password, and are authenticated as the admin user.

This obviously can have huge reprocutions. With statements that use SELECT and display data from it, you can essentially format strings to get a dump of the database. This can include passwords and other sensitive information, which is why this vulnerability so critical to be aware of in your applications.

So, how do you prevent SQL Injection? The only thing you have to do is escape your user inputs. This includes ignoring any characters that may be 'injected' into your SQL statements that can give unintended results. So, as easy as it is to make the mistake of leaving this open as a vulnerability, it's also incredibly easy to patch.

XSS

Cross-site scripting (XSS) is another fairly common web application vulnerability. Similarly to SQL Injection, what you essentially do is format an input string that when displayed to another user, can cause harm to them.

For example, if we are tweeting out to the world and try to include ¡script¿ in our tweet it escapes the ¡script¿ and doesn't ACTUALLY load a script like the browser usually would when displaying your tweet to other users. This can cause a lot of harm when a script is loaded that steals session tokens, cookies, or other data stored in your browser that can allow an attacker to impersonate you on other sites.

Privilege Escalation

Another type of vulnerability, not specifically on the web application side of things, is privilege escalation. This usually happens on the OS level and essentially means you gain access data which you are not supposed to have access to. There are two types of privilege escalation.

The first is vertical privilege escalation. Vertical privilege escalation is when an attacker grants himself higher privileges than he is supposed to have. An example of this is a developer granting himself root on a server where he is only is only supposed to have a basic level of access.

The other type of privilege escalation is horizontal. This is when an attacker uses the same level of privileges he already has, but assumes the identify of another user with the same privileges. For example, if a developer and system administrator both have root on a box and the box is using a shared file system for all users, th developer could "sudo su" as the system administrator, and have access to all of his/her files, keys, and other sensitive information.

Insecure File Creation

Another system level vulnerability is insecure file creation. A broken application may try to create a temporary file in a public writeable directory (e. g. /tmp) without forcing create-only. The attacker then can create a symlink pointing elsewhere (where the attacker does not have write permission). The vulnerable application will overwrite the target file (e. g. /home/someuser/.profile for normal users and a system files for root), giving them access to automatically run malicious code on the server.

After Testing

After finding vulnerabilities and exploiting them to the extent specified in the initial contract, the pentesting organization will present a report to the client.

If pentesting has revealed a bug in software that's used by others, a Common Vulnerability Exposure (CVE) would be published. Published CVEs are available from the National Vulnerability Database[7].

In reporting which vulnerabilities were available, pentesters often make security suggestions to the client. This can include advice on applying patches to known bugs, and strategic changes to mitigate the effects of unpatched vulnerabilities.

Bibliography

[1] https://www.sans.org/reading-room/analysts-program/PenetrationTesting-June06

[2] http://searchsoftwarequality.techtarget.com/definition/penetration-testing

[3] https://aws.amazon.com/security/penetration-testing/

[4] http://www.securitycurrent.com/en/writers/mark-rasch/legal-issues-in-penetration-testing

[5] http://insidetrust.blogspot.com/2011/01/penetration-testing-permission.html

[6] http://www.sans.org/top25-software-errors/

[7] http://nvd.nist.gov/