Vulnerability Assessment
# Table of Contents

Vulnerability Assessment ........................................................................................................... 2

Methodology .............................................................................................................................. 2

Reading a Scope/Rules of Engagement ..................................................................................... 2

Reconnaissance .......................................................................................................................... 3

Passive Reconnaissance ........................................................................................................... 3

Active Reconnaissance ............................................................................................................. 7

OS Fingerprinting ..................................................................................................................... 8

Service/Software Specific Enumeration .................................................................................... 8

Vulnerability Discovery .............................................................................................................. 11

Manual Vulnerability Discovery ............................................................................................... 11

Automated Vulnerability Discovery .......................................................................................... 12

Vulnerability Research .............................................................................................................. 12

Searchsploit .............................................................................................................................. 13

Vulnerability Confirmation ....................................................................................................... 14

Metasploit – to Check Vulnerabilities ..................................................................................... 14

NSE – ‘Vuln’ Category to Check Vulnerabilities ....................................................................... 14

Report Writing .......................................................................................................................... 15
Vulnerability Assessment

A vulnerability Assessment (VA) is the process of discovering weaknesses and misconfigurations on a target machine – followed by developing a detailed report, which will be of high value to the company hiring your services.

The report will outline the following:

- Vulnerabilities found
- Potential risks of associated vulnerabilities
- Affected systems
- How the client may remediate the issues found

Methodology

- **Reading a Scope/RoE**  
  How to plan your assessment based on supplied info

- **Reconnaissance**  
  Passive & Active - Scanning, Footprinting, Enumeration

- **Vulnerability Discovery**  
  Detect vulnerabilities from gathered info & automated tests

- **Vulnerability Research**  
  What further information is required?

- **Report Writing**  
  It’s all about supplying this end product  
  Therefore, record all information as you go

Reading a Scope/Rules of Engagement

The company requesting a pentest will supply you with either a scope or a RoE, this will outline the purpose of the assessment and what is to be tested.

The main items you need to be aware of:

- The reason the client has requested the assessment. This will help you understand what information is important to the company and how to tailor your report to provide greater value infrastructure to be tested network infrastructure or web server, a firewall, server or router, UAT or production etc.

- What is in Scope?
  - Should clearly state what you can test and what methods are acceptable
  - Be aware this can sometimes be vague, and you may have to work with the company to establish this further

- What is out of Scope?
  - This can be more important than what is in scope as it states what is forbidden – either testing methods, attacks (e.g. Dos) or infrastructure etc.
Reconnaissance
Reconnaissance is the first initial hacking stage where you need to begin thinking like an attacker. In this stage you’ll be gathering information on the target machine that can be used later in discovering vulnerabilities and attacking their system.

Reconnaissance is broken into two phases:

- **Passive Reconnaissance**  
  Reconnaissance without engaging the system
- **Active Reconnaissance**  
  Reconnaissance through interaction with the system

  - Footprinting: Create a network map - open ports, services & versions
  - Enumeration: Gain further info from individual services - i.e. username/groups, PWDs

Passive Reconnaissance
Passive reconnaissance is a very important step during the information gathering process. Passive reconnaissance is the ability to gather information about the target without interacting in any way with the target – in this way they are not alerted to an impending attack. As a pentester this is vital as it is important for an organization to understand what information can be gathered by an adversary without the company even being alerted as to the possibility of an impending attack. Furthermore, information gathered during the passive reconnaissance phase could be critical to exploiting a system. In this way a determined adversary could gather critical information and plan an attack allowing them the ability to get into the system, collect critical information and get out undetected in a very small time frame due to passive reconnaissance.

Passive reconnaissance relies on the public availability of information which can come from search engines and archived information.

Search Engines & Google Dorking
Search engines can be heavily utilized to find company specific information applicable to a pen test.
**Google dorking** is whereby you can use specific keywords to assist in your search queries. By adding keywords into your search queries, you can maximise your chance of discovering specific outputs.

**A list of these keywords are as follows:**

*Simple Google Dorks Syntax*

- **site** – will return website on following domain
- **allintitle** and **intitle** – contains title specified phrase on the page
- **inurl** – restricts the results contained in the URLS of the specified phrase
- **filetype** – search for specified filetype formats

See the images below:

![Google Dorking Example](image)

**For example**, sometimes metadata is contained in documents created internally by company staff. Such metadata could potentially include such things as employee names, internal IP addresses or software versions. Sometimes these company documents are available via public search engines either intentionally or unintentionally, it is just a matter of finding them. In this way google dorking keywords can be used to find this information – using such keywords as ‘ext’ or ‘filetype’ and ‘site’.

In this way you could use these keywords as follows:

**Discovering company emails**

Company emails can be found and used for phishing/spear phishing/waling emails. Emails can also be used to find personal details of company employees on other websites whereby you can gather information to formulate password wordlists to bruteforce company login pages. Sometimes IT professionals within a company even post code or information pertaining to the inner workings of system components on sites like GitHub – even using their company email within the code itself.
Theharvester

Theharvester is a tool used to crawl search engines looking for email address belonging to a specific domain.

theharvester Syntax:

```
Theharvester -d <domain name> -l <number of search results it will crawl through> -b <the search engine to search through> -h <output file name>
```

Once you have a list of company emails you can then get creative – you can search for social media accounts using the discovered company emails.

Recon-ng

Recon-ng is another versatile reconnaissance tool which can be used to aid in passive reconnaissance of an organization. Using recon-ng you can find social media accounts linked to employees of the company, email addresses of employees.

Shodan

Shodan (www.shodan.io) is a website that publicly displays information pertaining to every internet connected device. You can search for devices by their ip address, specific services running on connected devices, or even just connected devices registered to a given company.

As an example here is a screen shot of shodan’s result for google’s public dns server residing on the IP address 8.8.8.8.

As you can see it gives you a wealth of information including the device location, ports open and services running on open ports. You can use Shodan’s search field to query and filter results based on specific characteristics.
Basic keywords to be used in a shodan query

City: Find internet connected devices in each city
Country: Find internet connected devices in each country
Geo: Find internet connected devices for a given longitude and latitude
Hostname: Find internet connected devices with a given hostname
Net: Find internet connected devices within an IP range (can use CIDR notation)
OS: Find internet connected devices running a given operating system
Port: Find internet connected devices with a given port open

Shodan only provides a limited number of pages of results when using a free account. However, when using a paid account, you have unlimited access to Shodan’s results and services.

Shodan’s other services include:

Honeypot or Not – checks if a provided IP is a honeypot

Shodan Maps – provides a map in which you can view the location of your search results. The below example shows internet connected devices in and around Brisbane Australia running a Windows operating system.
**ScanHub** – Upload nmap scans (must be xml format) to shodan to generate reports and take advantage of shodan’s features for a specific subset of data.

**Shodan CLI** – use shodan directly from your Kali terminal.

**Shodan ICS Radar** – shows information about internet connected industrial control systems around the world.

**Shiptracker** – Track internet connected ships (using VSAT antennas) in almost real time as it scans the internet.

**Shodan Images** – screenshots of crawled devices – usually login screens or webcam images.

---

**Active Reconnaissance**

**Footprinting with Nmap**
Nmap is the most common host discovery and port scanner with its versatility expanding into service identification, OS identification, brute forcing and vulnerability discovery.

Nmap does have limitations; when no ports are specified it only scans the top most common 1000 ports. Therefore, it is essential to conduct a scan whereby specifying a large port range especially when there is a possibility of proprietary software being run on the target device. In some instances, nmap does not have the service information for a given port in its database or the port might be being used for something other than the registered service within nmap’s database. Therefore, it is always a good idea to manually check the services running either with netcat or wireshark. Another limitation is that UDP scanning can be unreliable with devices such as firewalls dropping ICMP packets resulting in false positives whereby you may return a large amount of open UDP ports and services that are actually not running.

**Nmap Syntax:** nmap <flags> <ip address>  
e.g. nmap –sS –sV –O 192.168.5.131

**View the Nmap video and cheat sheet for a list of commands.**

**Masscan**
Masscan is much quicker than nmap – it is the fastest port-scanner available and can scan the entire internet in 6 minutes. It is ideal for large IP/port ranges but may not be as accurate as nmap.

**Unicornscan**
Unicornscan is similar to masscan in how quick it scans. It is ideal for fast UDP scans – compared to nmap which may take a LONG time to complete a full UDP scan.
Zenmap
Zenmap provides a graphical interface for nmap scans. You can either import an nmap scan in xml format or you can scan directly with zenmap. This is great for larger networks with an increased range of hosts as you can quickly get an idea of low hanging fruit on the network.

OS Fingerprinting
Discovering which OS a target is running can greatly assist in the overall exploitation of a target – both in vulnerability discovery but also during exploitation when crafting OS specific payloads.

Nmap can be used to discover the OS of a target using the -O parameter. Nmap identifies the OS by looking at packets received from the target device. As each OS implements the TCP/IP stack in different ways nmap can quickly determine the OS by detecting patterns pertaining to a given OS. You can also use tools such as P0f.

Service/Software Specific Enumeration
Enumerating services and software versions can lead not only to discovering publicised vulnerabilities with a target system, but also highlights key misconfigurations of running services. The information gathered here will be also be utilised during a pentest e.g. creating a unique username wordlist.

Enumeration Tools Covered in the Videos
- Nmap Scripting Engine (NSE)
NSE is an extension of Nmap which offers a range of functionality from enumeration, vulnerability discovery and brute force attacks. NSE scripts are divided into the below categories.

  - **NSE Category Scripts Syntax**
  nmap --script <category name> <ip address> **Note
  – for more NSE commands view the NSE Cheat Sheet.

Categories to use during a vulnerability assessment:
- **auth** Returns info on user credentials and how to bypass them (doesn’t brute)
- **default/-sC** Contains scripts which are: useful, private, non-intrusive, reliable and fast
- **discovery** Network info gathering by querying external public registries
- **safe** Non-intrusive scripts, likely won’t cause adverse reactions e.g. DoS
- **vuln** Searches db for known vulnerabilities based on services running
Categories to use during a pentest:

- **broadcast** Broadcasts on local network to discover hosts not listed
- **brute** Brute forces authentication credentials - save time by searching protocols
- **exploit** Scripts which aim to exploit vulnerabilities
- **malware** Detects if target is infected by malware or backdoors

Categories which require extra consideration before use:

- **dos** Scripts from all categories which may cause denial of service
- **external** Scripts which send data to a third-party db – be careful in what data is sent
- **fuzzer** Fuzz a server with unexpected/randomised fields in each packet – slow and can crash the server
- **intrusive** Scripts which use significant target resources and likely to crash the server

- **Nikto**

Nikto is an automated web server vulnerability scanner – however I include it here as it offers a range of other functionality. Nikto returns items such as potential vulnerabilities, misconfigurations, files and directories that may be of interest and outdated software.

  **Nikto Syntax:** nikto -h <IP address>

- **Enum4Linux**

Enum4linux is a great tool for enumerating Windows and Samba hosts, and is essentially a wrapper of smbclient, rpcclient and nmblookup. It again returns a range of results including usernames, domains, shared files and directories and password configurations.

  **Enum4Linux Syntax:** enum4linux <IP address>

**Other Enumeration Tools**

As there are many different services that can be enumerated there are also many different tools to enumerate them – the below table outlines some commond services/protocols and associated enumeration tools.
<table>
<thead>
<tr>
<th>Service/Protocols</th>
<th>Tools</th>
<th>Metasploit auxiliary modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMB (Server Message Block)</td>
<td>Nbtscan, Nmap, enum4linux, nmblookup, smbclient, rpool, ridenum,</td>
<td>auxiliary/scanner/smb/smb_lookupsid</td>
</tr>
</tbody>
</table>
| SNMP                  | Nmap, onesixtyone, snmpwalk, snmpcheck, snmpenum | auxiliary/scanner/snmp/snmp_enum
 auxiliary/scanner/snmp/snmp_enumusers
 auxiliary/scanner/snmp/snmp_login |
| LLMNR /NBT-NS Spoofing | Responder.py                        | auxiliary/spoof/llmnr/llmnr_response
 auxiliary/spoof/nbns/nbns_response
 auxiliary/server/capture/smb
 auxiliary/server/capture/http_ntlm |
| SMTP                  | Smtp-user-enum                      | auxiliary/scanner/smtp/smtp_enum
 auxiliary/scanner/smtp/smtp_version
 auxiliary/server/capture/smtp |
| FTP                   | ftp, ftp-user-enum                  | auxiliary/scanner/ftp/anonymous
 auxiliary/scanner/ftp/ftp_login
 auxiliary/scanner/ftp/ftp_version |
| SSH                   | Ssh_enum                            | auxiliary/scanner/ssh/ssh_login
 auxiliary/scanner/ssh/ssh_login_pubkey |
| HTTP                  | Nikto, Zap, Dirbuster/gobuster, Burp, Tamper Data | auxiliary/scanner/http/http_version
 auxiliary/scanner/http/dir_listing
 auxiliary/scanner/http/dir_scanner_use
 auxiliary/scanner/http/enum_wayback
 auxiliary/scanner/http/files_dir
 auxiliary/scanner/http/http_login
 auxiliary/scanner/http/open_proxy
 auxiliary/scanner/http/options
 auxiliary/scanner/http/robots_txt
 auxiliary/scanner/http/verb_auth_bypass |
<p>| HTTPS                 | sslscan, nmap, testSSLServer        | auxiliary/scanner/http/ssl |</p>
<table>
<thead>
<tr>
<th>Service</th>
<th>Tools and Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPN</td>
<td>ike-scan, ikeforce</td>
</tr>
<tr>
<td>NFS</td>
<td>Rpcinfo, showmount, nfs-mount, nfspy</td>
</tr>
</tbody>
</table>

- It is important to note that the above table only lists basic generic enumeration tools and modules for each service. Tools and modules are available for version specific enumeration if the version of a service is known. For example, some openSSH versions can be enumerated using a Metasploit module (auxiliary/scanner/ssh/ssh_enumusers) because existing users take longer to return a “permission denied” error than for non-existing users. This time-based attack allows the simple enumeration of usernames for some openSSH versions.

Vulnerability Discovery

This is the phase we’ve been working towards, were we focus on discovering the target systems vulnerabilities. Discovery is divided into two sections – manual and automated discovery. Manual discovery relies upon your previous findings as certain vulnerabilities are only applicable in certain situations/combinations of the target systems services, versions and OS, and this is one reason why reconnaissance is such an important first step.

Manual Vulnerability Discovery

Manual discovery involves searching Google and vulnerability archives such as CVE details for published vulnerabilities and exploits related to your findings. To do this you can simply type the service, version and the word vulnerability into a Google search and view the results, a better method would be to visit one of the archives listed in the research section below (such as CVE details [https://www.cvedetails.com](https://www.cvedetails.com)), another option would be to use searchsploit at the command line (which is covered in the pentest section).
Automated Vulnerability Discovery

Nessus – is a powerful automated vulnerability scanner with an intuitive design. In the video course we download Nessus Home (free edition) this however is unfortunately restricted against commercial use. A free alternative would be Open Vas which comes loaded in the Kali repository.

OpenVas – automated vulnerability scanner, see below for installation commands.

    apt-get install openvas
    openvas-setup
    **verify openvas is running by typing: netstat -tulpn

Login to openvas at(using browser in Kali): https://127.0.0.1:9392

Nikto – as discussed in enumeration is an automated web scanner which discovers potential vulnerabilities, misconfigurations, files and directories that may be of interest and outdated software.

    Nikto Syntax: nikto -h <IP address>

OWASP Zap – is an automated web application vulnerability scanner. Whilst also offering other web app attacks e.g. fuzzing, spidering and directory brute forcing. And comes preloaded in Kali.

Vulnerability Research

This phase blurs with vulnerability discovery in that once you have gathered as much information in regard to all software components running on a device, and have discovered vulnerabilities through automated methods - you will need to further explore these vulnerabilities and possibly find other known vulnerabilities for the specific versions of software being used.

When a security researcher finds a vulnerability, they can publish the vulnerability and even a Proof Of Concept (POC) exploit for the vulnerability. A CVE identifier or “common vulnerability and exposures” identifier is assigned to each vulnerability that is discovered. When conducting a vulnerability assessment at a minimum it is your job to report if a company’s devices have any of these published vulnerabilities which could potentially allow an attacker to mandate an unintended response from the company’s systems.

Common online archives where you can search for published vulnerabilities include:

CVE details - https://www.cvedetails.com

Security focus – https://www.securityfocus.com/

Vulners - https://vulners.com/
All you need to do is go to the website and search for the software version you have found to check if there are any published vulnerabilities.

You can also search within Kali to find vulnerabilities for specific software versions – as it has an inbuilt database of exploit-db’s available exploits in which you can simply use from the command line (as opposed to downloading them directly from exploit-db) you use the tool searchsploit to access this database.

**Searchsploit**

In kali you can use searchsploit – a tool to discover published vulnerabilities on exploit-db. To do this in the kali CLI simply use the below command.

**Searchsploit Syntax:** Searchsploit <name of software>

As you can see in the screenshot above searchsploits alerts you to vulnerabilities based on software type and version. It also provides a path where kali houses the correlating exploits in which you can use to exploit these vulnerabilities.

It is important to understand that these vulnerabilities that you discover will fall into specific categories based on the way that you interact with the system to exploit them. Common vulnerability types can include:

- **Stack or Heap Based Buffer Overflows** – when a computer program writes data, beyond the confines of the “buffer” it has allowed for the given data, and therefore starts writing data in other memory locations.
- **Local and Remote File Inclusion Vulnerabilities** – where data can be input to a file or a file can be uploaded to the server.
- **Web Application Vulnerabilities** - XSS (cross site scripting), CSRF (Cross site request forgery), SQL Injection ect

Once it is clear the type of vulnerability discovered – that is understanding how the software is vulnerable and how it could be exploited it is important to understand what effect the vulnerability will have on a system. Exploited vulnerabilities can result in the below categories:

- **Denial of Service (DoS) conditions** – The system is brought down and may be unusable for some time.
• **Remote Code Execution** – A remote attacker can run arbitrary code on the system potentially leading to complete system compromise

• **Exposure of Sensitive Information** – critical information could be exposed to an attacker

Software misconfigurations also must be reported alongside published vulnerabilities including:

• **No or Limited Data Encryption**
• **No or Limited Authentication**
• **Exposed Critical Data**
• **Unrestricted File Upload or Data Input**

**Vulnerability Confirmation**

You’ve discovered the vulnerabilities, you’ve conducted your research, and you’re ready to write the report. However, there is one more step - Vulnerability Confirmation. Vulnerability confirmation methods are used to confirm that potentially important or critical vulnerabilities exist. However, these methods mostly rely on the vulnerabilities having published exploits and this is not the case for misconfigurations, poor design and implementation. Therefore, unlike the other steps confirmation isn’t always essential, and I’ve only chosen to include it for completeness, and for that reason it doesn’t form part of the methodology.

**Two main tools used for confirming vulnerabilities exist include:**

**Metasploit – to Check Vulnerabilities**

Metasploit has a **check function** for some exploits whereby before using an exploit against a target you can use Metasploit to check if the target is vulnerable. To do this you select the exploit in the usual way, set all options then instead of using the exploit or run commands type the command check. Metasploit usage will be covered further in the pentest module.

**note** - This will not run the exploit against the target but will check if the target is vulnerable.

**NSE – ‘Vuln’ Category to Check Vulnerabilities**

As covered above NSE can be used to check for vulnerabilities by using scripts found in the vuln category. However, it is good practice to do this either during the enumeration or discovery phases.
Report Writing

To write a report of high value, you need to be detailed, clear, and precise, and understand the motives of why the company requested a security assessment and what information they may find valuable — this information is gained both through clearly understanding the scope and during the vulnerability research phase for VAs or by understanding the exploitation methods and risks during a pentest.

Below is an overview of what is to be included in a report, the only difference between a VA report and a pentest report is the inclusion of the Proof of Concept (PoC) in the pentest report. The PoC should outline the methods you took to compromise the system as well as what was gained e.g. escalated privileges and data.

And remember that the majority of people who read your report won’t be technical so you still have to include the detailed technical information however your need to provide context to this so everyone can understand the importance of your findings.