

Info:

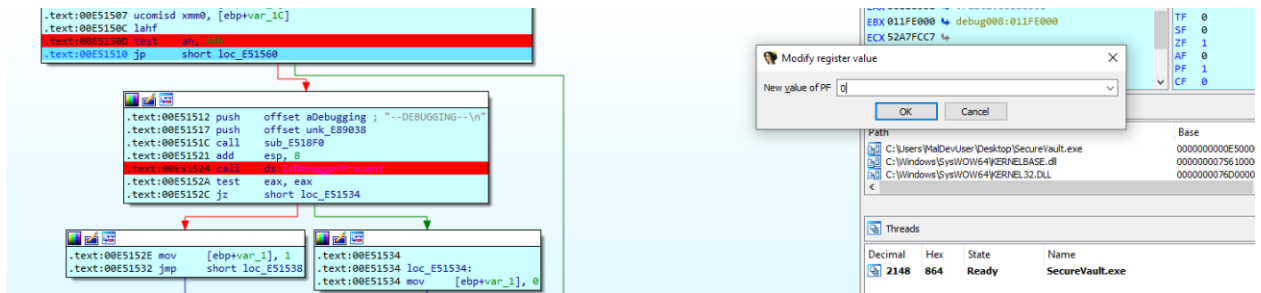
The Redteam found a seemingly very valuable binary. It looks like to be some sort of Password-Safe but we are missing the password

and need help to find a way into the vault. Can you help us?

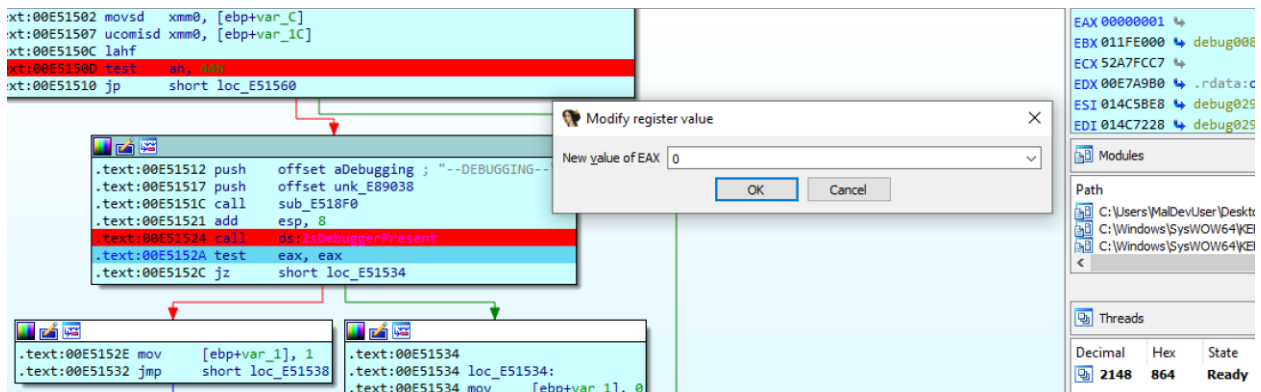
Task:

Find out what is inside the vault

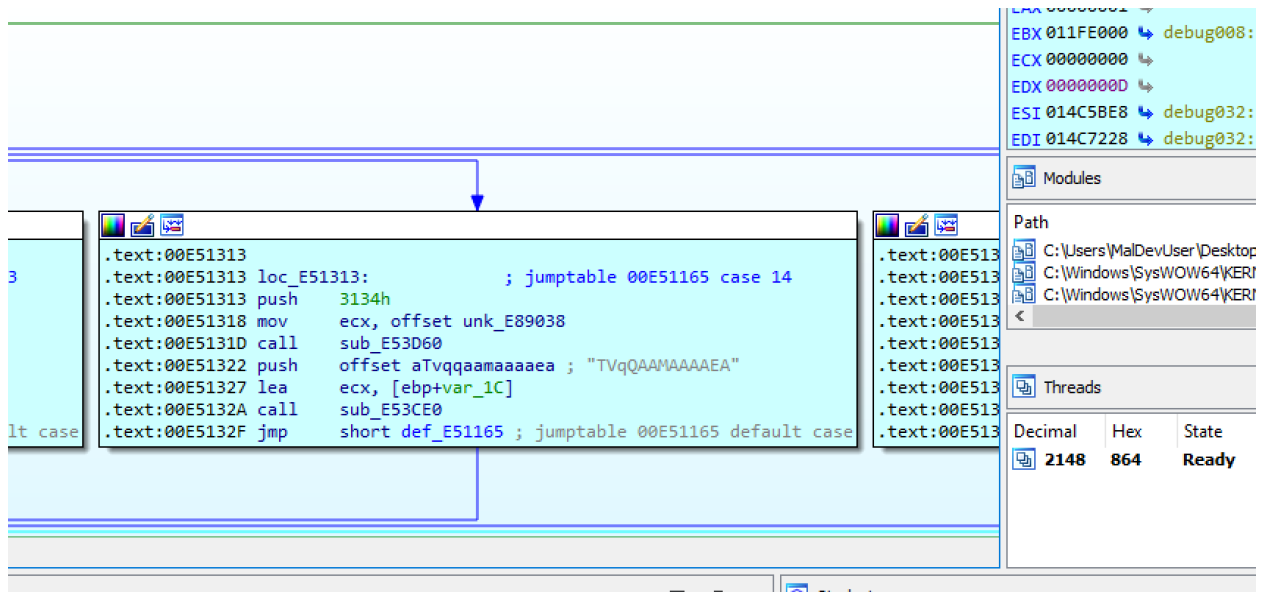
1. Simply checking the binary it looks like we want to end up on the left side where "IsDebuggerPresent" is located. By changing the register PF we can force the program to run the code on the left branch.



2. Next we want to get around the Debugger-Check by setting `eax` to 0



3. As we do not know which path is correct and will yield the flag we try just stepping through without changing anything. We do that until we end up at a bigger switch-case. The text in the console mentioned the key 14 so we want to try to select case 14. We can control the outcome of the switch-case by setting `edx` to 0xd



- Stepping through the rest of the code we can see the string "DEBUGGING INTERFACE" this seems like something we want to check out.

```
.text:00E513D2 lea     ecx, [ebp+var_4C]
.text:00E513D5 push    ecx
.text:00E513D6 lea     ecx, [ebp+var_34]
.text:00E513D9 call    sub_E54220
.text:00E513DE push    offset aDebuggingInter ; "\nDEBUGGING INTERFACE:\nSECRET VAULT CO"...
.text:00E513E3 push    offset unk_E89038
.text:00E513E8 call    sub_E518F0
.text:00E513ED add     esp, 8
.text:00E513F0 push    offset asc_E79B58 ; "\n"
.text:00E513F5 lea     edx, [ebp+var_34]
.text:00E513F8 push    edx
.text:00E513F9 lea     eax, [ebp+var_94]
.text:00E513FF push    eax
.text:00E51400 call    sub_E51C70
.text:00E51405 add     esp, 4
```

- After a few calls this will print out a very long code as "SECRET VAULT CODE"

```
12596
DEBUGGING INTERFACE:
SECRET VAULT CODE:
TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAAQAAAA
IGNhbm5vdCBiZSB5dW4gYW4gRE9TIG1vZGUuDQ
AABAAAAgAAAAgAABAAAAAAAAAAGAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAGAAAAwAAB4Pg
CCAAAEgAAAAAAAAAAAAAAAAAC50ZXh0AAAAyB8AAA
AABAAABALnJlbG9jAAAMAAAAAGAAAAACAAAAKA
AQAABgAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAooAgAABgoGclUAHAoEwAACgsHLA4ABigRAA
AQBwKBEAAAOAcnkBAHAoEQAACgByuWEAcGgRAA
rGcBAABKAAKQcPcAAQcTLFAAAcBAHAAcBAHAA
```

- Looking at the code this could be base64-encoded
- Seeing the MZ-header we can be sure that this is another binary so we copy the whole code and decode it and then save it as dump.exe


```
40         U2WFU1HB07XJ1dAHn1G1H225h1GFJ ;
41         string text3 = "delicious";
42         text3 = text2.Substring(1055, 36);
43         text3 = Encoding.UTF8.GetString(Convert.FromBase64String(text3));
44         string topping = "none";
45         try
46         {
47             topping = ares[0];
48         }
49     }
50 }
```

100 %

Locals

Name	Value	Type
System.Text.Encoding.UTF8.get returned	{System.Text.UTF8Encoding}	System.Text.UTF8Encoding
System.Convert.FromBase64String returned	{byte[0x0000001A]}	byte[]
System.Text.Encoding.GetString returned	"DS{T4a7_wa\$_pR337y_S1mP7e}"	string
args	{string[0x00000001]}	string[]
text	"CORRECT PASSWORD"	string
flag	false	bool
flag2	true	bool
text2	"ZWxlaWZlbmQgZG9uZWMgcHJldGI1bSB2dWxwdXRhdGUgc2FwaWVudG9uZS1mP7e"	string
text3	"DS{T4a7_wa\$_pR337y_S1mP7e}"	string
topping	null	string