

Desafío 2: Análisis forense digital

Tomas Santana - C.I. 30 604 530

2 de Julio de 2024

1 Introducción

Durante este desafío, se buscará analizar y resolver los retos dejados por el atacante en la máquina URU-Challenge-2, utilizando técnicas de análisis criptográfico y de análisis de tráfico de red. El presente informe documenta las acciones llevadas a cabo para resolver los retos propuestos.

2 Reconocimiento

2.1 Ingreso a la máquina

En este caso, la obtención de acceso fue muy sencilla, ya que se nos proporcionó con el usuario y contraseña de interés. Se estableció una conexión SSH con la máquina utilizando el comando que se muestra en la Figura 1.

Con esto, se obtuvo acceso a la máquina y se pudo comenzar a analizar los archivos y servicios que se encontraban en ella.

```
tomas@Laptop-Tomas:~$ ssh uruadmin@172.17.0.2
uruadmin@172.17.0.2's password:
Linux 1651ed40bfbe 5.15.133.1-microsoft-standard-WSL2 #1 SMP Thu Oct 5 21:02:42 UTC
2023 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
uruadmin@1651ed40bfbe:~$
uruadmin@1651ed40bfbe:~$ █
```

Figure 1: Conexión SSH a la máquina URU-Challenge-2

2.2 Exploración de archivos

Este paso también fue sencillo, ya que la descripción del problema indicó que los archivos de interés se encuentran en el directorio `/home/uruadmin`. Un simple comando `ls` nos permitió ver los archivos que se encontraban en el directorio, y se pudieron identificar cuatro archivos de interés. Se pueden ver los archivos en la Figura 2.

```
uruadmin@1651ed40bfbe:~$ ls
pista1.txt pista2 pista3.pcap pista4
uruadmin@1651ed40bfbe:~$
```

Figure 2: Archivos en el directorio `/home/uruadmin`

3 Análisis de pistas

3.1 Archivo 1: `pista1.txt`

Imprimiendo los contenidos de este archivo con el comando `cat`, se obtuvo el resultado que se muestra en la Figura 3. Se puede ver que el contenido es una cadena de texto cifrada. Como respeta una secuencia de caracteres similar a la del texto escrito (en cuanto a espacio y longitud de palabras), se puede inferir que es un tipo de cifrado clásico, como sustitución.

```
uruadmin@1651ed40bfbe:~$ cat pista1.txt

Rfghqvne ra HEH rf pbby.
Creb av pernf dhr ybf qrzáf ergbf freáa gna sápvyrf pbzb éfgr.
Fv gh grezvany qr péqhyn rf cne, ntertn ra ry vasbezr yn cnynoen pynir YNAPNFGRE
Fv cbe ry pbagenevb rf vzcne, ntertn ra ry vasbezr yn cnynoen pynir GBZNGR
```

Figure 3: Contenido del archivo `pista1.txt`

Por ser uno de los cifrados clásicos más comunes, se intenta descifrar el mensaje utilizando `rot13`, que puede lograrse fácilmente en linux con el comando `tr`. El comando utilizado será el siguiente.

```
cat pista1.txt | tr 'A-Za-z' 'N-ZA-Mn-za-m'
```

El primer argumento del comando `tr 'A-Za-z' 'N-ZA-Mn-za-m'` especifica que se aplicará a cualquier carácter alfabético en mayúscula o minúscula, y el

segundo argumento especifica que se reemplazará por el caracter que se encuentra 13 posiciones adelante en el alfabeto. El resultado de la ejecución de este comando se muestra en la Figura 4.

```
uruadmin@1651ed40bfbe:~$ cat pista1.txt | tr 'A-Za-z' 'N-ZA-Mn-za-m'

Estudiar en URU es cool.
Pero ni creas que los demás retos serán tan fáciles como éste.
Si tu terminal de cédula es par, agrega en el informe la palabra clave LANCASTER
Si por el contrario es impar, agrega en el informe la palabra clave TOMATE
```

Figure 4: Contenido del archivo `pista1.txt` decodificado

La decodificación de este archivo fue muy sencilla, ya que se utilizó un cifrado muy común y poco seguro. Estas técnicas criptográficas no se consideran nada seguras, y la información cifrada con estas es prácticamente pública. Siguiendo las indicaciones de la pista uno, se agrega la palabra indicada en este informe: LANCASTER.

3.2 Archivo 2: pista2

Se procedió a imprimir el contenido del archivo `pista2` y se obtuvo el resultado que se muestra en la Figura 5. Se puede ver que el contenido es una cadena de texto cifrada, pero no se puede inferir el tipo de cifrado utilizado con tanta facilidad.

```
uruadmin@1651ed40bfbe:~$ cat pista2

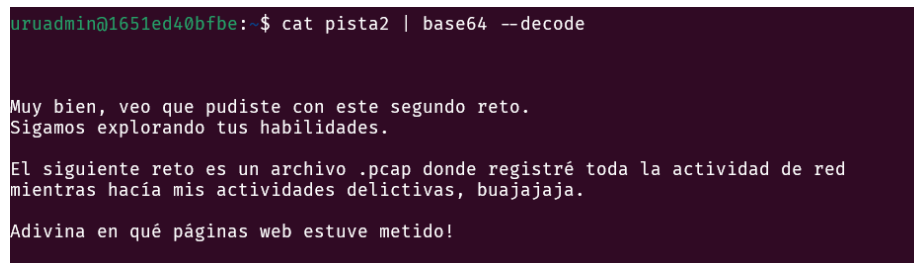
CgoKTXV5IGJpZW4sIHZlbyBxdWUgcHVkaXN0ZSBjb24gZXN0ZSBzZWd1bmRvIHJldG8uIApTaWdh
bW9zIGV4cGxvcmFuZG8gdHVzIGhhYmlsaWRhZGVzLgoKRWwgc2lnZWllbnRlIHJldG8gZXMgdW4g
YXJjaG12byAucGNhcCBkb25kZSB5ZWdpc3Ryw6kgdG9kYSB5YSBhY3RpdmlkYWQgZGUgcVkcmlp
ZW50cmFzIGhhY80tYSBtaXMyYWN0aXZpZGFkZXMgZGVsaWN0aXZhcycwYnVhamFqYWphLgoKQWRp
dmluYSB1biBxdC0pIHDDoWdpbmFzIHdlYiBlc3R1dmUgbWV0aWRvISAKCgo=
```

Figure 5: Contenido del archivo `pista2`

Esta secuencia de caracteres podría estar cifrada con algún tipo de algoritmo simétrico o asimétrico, para lo cual se necesitaría una llave de descryptación. Sin embargo, como la presentación del desafío estableció que la información necesaria se encuentra en el archivo `/home/uruadmin`, y siguiendo el orden de las pistas, aún no se ha encontrado ninguna llave, se procede a intentar decodificar el mensaje utilizando el comando `base64 --decode`, ya que no requiere de una llave de descryptación. El comando utilizado será el siguiente.

```
cat pista2 | base64 --decode
```

El comando `base64 --decode` decodifica la cadena de texto que se le pasa como argumento, y el resultado de la ejecución de este comando se muestra en la Figura 6.



```
urudadmin@1651ed40bfbe:~$ cat pista2 | base64 --decode

Muy bien, veo que pudiste con este segundo reto.
Sigamos explorando tus habilidades.

El siguiente reto es un archivo .pcap donde registré toda la actividad de red
mientras hacía mis actividades delictivas, buajajaja.

Adivina en qué páginas web estuve metido!
```

Figure 6: Contenido del archivo `pista2` decodificado

El resultado indica el objetivo para la pista tres, que será determinar los sitios web a los que accedió el atacante, y que están registrados en el archivo `pista3.pcap`.

3.3 Archivo 3: `pista3.pcap`

Como se conoce que el archivo `pista3.pcap` contiene un formato específico de captura de tráfico de red, utilizar `cat` para imprimir su contenido no será útil. En su lugar, se utilizará el comando `tcpdump` para analizar el tráfico de red y obtener la información necesaria.

Se quieren obtener los sitios web a los que accedió el atacante, por lo que se obtienen solo las peticiones HTTP que se realizaron en el archivo `pista3.pcap`. Utilizando el manual de `tcpdump` [1] podemos encontrar el siguiente comando, que nos proporciona el resultado que necesitamos.

```
sudo tcpdump 'tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2))
- ((tcp[12]&0xf0)>>2)) != 0)'
```

En este caso, el comando, el filtro `'tcp port 80'` se encarga de filtrar los paquetes que utilizan el protocolo TCP y el puerto 80 con IPv4, que es el puerto por defecto para el protocolo HTTP. El filtro `'(((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)'` se encarga de filtrar los paquetes que contengan datos. El resultado de la ejecución de este comando es extremadamente largo, por lo que se muestra solo una parte en la Figura 7.

En la salida, se puede observar que el atacante accedió únicamente a un sitio web alojado en la dirección IP `38.43.208.41`. Se puede intentar hacer una

```

urudadmin@1651ed40bfbe: $ sudo tcpdump -r pista3.pcap 'tcp port 80 and (((ip[2:2] - ((i
p[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) &#226; 0)'
reading from file pista3.pcap, link-type LINUX_SLL2 (Linux cooked v2), snapshot length
262144
Warning: interface names might be incorrect
01:31:11.133314 ?      Out IP 192.168.1.100.52954 > 38.43.208.41.80: Flags [P.], seq 431
091619:431092163, ack 2151262981, win 502, options [nop,nop,TS val 305666026 ecr 393428
761], length 544: HTTP: GET /educa/index.html HTTP/1.1
01:31:11.214919 ?      In  IP 38.43.208.41.80 > 192.168.1.100.52954: Flags [P.], seq 1:1
68, ack 544, win 248, options [nop,nop,TS val 393428842 ecr 305666026], length 167: HT
P: HTTP/1.1 304
01:31:11.239692 ?      Out IP 192.168.1.100.52970 > 38.43.208.41.80: Flags [P.], seq 377
399165:377399603, ack 4246650955, win 502, options [nop,nop,TS val 305666132 ecr 393428
802], length 438: HTTP: GET /educa/js/com/LoadCss.js HTTP/1.1
01:31:11.320972 ?      In  IP 38.43.208.41.80 > 192.168.1.100.52970: Flags [P.], seq 1:1
67, ack 438, win 249, options [nop,nop,TS val 393428949 ecr 305666132], length 166: HT
P: HTTP/1.1 304
01:31:11.321244 ?      Out IP 192.168.1.100.52992 > 38.43.208.41.80: Flags [P.], seq 419
2917423:4192917871, ack 3196946970, win 502, options [nop,nop,TS val 305666214 ecr 3934
28948], length 448: HTTP: GET /educa/js_portal/com/Flexivity.js HTTP/1.1
01:31:11.321338 ?      Out IP 192.168.1.100.52980 > 38.43.208.41.80: Flags [P.], seq 813
476742:813477198, ack 1886319346, win 502, options [nop,nop,TS val 305666214 ecr 393428
949], length 456: HTTP: GET /educa/js_portal/app/portal/bodyConfig.js HTTP/1.1
01:31:11.323062 ?      Out IP 192.168.1.100.53002 > 38.43.208.41.80: Flags [P.], seq 416
0500564:4160501018, ack 3757597106, win 502, options [nop,nop,TS val 305666215 ecr 3934
28949], length 454: HTTP: GET /educa/js_portal/app/portal/noticias.js HTTP/1.1
01:31:11.401749 ?      In  IP 38.43.208.41.80 > 192.168.1.100.52992: Flags [P.], seq 1:1
68, ack 448, win 249, options [nop,nop,TS val 393429028 ecr 305666214], length 167: HT
P: HTTP/1.1 304

```

Figure 7: Extracto del contenido de pista3.pcap

búsqueda inversa de la dirección IP para obtener el nombre del dominio, pero la máquina no cuenta con ninguna herramienta para lograr esto, como `nslookup` o `dig`. Sin embargo, se puede intentar acceder a la dirección IP en un navegador web para obtener el nombre del dominio. Al acceder a la dirección IP en un navegador web, se obtiene el resultado que se muestra en la Figura 8.

En este caso, no es necesario realizar una búsqueda inversa de la dirección IP, ya que se puede observar que es el sitio web del panel de estudiantes de la Universidad Rafael Urdaneta, alojado en `uru.insiemp.com`. El atacante estuvo accediendo a este sitio web para llevar a cabo sus actividades maliciosas. El atacante únicamente realizó peticiones HTTP GET, por lo que no se puede determinar con exactitud qué actividades realizó el atacante en el sitio web.

Sin embargo, realizando un análisis de tráfico al ingresar manualmente al sitio web, obtenemos un resultado muy similar al que se muestra en la Figura 8. Un extracto de este se muestra en la Figura 9. Esto indica que el atacante únicamente ingresó en la página principal del sitio web, y no realizó ninguna acción adicional.

3.4 Archivo 4: pista4

Se muestra el contenido el archivo `pista4` en la Figura 10.

Como se establece en el archivo, únicamente debemos generar información

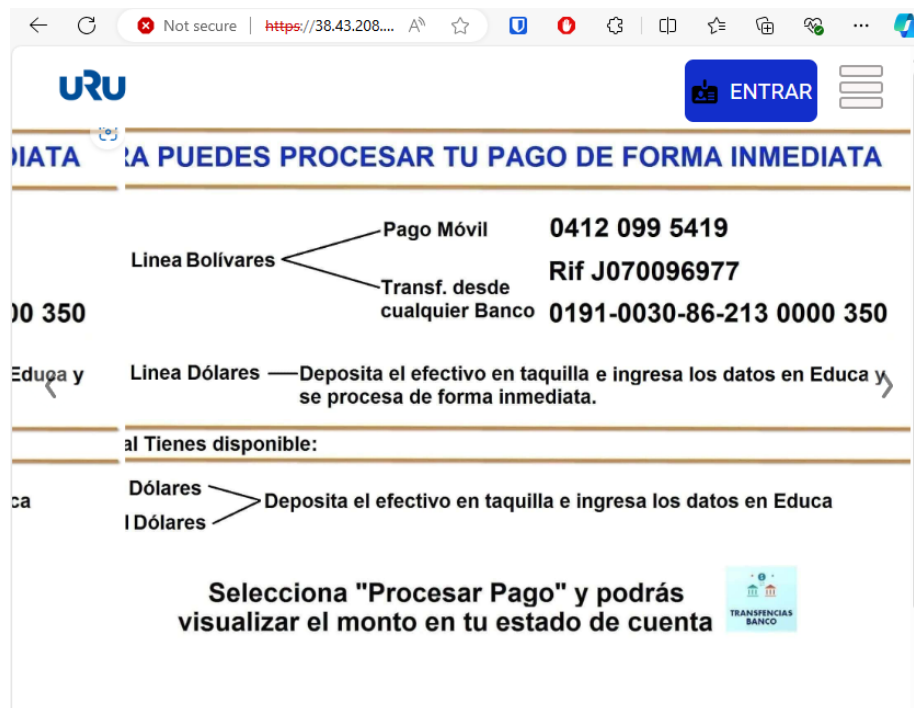


Figure 8: Acceso al sitio web alojado en la dirección IP 38.43.208.41

del estatus de los archivos de cada pista. Para esto, se utiliza el comando `stat` y se obtiene la información deseada. Las figuras 11, 12, 13 y 14 muestran la información de los archivos `pista1.txt`, `pista2`, `pista3.pcap` y `pista4` respectivamente.

4 Conclusión

Durante la ejecución de este desafío, se utilizaron distintas técnicas de análisis criptográfico y de análisis de paquetes para realizar un análisis forense digital de la máquina `URU-Challenge-2`. Uno por uno, se obtuvo la información contenida en los archivos, y se trazó la ruta de acción del atacante.

El primer archivo, el uso de un cifrado `rot13` hizo muy sencillo obtener la información necesaria. En la actualidad, las técnicas de cifrado clásicas no son seguras, y siempre se debe utilizar cifrados modernos y seguros, con una clave de cifrado robusta.

El segundo archivo, el uso de `base64` para codificar la información, también fue sencillo de decodificar. `base64` es un algoritmo de codificación muy común,

Name	Status	Type	Initiator	Size	Time
38.43.208.41	200	document	Other	3.6 kB	12 ms
extend-native-history-api.js	200	script	processor.js:2	2.8 kB	184 ms
requests.js	200	script	processor.js:2	5.7 kB	186 ms
LoadCss.js	200	script	(index):8	933 B	52 ms
Flexivity.js	200	script	(index):9	4.7 kB	63 ms
bodyConfig.js	200	script	(index):10	3.5 kB	76 ms
noticias.js	200	script	(index):11	2.8 kB	82 ms
location.js	200	script	content.js:2	7.3 kB	124 ms
page-script.js	200	script	page-script-append	26.4 kB	65 ms
vi-tr.js	200	script	content.js:1	2.5 kB	72 ms
imgButtonsCss.css	200	stylesheet	LoadCss.js:12	1.8 kB	141 ms
dialogsCss.css	200	stylesheet	LoadCss.js:12	2.1 kB	543 ms
buttonsCss.css	200	stylesheet	LoadCss.js:12	633 B	542 ms
calendarsCss.css	200	stylesheet	LoadCss.js:12	1.5 kB	771 ms
memoCss.css	200	stylesheet	LoadCss.js:12	1.3 kB	759 ms
editsCss.css	200	stylesheet	LoadCss.js:12	1.5 kB	826 ms

Figure 9: Acceso manual al sitio web alojado en la dirección IP 38.43.208.41

```

uruadmin@1651ed40bfbe: $ cat pista4

Para finalizar, solo debes generar información del status de los archivos de cada pista.
man stat te ayudará
No te olvides de agregar esa información al informe.

¡Éxitos!

```

Figure 10: Contenido del archivo pista4

```

uruadmin@1651ed40bfbe: $ stat pista1.txt
  File: pista1.txt
  Size: 255          Blocks: 8          IO Block: 4096   regular file
Device: 0,100   Inode: 60403       Links: 1
Access: (0664/-rw-rw-r--)  Uid: (  0/   root)   Gid: (  0/   root)
Access: 2024-06-28 00:27:46.000000000 +0000
Modify: 2024-06-28 00:27:46.000000000 +0000
Change: 2024-07-02 14:53:11.568444323 +0000
Birth: 2024-07-02 14:53:11.568444323 +0000

```

Figure 11: Estatus del archivo pista1.txt

y se utiliza para codificar información binaria en texto ASCII. Sin embargo, no es un algoritmo de cifrado, y no se debe utilizar para proteger información sensible.

El tercer archivo, el análisis de tráfico de red, fue un poco más complicado, ya que se necesitó utilizar `tcpdump` para analizar el tráfico de red y obtener

```

urudadmin@1651ed40bfbe:~$ stat pista2
  File: pista2
  Size: 375          Blocks: 8          IO Block: 4096   regular file
Device: 0,100   Inode: 60404       Links: 1
Access: (0664/-rw-rw-r--)  Uid: (   0/   root)   Gid: (   0/   root)
Access: 2024-06-28 01:23:47.000000000 +0000
Modify: 2024-06-28 01:23:47.000000000 +0000
Change: 2024-07-02 14:53:11.578444335 +0000
Birth: 2024-07-02 14:53:11.568444323 +0000

```

Figure 12: Estatus del archivo pista2

```

urudadmin@1651ed40bfbe:~$ stat pista3.pcap
  File: pista3.pcap
  Size: 40430906     Blocks: 78968       IO Block: 4096   regular file
Device: 0,100   Inode: 60405       Links: 1
Access: (0664/-rw-rw-r--)  Uid: (   0/   root)   Gid: (   0/   root)
Access: 2024-06-28 01:32:22.000000000 +0000
Modify: 2024-06-28 01:32:22.000000000 +0000
Change: 2024-07-02 14:53:11.958444767 +0000
Birth: 2024-07-02 14:53:11.578444335 +0000

```

Figure 13: Estatus del archivo pista3.pcap

```

urudadmin@1651ed40bfbe:~$ stat pista4
  File: pista4
  Size: 185          Blocks: 8          IO Block: 4096   regular file
Device: 0,100   Inode: 60406       Links: 1
Access: (0664/-rw-rw-r--)  Uid: (   0/   root)   Gid: (   0/   root)
Access: 2024-06-28 01:42:39.000000000 +0000
Modify: 2024-06-28 01:42:39.000000000 +0000
Change: 2024-07-02 14:53:11.958444767 +0000
Birth: 2024-07-02 14:53:11.958444767 +0000

```

Figure 14: Estatus del archivo pista4

la información necesaria. El manual de `tcpdump` [1] fue de gran ayuda para entender cómo utilizar la herramienta y obtener la información necesaria. La pista dos pidió obtener los sitios web a los que accedió el atacante, por lo que fue mucho más sencillo únicamente analizar las peticiones HTTP contenidas en el archivo `pista3.pcap`.

El cuarto archivo no contenía información relevante, y únicamente se pidió obtener el estatus de los archivos de cada pista. Para esto, se utilizó el comando `stat` [2] para obtener la información deseada.

Referencias

- [1] The Tcpdump Group. (2020). *tcpdump(1) - Linux*. Recuperado de <https://www.tcpdump.org/manpages/tcpdump.1.html>
- [2] Kerrisk, Michael. (2024). *stat(2) - Linux manual page*. Recuperado de <https://man7.org/linux/man-pages/man2/lstat.2.html>