Ethical Hacking

Casing the Establishment

It's All About Anonymity, Stupid

As the internet evolved, protecting one's anonymity has become a very difficult task, and many systems have been developed with the objective of providing strong anonymity while also being practical.

Among all these systems, the most widely used is **The Onion Router**, or **Tor** for short.

Tor is a low-latency anonymous network of *onion routers*, that allows users to communicate anonymously across the internet: volunteers operate an *onion proxy server* on their system, that allows users of the Tor's network to make anonymous outgoing TCP connections.
The users of the Tor's network must also use an *onion proxy server* on their end, which allows them access to the network and negotiate a *virtual circuit* that will be used to perform the connection.
The anonymity is guaranteed by advanced cryptography applied in a layered manner, exactly like an "onion", and uses a *SOCKetS Proxy* to apply this property on 90% of Internet's services (instant messaging, IRC, web browsing, etc.…).

Essentially, Tor works by making our connection jump from different nodes all around the world following a path that is generated randomly for each new request, making it very hard (although not impossible) tracing back the original source IP address.
The packet is also encrypted a number of times that is equal to the number of Tor nodes used in the path, and each time a Tor node process the information a single layer is resolved, using a private key that is only known between the current Tor node and the next one.

To ensure perfect anonymity, Tor is not enough for two main reasons:

1. web browsers tend to exchange data that are outside Tor's control, therefore leaking information
2. proxy servers may block the traffic generated by Tor, therefore special proxy servers should be used to ensure both that the traffic is allowed and that it remains encrypted

One example of browser that avoids leaking information while browsing the web is **Vidalia**, that also gives a GUI for controlling Tor functionalities.

One example of proxy service that is used with Tor is **Privoxy**, a web filtering proxy that ensures an additional layer of anonymity and security when surfing the web using Tor by modifying the web page data before the page is rendered by the browser.

After a weak site to exploit is found, we want to determine which services are running on the system while also avoiding contacting the system directly and giving away our location.
To do this, we can use a specialized tool called **NMap** *(Network Mapper)* and run it through the Tor network using Privoxy to ensure anonymity, but first we need an IP address to probe.
To do this, again, we need to avoid sending a direct packet to the target host, so we can use **Tor-resolve** to solve the IP address while making the request traverse the Tor network.

Now that we have an IP address to probe and a tool that allows to perform anonymous requests, we can start scanning for open ports and find out which services are currently running on the machine.

Assuming the scan was effective and a service with poor security is found, we can exploit it and then the machine will be ours.

Chapter 1: Footprinting

We define **footprinting** the fine art or information gathering about one's target, before conducting an attack.
The objective is to find as much information as possible about the target *without sending a single packet to it*, which means maintaining the highest level of anonymity before blowing the cover at the moment of the attack.
We are not only interested at the internal structure of the company, but also at their level of security, how they are prepared in case of attacks and which countermeasure to be aware.

We can summarize the main focus of footprinting in the following list:

- *Internet:* domain names, IP addresses used, DNS hostnames, TCP/UDP services, system architecture, Intrusion Detection System, usernames, ecc…
- *Intranet:* network protocols that are being used and internal IP addresses and domain names
- *Remote access:* phone numbers, authentication systems and VPNs
- *Extranet:* connection source and destination, access control mechanism, ecc…

Most of this information can be found publicly by simply querying Google in the right ways, by performing on-site scouting activities or through *social hacking*.

Performing this technique on your own company gives you a vision of what the hacker can gather about you, which security information are publicly available or can be deducted from the internet, so that you can prepare in the most efficient way against an attack.

STEP 1: DETERMINE THE SCOPE OF YOUR ACTIVITIES

First, we need to determine the **scope** of the footprinting activity.
Usually, big companies do not rely only exclusively on themselves to operate, therefore footprinting the entire organization only may not be enough: we also may need to check all the other companies that are doing business with the main one.

STEP 2: GET PROPER AUTHORIZATION

To perform a test cyber-attack on a company, to ensure that it is safe, we need to make sure that we are authorized to do so, otherwise we may end up in jail!
Our work may also corrupt (or disrupt) part of the system, so we need to be sure to the limit of *what* we are allowed, *when* we're allowed but most importantly *who* is allowing this job.

STEP 3: PUBLICLY AVAILABLE INFORMATION

The amount of information available in the web about you or an organization and its employees is quite vast.
We define the most important categories of information to look out for.

Company Web Pages

Browsing the target organization's **web page** is a good starting point, especially ones that provide excessive amount of information.
Reviewing the *HTML source code* of the page can also hide crucial information in the form of comments left by the developers, maybe making aware the next developer of a certain bug of vulnerability in need of patching.
Using tools like **Wget** (Linux) or **Teleport Pro** (Windows) we can make a local copy of the entire web server, allowing a more in-depth analysis of the source code.

Most web pages also have "hidden" files and directories, that are not indexed by Google, so we can use *enumeration* tools like **DirBuster** to discover them for further analysis.
Performing this kind of operation is rather noisy and detectable, so routing all the traffic through *Privoxy* helps maintaining a high level of anonymity.

While remaining in the realm of "hidden information", companies usually have *test sites* (still under construction) under special types of hostnames, such as ww1, web, test, etc…

Many companies rely on external services to handle *remote access* for their employees, like:

- Microsoft's Outlook Web Access
- Microsoft Exchange servers
- VPNs

Especially with VPNs, it's possible to find information such as vendor, version details or even documentation on how to download and configure the client software.

Related Organizations

While the target organization may be security-oriented with its infrastructure, they also usually outsource to external organization that do not care as much about privacy.
This means that they may expose crucial information about their clients that could be user later for *social hacking*.

Location Details

Having the physical address of the target organization may lead to *nonethical attacks*, such as:

- dumpster diving
- surveillance reconnaissance
- unauthorized access to buildings
- unauthorized access to wireless/wired networks
- satellite imagery of the location via Google Earth, Google Maps or Street View

Furthermore, as the Google Street's car drive around the country, it also tracks any Wi-Fi networks encountered and their MAC addresses, that can be used with Google Location to track the exact position of an individual.

Employee Information

Contact names, e-mail addresses and phone numbers are particularly useful data to exploit *social engineering* skills.
Since most organizations use some derivative of the employee's name for their username and e-mail addresses inside the company: if we are able to know even one of them, then we can easily figure out all the others.
Phone numbers can also be used on different sites to also obtain more private information like addresses and relatives of the owner, which again can lead to more effective *social engineering attacks*.
Gaining access to an employee's home computer and exploit it with a key-logger gives the attacker a free-ticket to access the organization.

Checking all social medias of the target organization and posts made on professional/career sites is also a good choice, specifically to retrieve information such as:

- past or current employees' names
- events of major relevance
- work locations

All these information can then be used to broaden the searches on Google, allowing the attacker to discover more and more about the target.

Another crucial source is job posting sites, which organizations use to look for specific professional figures.
Since the company must provide high level of details in order to get qualified leads (such as application's vendor, version, responsibilities, experience required, etc…), an attacker may use all these information to gain knowledge on the internal structure of the target, applications used, IDSs, etc…
It is also possible to look for past resumes sent by the target's employees, which again may contain useful information about the internal structure.

Lastly, an ex-employee may of the target represents may represent a real threat to the organization's security.
Some people have gone as much as stealing, selling or giving away sensitive information, damaging or destroying data, leaving back-doors and many more acts of revenge.

Current Events

When a big organization is undergoing a major change, security is usually softened with the excuse "we'll fix it later".
In reality, "later" as in few weeks transforms into months of even years, thus allowing an attacker to exploit this frailty in the defense system.

Morale also comes into play during these events, as people may be more interested in updating their resumes and look for another job rather than updating the entire system to the latest patches.

Another very simple case is when an organization is experiencing a rapid growth and a large number of new employees starts working, often their processes and security procedures lag behind.
It could happen, for example, that an outsider is able to blend with all the newly hired workers and gain an enormous amount of information in a relatively small time.

Lastly, one the oldest trick in the book is extortion, and having potential dirt on a major figure of the target represents a great advantage.

Archived Information

Any piece of information that provides insight into the target organization's privacy, security policies or technical details regarding hardware and software used to protect the organization, can be useful to an attacker for obvious reasons.
There are sites on the internet where you can retrieve archived copies of information that are no longer available from the original source, maybe that have been deliberately removed for security reasons.

Search Engines and Data Relationships

Search engines today, within seconds, can provide you with anything you could ever want to know about.
Becoming familiar with the advanced searching capabilities of these sites, also known as *hacking*, can allow an attacker to filter and narrow its search to the most relevant sites that will contain the information he's looking for.
One of the most common tools used is the **Google Hacking Database**, that provides many of the best Google search string used by hackers to dig up information on the web.

Another operation that is usually performed on websites' document is to analyze their hidden *metadata*, which may lead to discover information such as:

- usernames
- server names and folders
- passwords
- operating system

Programming forums also offers opportunities for *social engineering attacks*, where an attacker could impersonate a friendly user in need of help to gain more precise information about a certain application usage.

STEP 4: WHOIS AND DNS ENUMERATION

Internet infrastructure is centrally managed to ensure interoperability, prevent IP conflicts and ensure universal resolvability across geographical and political boundaries.
The core functions of the internet are managed by a non-profit organization called **ICANN**, which assigns the following identifiers that must be globally unique for the internet to function:

- Domain Names
- IP addresses
- Protocols' parameters and ports number

Furthermore, it coordinates the stable operation of the *root DNS system*.

Even with all this centralized management in place, mining for this information is very difficult since the actual data is spread across the globe among numerous *WHOIS* servers.
The infrastructure of these system has a very high-level of security in place, to a degree where each server may use its own syntax to perform queries (assuming you can access the data requested).
Usually, domain-related items (such as the name of the site) are registered separately from IP-related item (such as the range of IP assigned): for this reason, we can define two different paths for finding these details.

Domain-Related Searches

The first step of this methodology is to determine which of the many *WHOIS* server contains the information we're looking for.
The general process defines the concept of **"Three R"**, which can be described as follows.
First, the authoritative *Registry* for a certain top-level domain (ex. ".com") contains information about which *Registrar* the target entity registered its domain with.
Then you query the appropriate *Registrar* to find the *Registrant* details for that particular domain name you're looking for.
The R*egistrant's* details are usually the juiciest, contain information such as name, physical addresses, phone numbers, e-mails, DNS server names, IPs and so on.

Since information are hierarchically stored, the best pattern is to start from the top of the tree with a top-down approach.

IP-Related Searches

The WHOIS server cannot directly expose information about an IP address, but it shows which organization is managing the range of IPs in which the queried address belongs to.
From this, we can backtrack all the companies that are managing that IP address until we reach the owner or (at least) some point of contact that has certain information.

In modern times, hackers masquerade their legitimate IP address, so the IP addresses that we see in logs are often untraceable.

Public Database Countermeasures

Much of the information stored in the aforementioned databases are for public disclosure, however security considerations should be employed to make the job of an attacker more difficult.
Oftentimes, an ex-administrative that leaves the organization is still able to change the domain information stored on the servers, and this is because the authorized personnel list is not updated frequently internally.
Ensuring that the information listed are accurate is a must, to ensure that attackers are not able to contact an ex-employee with still too much power on their hands.

Another solution is to pay companies to hide them, meaning that the external organization's contacts will appear as listed in the database effectively making the information anonymous, and they can also be used as an ulterior defense and alert the main companies in case of strange calls or e-mails.

Lastly, companies need to be sure that the **update policies** of this sites are strong enough so that an attacker cannot forge e-mails or impersonate other people and *hijack* the domain.

STEP 5: DNS INTERROGATION

After identifying all the associated domains, the next step is to start querying the DNS server because, if configured incorrectly, an attacker might discover very sensible information about the organization.

One of the most serious errors a sysadmin can make is allowing untrusted users to perform a **DNS zone transfer**, which essentially means making a hard copy of the organization's DNS onto a secondary server owned by the attacker. Unfortunately, this operation is not usually blocked in organizations that have both *primary* and *secondary DNS* servers, since it is an operation usually performed by a *secondary server* to update its content and make it on par with the *primary* one.

Although this isn't necessarily bad since DNS servers usually contain information about the active systems that are browsing the internet, it becomes very dangerous if the organization did not implement a DNS mechanism that distinguish between external DNS information (which are public) and internal DNS information.
In this second case, all IP addresses and hostnames of the organization's internal network may be disclosed to an attacker, essentially giving it the entire map of the organization's network.

One of the tools that can be used to perform such attack is **nslookup**, usually provided in UNIX and Windows OS.

DNS Security Countermeasures

DNS enumeration provides a plethora of data to attackers, so reducing the amount of information available is important.
One of the simplest defense mechanisms is to create a whitelist of IP addresses that are authorized to perform such operations, so that untrusted users cannot obtain a local copy of its content.

On the network side, using firewalls or packet-filtering policies can limit unauthorized inbound TCP connections on port 53.
However, this operation is a violation of the RFC standard, so implementing cryptographic authentication might be a better solution.

Finally, even if you are able to block any king of zone transfer, name lookups cannot be blocked and an attacker could manually perform reverse lookups against every IP addresses it knows.

STEP 6: NETWORK RECONNAISSANCE

Once the potential networks have been identified, we can attempt to determine their topology as well as potential access path.

The main tool used in this phase is **traceroute**, that lets you view the route that an IP packet follows toward its path to the destination.
By looking at the exact path that packets are taking, we can determine the network topology and also any *access control device* (ex. firewalls of packet-filtering routers) that may be filtering the traffic.
Obviously, this task may be very tedious for the following reasons:

- each node can have multiple routing paths
- each interface can have a different ACL implementation
- sending UDP packets, ICMP packets or any other type of packets may result in very different outputs

Network Reconnaissance Countermeasures

Implementing and (correctly) configuring an **Intrusion Detection System** helps in detecting this type of operations.
Also, configuring border routers to limit ICMP and UDP traffic to specific systems can minimize the internal network exposure.

Chapter 2: Scanning

The concept of **scanning** is equivalent to inspecting walls, doors or windows as potential *entry points*.
During *footprinting* we obtained a list of IP network block or IP addresses using various techniques, and now we want to determine the following information:

- which system is *alive*, meaning it is listening for incoming network traffic
- which services are currently running on the system
- which OS is used on the system

One of the main obstacles that we'll encounter during this phase are **firewalls**, systems designed to apply *packet-filtering rules* to drop malicious traffic, but some of these information can be retrieved completely anonymously via **passive scanning**.

DETERMINING IF A SYSTEM IS ALIVE

Although we may have a list of IP ranges, we don't know if a certain IP has been assigned to a host and, furthermore, if that host is currently up and running.
To detect this, we will introduce some techniques that will help us in detecting an active system.

Network Ping Sweeps

The operation of **network pinging** is the act of sending certain types of traffic to a target and analyze the responses or the lack of them.
We can define different types of *pinging*, depending on the type of traffic used to perform the scan.

- **ARP Host Discovery**
  The **Address Resolution Protocol** translates a system's MAC address to the IP that will be assigned to it.
  To discover another host, the system has to send some sort of ARP_request in the network and wait for a response from the destination host, which will then contain its IP address.
  If an attacker is positioned on the same *network segment* of the target, it can send an ARP_request to all hosts on the subnet, and every host that responds with an ARP_reply is considered to be alive.
  This technique is also powerful, because it identifies hosts that have a local firewall in place and are filtering higher layers traffic (TCP, ICMP, etc…).
  Some tools that can perform this type of operation are:
  - *arp-scan*
  - *nmap*
  - *Cain*

- **ICMP Host Discovery**
  The **Internet Control Message Protocol** was created to (legitimately) identify if a system on a network is alive and reachable, so it provides a variety of message types to help diagnose the status of a host and its network path.
  Traditionally, the term *pinging* refers to the process of sending ICMP ECHO_REQUEST packets to a system in hope of a ICMP ECHO_REPLY response, which indicates that the target is alive.
  Two other notable message types are ICMP TIMESTAMP, which can be used to identify the current system time of the target, and ICMP ADDRESS MASK, which can be used to identify the local subnet mask of the target.
  Some tools that can perform this type of operation are:
  - *ping*
  - *nmap*
  - *nping*
  - *SuperScan*

- **TCP/UDP Host Discovery**
  Although ICMP can provide valuable information to an attacker, it is also extremely useful for troubleshooting purposes by network engineers.
  In the real world, networks usually are configured so that ICMP packets are allowed only if generated and scoped *internally*, otherwise they are blocked: if a network limits ICMP, the next approach an attacker can take is to identify live hosts via **TCP/UDP packets**.
  Usually, servers (firewalled or not) need this type of traffic so that they can perform their functions and have some ports that are open to connections, but blindly probing different ports is very noisy and time consuming for the attacker.
  Desktops, on the other hand, often do not accept inbound connections and have local firewalls enabled by default, making them hard targets.
  Some tools that can perform this type of operation are:
    - *nmap*
    - *SuperScan*
    - *nping*

Network Ping Sweeps Countermeasures
Although this technique may not seem dangerous, it's important to **limit** incoming traffic or, at least, be able to **detect** this kind of activity.

When talking about *detection*, it's crucial to understand when an attack may occur and track down the attacker.
The primary method for detecting malicious traffic is using Network-based IDSs like **Snort**, which analyze and log network activity and is able to recognize particular patterns of traffic that may be correlated to network reconnaissance by an attacker.

When talking about *prevention*, we need to study all the traffic that is being exchanged in the network and define rules that will help us in deciding whether a packet (incoming or outgoing) should be allowed or dropped.
In particular, ICMP traffic should be carefully analyzed since it is one of the main sources of information used by attackers.
Most routers do not allow all types of packets, but we show a short list of packets that could be allowed to minimize the leak of information:

- `ICMP ECHO_REPLY`
- `HOST_UNREACHABLE`
- `TIME_EXCEEDED`

In addition, ICMP traffic can be limited with the use of **Access Control Lists** to your ISP's specific IP address, meaning that the ISP monitor the connectivity making it more difficult to perform this technique.

DETERMINE WHICH SERVICES ARE RUNNING ON THE TARGET
Now that we have identified which hosts are currently up and running, we are ready to begin *probing* each system to identify which ports and services are available to attack.

Port Scanning
The operation of **port scanning** is the process of sending packets to TCP and UDP ports on the target to determine which services are running or are in a *listening* state.
Identifying ports is critical to determine possible vulnerabilities present on the system, especially if the attacker is able to retrieve OS installed and the version of the services currently running.
Since every system has 65536 ports available, scanning all of them is very slow and may expose the attacker if the operation is performed too aggressively, so usually only a subset of them are scanned and all of them are related to the most common services and applications that can be found in real-world applications.

For the following steps we assume systems that are alive, and the objects are the following:

- identify all the TCP and UDP services that are running on the target
- identify which OS is being used
- identify applications' versions

TCP Connection scan
This type of scan is identical to a legitimate TCP connection, performing the full *3-way handshake* protocol.
Since it uses the full connection paradigm, it takes longer than most of the other scan types and it is also more likely to be logged on the target system.

TCP-SYN scan
Also called *half-open scanning*, it does not perform a full TCP connection but only the `SYN` packet is generated and sent to the target.
This type of connection is stealthier that a full connection, and it may not be logged, but it can cause a DoS on the target system if too many half-connections are opened.
Depending on the response received, we can determine the port's state:

- SYN/ACK: the port responded with the second part of a full TCP connection, therefore is active and listening
- RST/ACK: the port responded by closing the incoming connection, therefore is active but not listening

TCP-FIN scan
This type of scan sends a `FIN` packet to the target, which is used when closing a connection, and the response (or the lack of it) can be used to determine if the port's state: the RFC 793 standard defines that the port must respond with a RST packet if the port is closed, otherwise the incoming packet is ignored.
Useful because `FIN` packets are usually allowed by all firewalls.

TCP-null scan
This type of scan sends a packet where all *flags* are turned off, which represent an illegal packet for the RFC 793 standard.
The behavior is identical to the scan above: RST packet if the port is closed, otherwise the incoming packet is ignored.

TCP-ACK scan
This type of scan sends an ACK packet to the target, and is useful to map out firewall rules: if the packet is accepted, it means that the firewalls has be configured to block only new connections and not connections that have already been established.
This also allows to determine whether the firewall is *stateless* or *stateful*.

TCP-RPC scan
This type of scan is specific to UNIX systems and sends an RPC packet to the target, it is used to detect and identify *Remote Procedure Call* ports and their associated program and version number.

TCP Xmas Tree scan
This type of scan sends a `FIN`, URG and PUSH packets to the target port, and the response (or the lack of it) can be used to determine if the port's state: the RFC 793 standard defines that the port must respond with a RST packet if the port is closed, otherwise the incoming packet is ignored.

TCP Windows scan
This type of scan may detect filtered/non-filtered ports on some system by using an anomaly in the way the TCP *windows size* is reported.

UDP Connection scan
This type of scan is identical to a legitimate UDP connection but, since it is a *connectionless* protocol, it is not very reliable.
An "ICMP port unreachable" message means that the port is surely closed, while (in theory) not receiving anything implies that the port is open.
Again, the accuracy of this technique is highly dependent on the network utilization (at the moment of the scanning) and filtering, also it is very slow if the target employs a very heavy packet-filtering mechanism.

FTP Bounce scan
Although probably unusable on the internet today, the *FTP bounce* attack uses an insidious method of laundering connections through an FTP server by abusing the support for "proxy" FTP connections.
It can be used to make *virtually untraceable* any request made on the internet and even bypass firewalls, making it very hard to track the attacker while also hiding its identity.

Out of all the scans aforementioned, certain IP implementations have the distinction of sending RST packets for all ports scanned, regardless of whether they are open or not.
However, *TCP-SYN Scan* and *TCP Connection Scan* should work against all hosts.

Some tools that can perform this type of operation are:

- *nmap*
- *SuperScan*
- *ScanLine*
- *Netcat*

Port Scanning Countermeasures
Preventing port scanning is very hard and, as before, we distinguish between the two aspects of *detection* and *prevention*.

When talking about *detection*, the primary method used is Network-based IDSs like **Snort**.
From a UNIX perspective, tools like **scanlogd** can detect and log such attack, while Windows has tools like **Attacker** that provides the same functionalities.
Obviously, we need to keep in mind that paying close attention to the network traffic and being able to recognize a pattern of port scanning is the most basic form of defense against attackers, but if we're using automated tools then they need to be configured in a not so strict manner that everything is blocked.
For example, an attacker could spoof the IP address of an innocent party, in which case the system either allows the connection or retaliates against an honest client.
Most firewalls can (and should) be configured to detect this kind of activity, some of them can even detect the stealth ones, therefore configuring the automated tool in the correct manner becomes a must.

When talking about *prevention,* although preventing someone from launching a port scan if borderline impossible, we can minimize the exposure by disabling all unnecessary services and blocking their startup files.
While this is easy in UNIX-like environments, for Windows is a bit harder due to the way it operates, as TCP ports 139 and 445 provide most of the native functionalities of the OS.

DETECTING THE OPERATING SYSTEM
Once we've identified which services are currently running on the target, to determine whether they have potential vulnerabilities to exploit, we need more information about the versions and which OS is based on.

Active System Detection
Specific OS information will be useful during our vulnerability-mapping phase.
The simplest way is to perform **banner-grabbing** techniques, which we will discuss later on, but for now we will show that the available ports could be enough to make a fairly correct guess on which OS is currently running.
Conversely, it is impossible do compromise the security of a remote service that is not listening.

For **Windows** systems, ports 135 (Microsoft End-point Mapper), 139 (NetBIOS) and 445 (Microsoft Server Message Block) are essential to its functionalities, as well as TCP port 3389 that is used for the *Remote Desktop Protocol*.

For **UNIX** systems, TCP port 22 (SSH) is a good indicator, but also other TCP ports like 111 (Portmapper), 2049 (Network File System) and any high number port > 32770 are used only on these systems.

Another technique is **stack fingerprinting**, that also allows to deduct the OS with high probability: essentially each vendor has its own *IP stack implementation*, which is very different from another because everyone interpret specific RFC guidance differently when writing their TCP/IP stack.

In the following scans we assume that there is *at least* one port open, otherwise the probability of guessing if fairly low.

FIN probe
When a FIN packet is sent to an open port, as mentioned before, the RFC 793 states that the correct behavior is to simply ignore the packet.
However, many *Windows* implementations respond with a FIN/ACK packet.

Bogus flag probe
Some *Linux* implementations, when receiving an undefined TCP flag in the header of a SYN packet, will respond with the same flag set in their response packet.
Other systems, instead, reply with an RST packet.

Initial Sequence Number sampling
We can observe how the ISN changes between new connections: by convention they should be random, as *Linux* does, but not all systems follow this convention:

- old *UNIX* systems increment it by a fixed amount
- *Windows* systems increment it periodically by a small amount
- other systems increment it by a random amount or even use a fixed one

Fragmentation monitoring
Some systems always set the "Don't fragment" flag, for performance reasons.

TCP initial window size
For some systems the initial window size is fixed, and this is one of the most effective tests to determine the OS.

ACK value
Some systems, when analyzing the ACK field of a packet, will respond with the same value incremented by one.
Other system, instead, reply with the same value.

ICMP error message quenching
The RFC 1812 defines that ICMP packets should be limited, as *Linux* does, but very few other systems follow this rule.
By sending UDP packets to some random high-numbered port, we can count the number of HOST_UNREACHABLE messages received within a give amount of time, and at the same time determine if UDP ports are open.

ICMP message quoting
Some *Linux* implementations repeat the packet (or part of it) that generated an ICMP error.

ICMP error message (echoing integrity)
Some systems alter the IP headers of the packet that generated an ICMP error.

Type of service
The "ICMP port unreachable" messages should have the TOS field sets to 0x0, but some *Linux* implementations use the value 0xC0.

Fragmentation handling
Systems behave differently when receiving overlapping fragments: some stacks overwrite the old data with the new data, others the contrary.

TCP options
All systems implement only a subset of the TCP options defined in RCF 1323, therefore analyzing them is another one of the most effective tests.

Passive System Identification
Actively sending packets to the target allowed us to uncover valuable information on which OS is currently running, but an IDS could have tracked the traffic and already reported us to the administrator.
Therefore, *active stack fingerprinting* is not one of the stealthiest techniques that an attacker will employ.

On the other hand, **passive stack fingerprinting** revolves around monitoring the network traffic originated by the target.
Various traffic characteristics can be used to identify an OS, but we will show us only attributes associated with a TCP/IP session:

- *TTL:* what does the OS set as the TTL on outbound packets?
- *Window size:* what does the OS set as the window size?
- *"Don't fragment" flag:* does the OS use this bit?

After acquiring this information, tools like **Siphon** are able to correlate the connection's parameters to operating systems.

Although this technique is effective, it does have some limitations: for example, some applications may build their own packets without using the same signature as the operating system.
Also, the attacker needs to be in a position of the internal network that is able to capture packets and hope that the target did not manipulate the connections' attributes.

System Detection Countermeasures
When talking about *detection*, the aforementioned port-scanning detection tools can be used to detect when an attacker is probing the system.

When talking about *prevention*, it's possible to alter OS's parameters to change one (or more) of the unique *stack fingerprint* characteristics, however this may adversely affect the functionality of the system.

A new concept that is arising these days is the concept of **Active Defense**.
By means of *neural networks*, a target machine is able to adapts itself according to what the attacker is doing, which may not block it forever but could be enough to assess the situation and respond accordingly.
Virtualization is one example, where the target machine changes periodically the OS emulated to confuse the attacker.

PROCESSING AND STORING SCAN DATA
Mapping a target can result in a large amount of data, which implies the requirement of an efficient method of management: the more efficient you are in storing and analyzing data, the faster you are able to compromise a large number of systems.

One of the most used tools when talking about cybersecurity is **Metasploit**, a framework that contains exploits, tools and attack management functionalities.
In particular, we are interested in the *PostgreSQL* database inside it, that will allow us to make specific queries on the data gathered from the scanning phase, like:

- listing all the active hosts and the OS they are running
- listing all hosts that have a particular service running
- etc…

Chapter 3: Enumeration

The concept of **scanning** is equivalent to inspecting walls, doors or windows as potential *entry points*.
During *footprinting* we obtained a list of IP network block or IP addresses using various techniques, and now we want to determine the following information: