MAR-10292089-1.v2 – Chinese Remote Access Trojan: TAIDOOR

us-cert.cisa.gov/ncas/analysis-reports/ar20-216a

Official website of the Department of Homeland Security

Notification

This report is provided "as is" for informational purposes only. The Department of Homeland Security (DHS) does not provide any warranties of any kind regarding any information contained herein. The DHS does not endorse any commercial product or service referenced in this bulletin or otherwise.

This document is marked TLP:WHITE--Disclosure is not limited. Sources may use TLP:WHITE when information carries minimal or no foreseeable risk of misuse, in accordance with applicable rules and procedures for public release. Subject to standard copyright rules, TLP:WHITE information may be distributed without restriction. For more information on the Traffic Light Protocol (TLP), see http://www.us-cert.gov/tlp.

Summary

Description

This Malware Analysis Report (MAR) is the result of analytic efforts between the Cybersecurity and Infrastructure Security Agency (CISA), the Federal Bureau of Investigation (FBI), and the Department of Defense (DoD). Working with U.S. Government partners, CISA, FBI, and DoD identified a malware variant used by Chinese government cyber actors, which is known as TAIDOOR. For more information on Chinese malicious cyber activity, please visit https[:]//www[.]us-cert.gov/china.

FBI has high confidence that Chinese government actors are using malware variants in conjunction with proxy servers to maintain a presence on victim networks and to further network exploitation. CISA, FBI, and DoD are distributing this MAR to enable network defense and reduce exposure to Chinese government malicious cyber activity.

This MAR includes suggested response actions and recommended mitigation techniques. Users or administrators should flag activity associated with the malware and report the activity to the Cybersecurity and Infrastructure Security Agency (CISA) or the FBI Cyber Watch (CyWatch), and give the activity the highest priority for enhanced mitigation.

Malicious binaries identified as a x86 and x64 version of Taidoor were submitted for analysis. Taidoor is installed on a target's system as a service dynamic link library (DLL) and is comprised of two files. The first file is a loader, which is started as a service. The loader decrypts the second file, and executes it in memory, which is the main Remote Access Trojan (RAT).

For a downloadable copy of IOCs, see MAR-10292089-1.v2.stix.

Submitted Files (4)

0d0ccfe7cd476e2e2498b854cef2e6f959df817e52924b3a8bcdae7a8faaa686 (svchost.dll)

363ea096a3f6d06d56dc97ff1618607d462f366139df70c88310bbf77b9f9f90 (svchost.dll)

4a0688baf9661d3737ee82f8992a0a665732c91704f28688f643115648c107d4 (ml.dll)

6e6d3a831c03b09d9e4a54859329fbfd428083f8f5bc5f27abbfdd9c47ec0e57 (rasautoex.dll)

Domains (2)

cnaweb.mrslove.com

infonew.dubya.net

IPs (1)

210.68.69.82

Findings

4a0688baf9661d3737ee82f8992a0a665732c91704f28688f643115648c107d4

Tags

backdoorloadertrojan

Details

Name	ml.dll
Size	43520 bytes
Type	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
MD5	6aa08fed32263c052006d977a124ed7b
SHA1	9a6795333e3352b56a8fd506e463ef634b7636d2
SHA256	4a0688baf9661d3737ee82f8992a0a665732c91704f28688f643115648c107d4
SHA512	179e9d9ccbc268cc94a7f6d31f29cf0f7a163db829a4557865f3c1f98614f94ceb7b90273d33eb49ef569cfc9013b76c7de32d7511639a7ab2c352f3137d51b6
ssdeep	768:uGRVnBnwS5kBKsl4anxKFhx3W3kGmifmUED7Bn5f6dBywFmZb:fDeSnbx3okvxVwFI
Entropy	5.864467

Antivirus

Ahnlab	Trojan/Win32.Agent
Avira	TR/Agent.aavma
BitDefender	Trojan.GenericKD.34284857
ClamAV	Win.Packer.Taidoor-9209869-0

Comodo	Malware
Cyren	W32/Trojan.DRSK-8300
ESET	a variant of Win32/Agent.ACFH trojan
Emsisoft	Trojan.GenericKD.34284857 (B)
Ikarus	Trojan.Win32.Agent
K7	Trojan (0056be3e1)
Lavasoft	Trojan.GenericKD.34284857
McAfee	RDN/Generic trojan.ks
Microsoft Security Essentials	Trojan:Win32/Taidoor.DA!MTB
NANOAV	Trojan.Win32.Dllhijacker.hqfyaa
Quick Heal	Trojan.Taidoor.S15351536
Sophos	Mal/Taidoor-A
Symantec	Trojan Horse
Systweak	trojan-backdoor.taidoor
TrendMicro	Trojan.2826E77D
TrendMicro House Call	Trojan.2826E77D
VirusBlokAda	Trojan.Dllhijacker
Zillya!	Trojan.Agent.Win32.1363180

YARA Rules

No matches found.

ssdeep Matches

No matches found.

PE Metadata

Compile Date	2019-01-03 07:16:12-05:00
Import Hash	dbb469cb14550e6085a14b4b2d41ede9

PE Sections

MD5	Name	Raw Size	Entropy
62ab3bae7859f6f6dc68366d283ad53e	header	1024	2.511204
63550f7c47453c2809834382e228637d	.text	23040	6.442964
a30bb3ac9b6694a8980c39c0267c9a83	.rdata	11264	4.926331

ad5814673b8579de78be5b6b929d2405	.data	3072	2.629944
619ecca9c8d1073a0b90f5fffac42ec8	.rsrc	512	5.105029
0f292021853e7ca76c4196bcbe9afdaf	.reloc	4608	3.712197

Packers/Compilers/Cryptors

Microsoft Visual C++ DLL *sign by CodeRipper

Relationships

4a0688baf9...	Used	363ea096a3f6d06d56dc97ff1618607d462f366139df70c88310bbf77b9f9f90

Description

This file is a 32-bit Windows DLL file. The file “ml.dll” is a Taidoor loader. The file utilizes the export function called “MyStart” to decrypt and load “svchost.dll” (8CF683B7D181591B91E145985F32664C), which was identified as Taidoor malware. Taidoor is a traditional RAT.

The “MyStart” function looks for the file name “svchost.dll” in its running directory. If that file is located, the DLL will read “svchost.dll” into memory. After the file is read into memory, the DLL uses a RC4 encryption algorithm to decrypt the contents of the file. The RC4 key used for decryption is, “ar1z7d6556sAyAXtUQc2”.

After the loader has finished decrypting “svchost.dll”, the loader now has a decrypted version of Taidoor, which is a DLL. The loader then uses the API calls GetProcessHeap, GetProcAddress, and LoadLibrary to load the following DLLs, KERNEL32.dll, ADVAPI32.dll, and WS2_32.dll, which Taidoor will utilize.

Next, the loader looks for the export “Start” in the Taidoor DLL and executes that function.

363ea096a3f6d06d56dc97ff1618607d462f366139df70c88310bbf77b9f9f90

Tags

remote-access-trojantrojan

Details

Name	svchost.dll
Size	158208 bytes
Type	data
MD5	8cf683b7d181591b91e145985f32664c
SHA1	f0a20aaf4d2598be043469b69075c00236b7a89a
SHA256	363ea096a3f6d06d56dc97ff1618607d462f366139df70c88310bbf77b9f9f90

SHA512	b75401d591caee812c5c1a669ce03c47f78f1c40a2fa31cf58a0318ffbfc032b82cb1b6d2a599ce1b3547be5a404f55212156640b095f895a9aac3c58ec4bad8
ssdeep	3072:fRxYk0d5+6/kdGyfitoxNsUZE2XZ+4Duz6fCKmjjwF5PaT:JqkoiGiZxE4qRKqgIT
Entropy	7.998691

Antivirus

Ahnlab	Data/BIN.EncPe
Antiy	Trojan/Win32.Taidoor
Avira	TR/Taidoor.BD
BitDefender	Trojan.Agent.EUMT
ClamAV	Win.Packed.Taidoor-9209834-1
Cyren	W32/Taidoor.A.enc!Camelot
Emsisoft	Trojan.Agent.EUMT (B)
Ikarus	Trojan.Win32.Taidoor
Lavasoft	Trojan.Agent.EUMT
McAfee	Trojan-Taidoor
Microsoft Security Essentials	Trojan:Win32/Taidoor.DC!MTB
Sophos	Troj/Taidoor-A
Symantec	Trojan Horse
TrendMicro	Backdoo.7F53B305
TrendMicro House Call	Backdoo.7F53B305
Zillya!	Trojan.Taidoor.Win32.6

YARA Rules

```
rule CISA_10292089_01 : rat loader TAIDOOR
{
  meta:
     Author = "CISA Code & Media Analysis"
     Incident = "10292089"
     Date = "2020-06-18"
     Last_Modified = "20200616_1530"
     Actor = "n/a"
     Category = "Trojan Loader Rat"
     Family = "TAIDOOR"
     Description = "Detects Taidoor Rat Loader samples"
     MD5_1 = "8cf683b7d181591b91e145985f32664c"
     SHA256_1 =
"363ea096a3f6d06d56dc97ff1618607d462f366139df70c88310bbf77b9f9f90"
     MD5_2 = "6627918d989bd7d15ef0724362b67edd"
     SHA256_2 =
"0d0ccfe7cd476e2e2498b854cef2e6f959df817e52924b3a8bcdae7a8faaa686"
  strings:
     $s0 = { 8A 46 01 88 86 00 01 00 00 8A 46 03 88 86 01 01 00 00 8A 46 05 88 86 02 01
00 00 8A 46 07 88 86 03 01 00 00 }
     $s1 = { 88 04 30 40 3D 00 01 00 00 7C F5 }
     $s2 = { 0F BE 04 31 0F BE 4C 31 01 2B C3 2B CB C1 E0 04 0B C1 }
     $s3 = { 8A 43 01 48 8B 6C 24 60 88 83 00 01 00 00 8A 43 03 }
     $s4 = { 88 83 01 01 00 00 8A 43 05 88 83 02 01 00 00 8A 43 07 88 83 03 01 00 00 }
     $s5 = { 41 0F BE 14 7C 83 C2 80 41 0F BE 44 7C 01 83 C0 80 C1 E2 04 0B D0 }
     $s6 = { 5A 05 B2 CB E7 45 9D C2 1D 60 F0 4C 04 01 43 85 3B F9 8B 7E }
  condition:
     ($s0 and $s1 and $s2) or ($s3 and $s4 and $s5) or ($s6)
}
```

ssdeep Matches

No matches found.

Relationships

363ea096a3...	Used_By	4a0688baf9661d3737ee82f8992a0a665732c91704f28688f643115648c107d4
363ea096a3...	Connected_To	cnaweb.mrslove.com
363ea096a3...	Connected_To	210.68.69.82

Description

This encrypted file has been identified as the Taidoor RAT loaded by “ml.dll” (6AA08FED32263C052006D977A124ED7B). After the loader has finished decrypting this file, the loader has a decrypted version of Taidoor, which is a DLL. The loader then uses the API calls GetProcessHeap, GetProcAddress, and LoadLibrary to load the following DLLs, KERNEL32.dll, ADVAPI32.dll, and WS2_32.dll, which this file will utilize.

Next, the loader “ml.dll” (6AA08FED32263C052006D977A124ED7B) looks for the export “Start” in the Taidoor DLL and executes that function. Taidoor’s “Start” function kicks off by decrypting a multitude of import strings that it will use to dynamically import functions from the DLLs that have been loaded. A complex stream cipher is used to decrypt the encrypted strings utilized by this malware. The 85 strings include APIs and strings used by other structures, such as a structure capable of allowing the malware to load external plugin payloads. The malware utilizes the following 7-byte key to generate a 256-byte initial stream cipher value: “19 34 F4 D2 E9 B3 0F”.

Next, the algorithm pads the 256 initial cipher value out to 260 bytes utilizing 4-bytes already contained within the 256-byte block (Figure 2). The algorithm performs the encryption 2-bytes at a time from the encrypted string blocks. It compresses the 2-bytes into 1 byte before the decryption process by subtracting the first byte and second byte by 0x80h. The result of the performing the subtraction on the first byte is then shifted left by four. Both values are then added together by using Boolean addition (OR) resulting in a single byte that is decrypted by the cipher.

Using a simple Exclusive OR (XOR) operation, the 260-byte block is shuffled and modified to produce the byte that is used to decrypt the newly compressed byte. The byte being decrypted is then placed back into the 260-byte cipher block buffer. This effectively produces a recurrent block shifting effect where the 260-byte cipher block value changes as a result of the sequence of bytes it receives. This is an effective method of thwarting heuristic or brute force attacks.

Taidoor also uses the AES algorithm to decrypt a "1616 byte" configuration file. This configuration file contains the command and control (C2) servers and possibly another encryption key used later. The AES key used in hex is, “2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C” IV: “00”.

--Begin C2--
cnaweb.mrslove.com
210.68.69.82
--End C2--

After completing this decryption function Taidoor iterates through the System Event Log. Looking specifically for event IDs 6005 (event service started) and 6006 (event service stopped). After completing its decryption functions, Taidoor tries to connect to its C2 server. Once Taidoor and the C2 server finish the TCP handshake, Taidoor waits for at least one byte of data to be sent from the C2 server. This byte or bytes are not checked by Taidoo, anything can be sent.

After Taidoor has confirmed it has received at least one byte of data form the server, Taidoor sends a custom formatted packet over port 443. Note: this packet does not follow TLS protocol, and is easily identifiable. The initial packet sent from Taidoor to the C2 server in this case always starts with “F::” followed by the encryption key that Taidoor, and the C2 server will use to encrypt all following communications.

After sending the encryption key to the C2 server, Taidoor expects the server to respond with “200 OK\r\n\r\n”. Note: This response is over port 443, but is not encrypted, it is sent in clear text.

After Taidoor has successfully connected to its C2, it creates a Windows INI configuration file, and copies cmd.exe into the system temp folder.

```
--Begin Windows INI file created--
C:\ProgramData\Microsoft\~svc_.TMp
--End Windows INI file created--
```

```
--Begin contents of INI file--
[Micros]
source=c:\temp\cmd.exe
--End contents of INI file--
```

Note: Taidoor does not have a function built it that enables it to persist past a system reboot. It appears from the memory dump of the infected system, it was installed as a service DLL by some other means.

The malware author never removed the symbol file for the “ml.dll” build. This artifact provides additional information that the malware author intended this binary to do, “DllHijackPlushInject”.

```
--Begin symbol file artifact--
c:\Users\user\Desktop\DllHijackPlushInject\version\Release\MemoryLoad.pdb
--End symbol file artifact--
```

The following IDA script can be used to decrypt all the encrypted strings and demonstrate how a sequence of bytes is encrypted utilizing the initial 260 byte cipher block generated from the key value “19 34 F4 D2 E9 B3 0F”:

```
--Begin IDA script--
import os
import sys
import idaapi
cwd = os.getcwd()
cwd = '/Users/terminator/PycharmProjects/rc4_test//'
cipherblock = []
pb_fname = cwd + "//" + 'pristine_block.bin'
es_fname = cwd + "//" + 'encrypted_strings.bin'
secure_strings_func = 0x10003cb7
encrypted_strings_block = 0x1001c434
enc_string_size = 2875
global_decrypted_stringz = []
try:
fh = open(pb_fname, 'rb')
read_bitez = fh.read()
fh.close()
except Exception as e:
print("Couldnt read filename. Reading from code (Attempt)")
print("Cipher Block len: " + str(len(cipherblock)))
for idx in read_bitez: # convert them to ords to do the math!
idx = ord(idx)
cipherblock.append(idx)
def decrypt(encrypted_string, cipherblock): # **CALL THIS FUNC to decrypt stuff!
```

```
string_len = len(encrypted_string)
string_len = string_len / 2
throttle = 0
da_string = ""
while True:
cipherblock, decoded_byte = decrypt_it(cipherblock, encrypted_string, throttle)
try:
charr = chr(decoded_byte)
if throttle:
da_string += charr
except Exception as e:
pass
throttle += 1 # INCREMENT before doing the compare
if throttle == string_len:
global_decrypted_stringz.append(da_string)
return da_string
def decrypt_it(cipherblock, encoded_data, throttle):
ebx = 128 # *0x80
ecx = throttle
ecx = ecx + ecx
eax = encoded_data[ecx]
ecx = encoded_data[ecx + 1]
eax = eax - ebx
ecx = ecx - ebx
eax = eax << 4
eax = eax | ecx
cipherblock, decoded_byte = outter_shuffle_func(cipherblock, eax)
return cipherblock, decoded_byte
def outter_shuffle_func(cipherblock, encoded_bite):
# before inner func
cipherblock = inner_shuffle_func(cipherblock)
# after inner func
eax = cipherblock[258]
ecx = cipherblock[eax]
eax = cipherblock[260]
eax = cipherblock[eax]
edx = cipherblock[257]
edi = cipherblock[256]
edx = cipherblock[edx]
edi = cipherblock[edi]
ecx = eax + ecx
eax = cipherblock[259]
eax = cipherblock[eax]
ecx = eax + ecx
eax = 255
ecx = ecx & eax
ecx = cipherblock[ecx]
cl = cipherblock[ecx]
```

```
edx = edx + edi
edx = edx & eax
cl = cipherblock[edx] ^ cl # **actual manipulation here
al = encoded_bite
cl = cl ^ al
cipherblock[260] = al
cipherblock[259] = cl
al = cl
decoded_byte = al
return cipherblock, decoded_byte
def wrap_around_strip(da_byte):
da_byte_str = str(hex(da_byte))
da_byte_str = da_byte_str.split("x")
da_byte_str = da_byte_str[1]
str_length = len(da_byte_str)
if str_length > 2:
got_em = "0x"
got_em += da_byte_str[str_length - 2]
got_em += da_byte_str[str_length - 1]
got_em = int(got_em, 16)
return got_em
return da_byte
def add_bites(a, b):
for_return = a + b
for_return = wrap_around_strip(for_return)
return for_return
def inner_shuffle_func(cipherblock_orig): # *SHUFFLE The cipher block here!
cipherblock = []
for idx in cipherblock_orig: # lets make a copy!
cipherblock.append(idx)
al = cipherblock[256]
esi = cipherblock[260]
dl = cipherblock[esi]
al = al & 0xffffff
edi = al
bl = cipherblock[edi]
da_byte = cipherblock[257]
da_byte = add_bites(da_byte, bl)
cipherblock[257] = da_byte
al += 1
cipherblock[256] = al
eax = cipherblock[257]
al = cipherblock[eax]
cipherblock[esi] = al
esi = cipherblock[259]
bl = cipherblock[esi]
edi = cipherblock[257]
cipherblock[edi] = bl
```

```
esi = cipherblock[256]
eax = cipherblock[259]
bl = cipherblock[esi]
cipherblock[eax] = bl
eax = cipherblock[256]
cipherblock[eax] = dl
eax = dl
al = cipherblock[eax]
temp_byte = cipherblock[258]
temp_byte = add_bites(temp_byte, al)
cipherblock[258] = temp_byte
return cipherblock
def decode_from_addr(target_addr, label_loc, pointer_addr, label_them):
init_bitez = []
ord_bitez = []
while True:
temp_bite = idaapi.get_byte(target_addr)
if not temp_bite:
break
init_bitez.append(temp_bite)
target_addr += 1
for idx in init_bitez:
ord_bitez.append(idx)
cipher_block_copy = []
for idx in cipherblock:
cipher_block_copy.append(idx)
dec_string = decrypt(ord_bitez, cipher_block_copy)
if label_them:
SetColor(label_loc, CIC_ITEM, 0xc7c7ff)
MakeComm(label_loc, dec_string)
SetColor(pointer_addr, CIC_ITEM, 0xc7c7ff)
MakeComm(pointer_addr, dec_string)
print(dec_string)
def find_initial_loc(target_addr):
addr = target_addr
give_up = 5
attempts = 0
while True:
addr = idc.PrevHead(addr)
if GetMnem(addr) == "push" and "off_" in GetOpnd(addr, 0):
string_addr = GetOperandValue(addr, 0)
print("Found String Loc: " + str(hex(string_addr)))
pointer_addr = idaapi.get_dword(string_addr)
print(hex(pointer_addr))
decode_from_addr(pointer_addr, addr, string_addr, 1)
return string_addr
attempts += 1
if attempts == give_up:
```

```
return 0
enc_stringz_data = []
try:
fh = open(es_fname)
da_data = fh.read()
fh.close()
for idx in da_data:
x = ord(idx)
enc_stringz_data.append(x)
except Exception as e:
print("Couldnt read encrypted strings file. Reading from Malware!")
addr_throttle = encrypted_strings_block
while len(enc_stringz_data) < enc_string_size:
x = idaapi.get_byte(addr_throttle)
enc_stringz_data.append(x)
encrypted_stringz = [] # *list of lists
temp_string = []
for idx in enc_stringz_data:
if idx:
temp_string.append(idx)
if not idx:
if len(temp_string):
encrypted_stringz.append(temp_string)
temp_string = []
decrypted_stringz = []
debug_it = False
if debug_it:
for enc_string in encrypted_stringz:
cipher_block_copy = []
for idx in cipherblock:
cipher_block_copy.append(idx)
dec_string = decrypt(enc_string, cipher_block_copy)
decrypted_stringz.append(dec_string)
print("----------------------")
for idx in decrypted_stringz:
print(idx)
print("Complete")
addresses_to = []
for addr in XrefsTo(secure_strings_func):
print("----------")
print(hex(addr.frm))
find_initial_loc(addr.frm)
print("----------")
print("\n")
addresses_to.append(addr.frm)
print("IDA IDB Labeled. Decrypted Strings Below:")
print("-----------------------------")
```

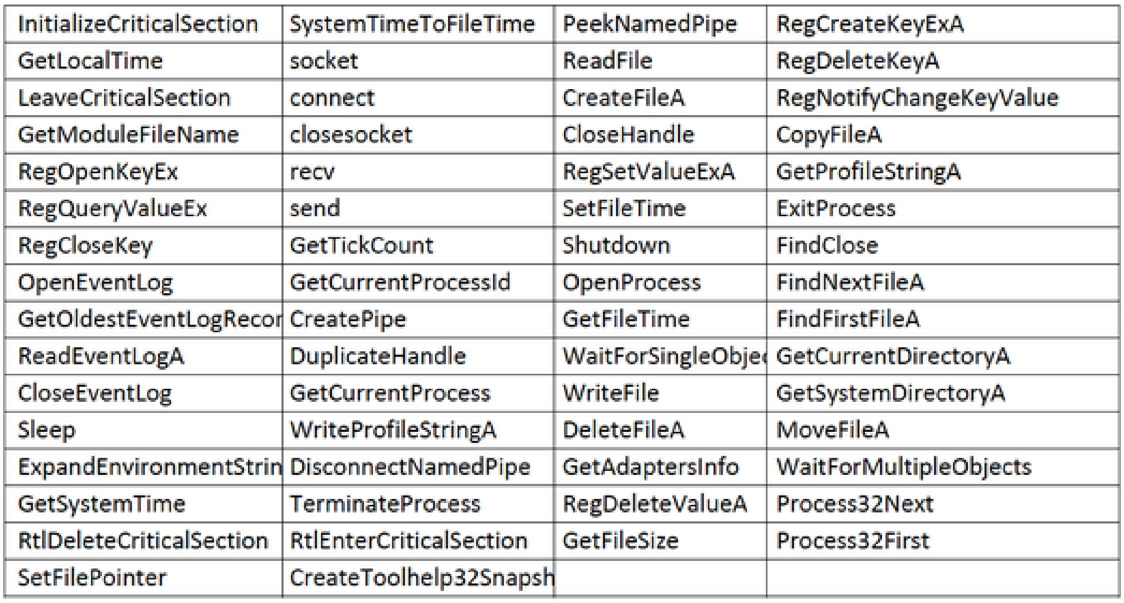
```
for idx in global_decrypted_stringz:
print idx
--End IDA script--
```

String decrypted by the IDA script are displayed below:

```
--Begin decrypted strings--
kernel32.dll
InitializeCriticalSection
GetLocalTime
LeaveCriticalSection
GetModuleFileNameA
Sleep
ExpandEnvironmentStringsA
GetSystemTime
SystemTimeToFileTime
GetTickCount
CreatePipe
DuplicateHandle
GetCurrentProcess
DisconnectNamedPipe
TerminateProcess
PeekNamedPipe
ReadFile
CreateFileA
SetFileTime
OpenProcess
GetFileTime
WaitForSingleObject
WriteFile
DeleteFileA
GetCurrentProcessId
GetAdaptersInfo
advapi32.dll
RegOpenKeyExA
RegQueryValueExA
RegCloseKey
OpenEventLogA
ReadEventLogA
CloseEventLog
RegDeleteValueA
RegCreateKeyExA
RegNotifyChangeKeyValue
Can't open update file.
File too small.
SOFTWARE\Microsoft\Windows NT\CurrentVersion
RValue
SOFTWARE\Microsoft\Windows NT\CurrentVersion
```

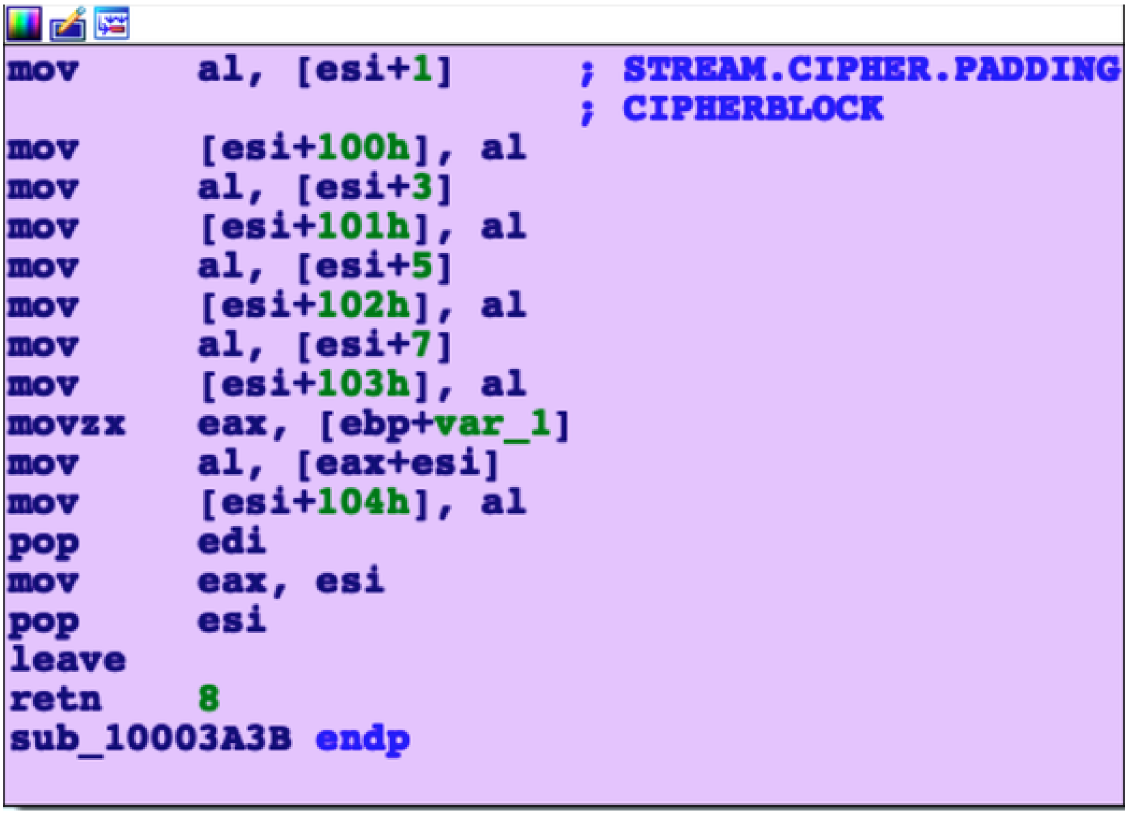
```
RValue
%temp%\~lpz.zp
Can't find plug file
Can't find plug file
Can't load more plug
Load Dll Plug Failed
%s\uaq*.dll
\services.exe
Create File Failed
Create File Failed
rundll32.exe
SOFTWARE\Microsoft\Windows NT\CurrentVersion
RValue
RValue
%SystemRoot%\system32\cmd.exe
source
Micros
CmdPage
InfoPage
cmd.exe
source
Micros
avp.exe
shell process Terminated
ReadShellThread closed
Create result file failed
Create result file failed
CreateProcess Error: %d
CreateProcess Error: %d
CreateProcess succ
Open file Failed
File Size is 0
Open file Failed
Create File Failed
Create File Failed
no shell
\services.exe
200
F::
200 OK
--End decrypted strings--
```

Screenshots



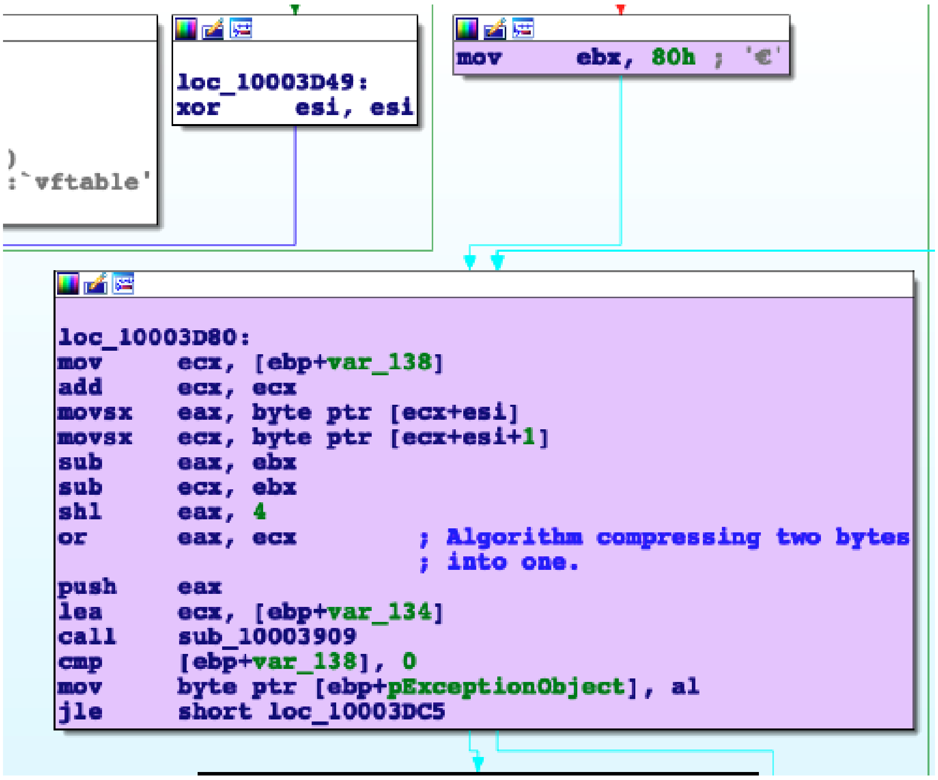
InitializeCriticalSection	SystemTimeToFileTime	PeekNamedPipe	RegCreateKeyExA
GetLocalTime	socket	ReadFile	RegDeleteKeyA
LeaveCriticalSection	connect	CreateFileA	RegNotifyChangeKeyValue
GetModuleFileName	closesocket	CloseHandle	CopyFileA
RegOpenKeyEx	recv	RegSetValueExA	GetProfileStringA
RegQueryValueEx	send	SetFileTime	ExitProcess
RegCloseKey	GetTickCount	Shutdown	FindClose
OpenEventLog	GetCurrentProcessId	OpenProcess	FindNextFileA
GetOldestEventLogRecor	CreatePipe	GetFileTime	FindFirstFileA
ReadEventLogA	DuplicateHandle	WaitForSingleObje	GetCurrentDirectoryA
CloseEventLog	GetCurrentProcess	WriteFile	GetSystemDirectoryA
Sleep	WriteProfileStringA	DeleteFileA	MoveFileA
ExpandEnvironmentStrin	DisconnectNamedPipe	GetAdaptersInfo	WaitForMultipleObjects
GetSystemTime	TerminateProcess	RegDeleteValueA	Process32Next
RtlDeleteCriticalSection	RtlEnterCriticalSection	GetFileSize	Process32First
SetFilePointer	CreateToolhelp32Snapsh		

Figure 1 - Screenshot of the following strings that are used as imports.



```
mov     al, [esi+1]         ; STREAM.CIPHER.PADDING
                            ; CIPHERBLOCK
mov     [esi+100h], al
mov     al, [esi+3]
mov     [esi+101h], al
mov     al, [esi+5]
mov     [esi+102h], al
mov     al, [esi+7]
mov     [esi+103h], al
movzx   eax, [ebp+var_1]
mov     al, [eax+esi]
mov     [esi+104h], al
pop     edi
mov     eax, esi
pop     esi
leave
retn    8
sub_10003A3B endp
```

Figure 2 - Screenshot of the complex stream cipher padding the initial cipher value.



```
)
:`vftable'
```

```
loc_10003D49:
xor     esi, esi
```

```
mov     ebx, 80h ; '€'
```

```
loc_10003D80:
mov     ecx, [ebp+var_138]
add     ecx, ecx
movsx   eax, byte ptr [ecx+esi]
movsx   ecx, byte ptr [ecx+esi+1]
sub     eax, ebx
sub     ecx, ebx
shl     eax, 4
or      eax, ecx          ; Algorithm compressing two bytes
                          ; into one.
push    eax
lea     ecx, [ebp+var_134]
call    sub_10003909
cmp     [ebp+var_138], 0
mov     byte ptr [ebp+pExceptionObject], al
jle     short loc_10003DC5
```

Figure 3 - Screenshot of the complex steam cipher compressing 2-bytes into 1-byte.

cnaweb.mrslove.com

Tags

command-and-control

Ports

443 TCP

Whois

Queried whois.publicdomainregistry.com with "mrslove.com"...

Domain Name: MRSLOVE.COM
Registry Domain ID: 70192241_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.publicdomainregistry.com
Registrar URL: www.publicdomainregistry.com
Updated Date: 2020-02-26T08:01:27Z
Creation Date: 2001-05-02T02:10:12Z
Registrar Registration Expiration Date: 2021-05-02T02:10:12Z
Registrar: PDR Ltd. d/b/a PublicDomainRegistry.com
Registrar IANA ID: 303
Domain Status: OK https://icann.org/epp#OK
Registry Registrant ID: Not Available From Registry
Registrant Name: changeip operations
Registrant Organization: changeip.com
Registrant Street: 1200 brickell ave
Registrant City: miami

```
Registrant State/Province: florida
Registrant Postal Code: 33131
Registrant Country: US
Registrant Phone: +1.800791337
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email: noc@changeip.com
Registry Admin ID: Not Available From Registry
Admin Name: changeip operations
Admin Organization: changeip.com
Admin Street: 1200 brickell ave
Admin City: miami
Admin State/Province: florida
Admin Postal Code: 33131
Admin Country: US
Admin Phone: +1.800791337
Admin Phone Ext:
Admin Fax:
Admin Fax Ext:
Admin Email: noc@changeip.com
Registry Tech ID: Not Available From Registry
Tech Name: changeip operations
Tech Organization: changeip.com
Tech Street: 1200 brickell ave
Tech City: miami
Tech State/Province: florida
Tech Postal Code: 33131
Tech Country: US
Tech Phone: +1.800791337
Tech Phone Ext:
Tech Fax:
Tech Fax Ext:
Tech Email: noc@changeip.com
Name Server: ns1.changeip.com
Name Server: ns2.changeip.com
Name Server: ns3.changeip.com
Name Server: ns4.changeip.com
Name Server: ns5.changeip.com
DNSSEC: Unsigned
Registrar Abuse Contact Email: abuse-contact@publicdomainregistry.com
Registrar Abuse Contact Phone: +1.2013775952
URL of the ICANN WHOIS Data Problem Reporting System: http://wdprs.internic.net/
```

Relationships

cnawe-b.mrslove.com	Connected_From	363ea096a3f6d06d56dc97ff1618607d462f366139df70c88310bbf77b9f9f90

Description

svchost.dll (8cf683b7d181591b91e145985f32664c) attempts to connect to the following domain.

210.68.69.82

Tags

command-and-control

Ports

443 TCP

Whois

```
Queried whois.apnic.net with "210.68.69.82"...

% Information related to '210.68.0.0 - 210.68.255.255'

% Abuse contact for '210.68.0.0 - 210.68.255.255' is 'hostmaster@twnic.net.tw'

inetnum:       210.68.0.0 - 210.68.255.255
netname:       SEEDNET
descr:         Digital United Inc.
descr:         9F, No. 125, Song Jiang Road
descr:         Taipei, Taiwan
country:       TW
admin-c:       JC256-AP
tech-c:        JC256-AP
mnt-by:        MAINT-TW-TWNIC
mnt-irt:       IRT-TWNIC-AP
status:        ALLOCATED PORTABLE
last-modified: 2018-12-12T06:04:02Z
source:        APNIC

irt:           IRT-TWNIC-AP
address:       Taipei, Taiwan, 100
e-mail:        hostmaster@twnic.net.tw
abuse-mailbox: hostmaster@twnic.net.tw
admin-c:       TWA2-AP
tech-c:        TWA2-AP
auth:          # Filtered
remarks:       Please note that TWNIC is not an ISP and is not empowered
remarks:       to investigate complaints of network abuse.
mnt-by:        MAINT-TW-TWNIC
last-modified: 2015-10-08T07:58:24Z
source:        APNIC

person:        Jonas Chou
nic-hdl:       JC256-AP
e-mail:        Jonaschou@fareastone.com.tw
address:       2F, No.218, Rueiguang Road
address:       Taipei, 114, R.O.C
```

```
phone:        +886-2-7700-8888
fax-no:       +886-2-7700-8888
country:       TW
mnt-by:        MAINT-TW-TWNIC
last-modified: 2012-12-18T10:10:01Z
source:       APNIC

% Information related to '210.68.69.80 - 210.68.69.87'

inetnum:       210.68.69.80 - 210.68.69.87
netname:       42888423-TW
descr:       Taipei Taiwan
country:       TW
admin-c:       NN3251-TW
tech-c:       NN3251-TW
mnt-by:        MAINT-TW-TWNIC
remarks:       This information has been partially mirrored by APNIC from
remarks:       TWNIC. To obtain more specific information, please use the
remarks:       TWNIC whois server at whois.twnic.net.
changed:       DavidLin1@fareastone.com.tw 20180330
status:       ASSIGNED NON-PORTABLE
source:       TWNIC

person:       NULL
address:       N/A Taiwan
country:       TW
e-mail:       joy25488@gmail.com
nic-hdl:       NN3251-TW
changed:       hostmaster@twnic.net.tw 20180331
source:       TWNIC

% This query was served by the APNIC Whois Service version 1.88.15-SNAPSHOT (WHOIS-US4)
```

Relationships

210.68.69.82	Connected_From	363ea096a3f6d06d56dc97ff1618607d462f366139d-f70c88310bbf77b9f9f90

Description

svchost.dll (8cf683b7d181591b91e145985f32664c) attempts to connect to the following IP address.

6e6d3a831c03b09d9e4a54859329fbfd428083f8f5bc5f27abbfdd9c47ec0e57

Tags

loadertrojan

Details

Name	rasautoex.dll

Size	50176 bytes
Type	PE32+ executable (DLL) (GUI) x86-64, for MS Windows
MD5	4ec8e16d426a4aaa57c454c58f447c1e
SHA1	5c89629e5873072a9ca3956b67cf7b5080312c80
SHA256	6e6d3a831c03b09d9e4a54859329fbfd428083f8f5bc5f27abbfdd9c47ec0e57
SHA512	284e0dff33f4ffb6d55f2fdb1de81d5644fb2671aa358dfb72b34a50632f708b7b071202202efec0b48bc0f622c6947f8ccf0818ebaff7277eda854cee67eeaa
ssdeep	768:DN5oCkAI3effi5djegTXLzAI78S3ge0eYUi3EaQkDdXptOKeosAmMotwEX1:DN5oCk1eyTXn+qXUi3pptJMwE
Entropy	5.681253

Antivirus

Ahnlab	Trojan/Win64.Loader
Avira	TR/Agent.ojanf
BitDefender	Trojan.GenericKD.34284956
ClamAV	Win.Packer.Taidoor-9209869-0
Comodo	Malware
Cyren	W64/Kryptik.AVM
ESET	a variant of Win64/Agent.ACK trojan
Emsisoft	Trojan.GenericKD.34284956 (B)
Ikarus	Trojan.Win64.Agent
K7	Trojan (0056be3d1)
Lavasoft	Trojan.GenericKD.34284956
McAfee	RDN/Generic trojan.ks
Microsoft Security Essentials	Trojan:Win32/Taidoor.DA!MTB
NANOAV	Trojan.Win64.Mlw.hqmqtg
Quick Heal	Trojan.Taidoor.S15351536
Sophos	Mal/Taidoor-A
Symantec	Trojan Horse
TACHYON	Trojan/W64.Dllhijacker.50176
TrendMicro	Trojan.161033AF

TrendMicro House Call	Trojan.161033AF
VirusBlokAda	Trojan.Win64.Dllhijacker
Zillya!	Trojan.Agent.Win64.5841

YARA Rules

No matches found.

ssdeep Matches

No matches found.

PE Metadata

Compile Date	2019-01-04 02:11:55-05:00
Import Hash	956b48719c7be61f48572c8fa464e00c

PE Sections

MD5	Name	Raw Size	Entropy
a9b389fc8171131551c6570d2395de57	header	1024	2.619293
8dabe7bfc2ee6b9819f554b2694c98eb	.text	26624	6.217867
8e63e6b885c3d270ccfb7607b9662601	.rdata	14848	4.618383
d44f2a519c2649244a8c87581872b483	.data	4096	2.280898
0aa4114597794059e1d4a2c246c7d7a5	.pdata	2048	4.331432
7197f896bddfd6e434b1d5703bf0c5a2	.rsrc	512	5.097979
54bb45b94c64d3717b1be8194fb4a6a7	.reloc	1024	3.689756

Description

This file is a 64-bit Windows DLL file. The file "rasautoex.dll" is a Taidoor loader and will decrypt and execute the 64-bit version of Taidoor “svchost.dll" (6627918d989bd7d15ef0724362b67edd) in memory.

0d0ccfe7cd476e2e2498b854cef2e6f959df817e52924b3a8bcdae7a8faaa686

Tags

remote-access-trojantrojan

Details

Name	svchost.dll
Size	183808 bytes
Type	data

MD5	6627918d989bd7d15ef0724362b67edd
SHA1	21e29034538bb4e3bc922149ef4312b90b6b4ea3
SHA256	0d0ccfe7cd476e2e2498b854cef2e6f959df817e52924b3a8bcdae7a8faaa686
SHA512	83ee751b15d8fd8477b8ecf8d33a4faf30b75aceb90c0e58ebf9dbbfc1d354f7e772f126b8462fd5897a4015a6f5e324d34900ff7319e8cc791fb239ca603ddc
ssdeep	3072:7PR4kaQOrd41zdruwiAyr/Ta1XxKH3zVrWvcfWsImOLdXFKY8SIMjUPpF5:3aQLgwiAyr/TiXxMsvcrxbnjUPP5
Entropy	7.999011

Antivirus

Ahnlab	Data/BIN.EncPe
Antiy	Trojan/Win32.Taidoor
Avira	TR/Taidoor.AO
BitDefender	Trojan.Agent.EUMT
ClamAV	Win.Malware.Agent-9376986-0
Cyren	W32/Taidoor.A.enc!Camelot
Emsisoft	Trojan.Agent.EUMT (B)
Ikarus	Trojan.Win32.Taidoor
Lavasoft	Trojan.Agent.EUMT
McAfee	Trojan-Taidoor
Microsoft Security Essentials	Trojan:Win32/Taidoor.DB!MTB
Sophos	Troj/Taidoor-A
Symantec	Trojan Horse
TrendMicro	Backdoo.4FA5823A
TrendMicro House Call	Backdoo.4FA5823A

YARA Rules

```
rule CISA_10292089_01 : rat loader TAIDOOR
{
   meta:
       Author = "CISA Code & Media Analysis"
       Incident = "10292089"
       Date = "2020-06-18"
       Last_Modified = "20200616_1530"
       Actor = "n/a"
       Category = "Trojan Loader Rat"
       Family = "TAIDOOR"
       Description = "Detects Taidoor Rat Loader samples"
       MD5_1 = "8cf683b7d181591b91e145985f32664c"
       SHA256_1 =
"363ea096a3f6d06d56dc97ff1618607d462f366139df70c88310bbf77b9f9f90"
       MD5_2 = "6627918d989bd7d15ef0724362b67edd"
       SHA256_2 =
"0d0ccfe7cd476e2e2498b854cef2e6f959df817e52924b3a8bcdae7a8faaa686"
   strings:
       $s0 = { 8A 46 01 88 86 00 01 00 00 8A 46 03 88 86 01 01 00 00 8A 46 05 88 86 02 01
00 00 8A 46 07 88 86 03 01 00 00 }
       $s1 = { 88 04 30 40 3D 00 01 00 00 7C F5 }
       $s2 = { 0F BE 04 31 0F BE 4C 31 01 2B C3 2B CB C1 E0 04 0B C1 }
       $s3 = { 8A 43 01 48 8B 6C 24 60 88 83 00 01 00 00 8A 43 03 }
       $s4 = { 88 83 01 01 00 00 8A 43 05 88 83 02 01 00 00 8A 43 07 88 83 03 01 00 00 }
       $s5 = { 41 0F BE 14 7C 83 C2 80 41 0F BE 44 7C 01 83 C0 80 C1 E2 04 0B D0 }
       $s6 = { 5A 05 B2 CB E7 45 9D C2 1D 60 F0 4C 04 01 43 85 3B F9 8B 7E }
   condition:
       ($s0 and $s1 and $s2) or ($s3 and $s4 and $s5) or ($s6)
}
```

ssdeep Matches

No matches found.

Relationships

0d0ccfe7cd...	Connected_To	infonew.dubya.net

Description

This encrypted file has been identified as the Taidoor RAT loaded by “rasautoex.dll" (4ec8e16d426a4aaa57c454c58f447c1e). This file contains the same functionality and encryption keys as the 32-bit version “svchost.dll” (8CF683B7D181591B91E145985F32664C).

This file calls out to a different C2. This C2 was also observed in memory of the infected system provided for analysis.

--Begin C2--
infonew.dubya.net
--End C2--

The malware author never removed the symbol file for “rasautoex.dll” as with the 32-bit version. However, this artifact provides some additional information that the malware author intended this binary to do, “MemLoad(pass symantec)”.

```
--Begin symbol file artifact--
C:\Users\user\Desktop\MemLoad(pass symantec)\version\x64\Release\MemoryLoad.pdb
--End symbol file artifact--
```

infonew.dubya.net

Tags

command-and-control

Whois

Queried whois.publicdomainregistry.com with "dubya.net"...

```
Domain Name: DUBYA.NET
Registry Domain ID: 1861808123_DOMAIN_NET-VRSN
Registrar WHOIS Server: whois.publicdomainregistry.com
Registrar URL: www.publicdomainregistry.com
Updated Date: 2020-04-02T07:01:52Z
Creation Date: 2014-06-06T17:44:43Z
Registrar Registration Expiration Date: 2021-06-06T17:44:43Z
Registrar: PDR Ltd. d/b/a PublicDomainRegistry.com
Registrar IANA ID: 303
Domain Status: OK https://icann.org/epp#OK
Registry Registrant ID: Not Available From Registry
Registrant Name: changeip operations
Registrant Organization: changeip.com
Registrant Street: 1200 brickell ave
Registrant City: miami
Registrant State/Province: florida
Registrant Postal Code: 33131
Registrant Country: US
Registrant Phone: +1.800791337
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email: noc@changeip.com
Registry Admin ID: Not Available From Registry
Admin Name: changeip operations
Admin Organization: changeip.com
Admin Street: 1200 brickell ave
Admin City: miami
Admin State/Province: florida
Admin Postal Code: 33131
Admin Country: US
Admin Phone: +1.800791337
Admin Phone Ext:
```

Admin Fax:
Admin Fax Ext:
Admin Email: noc@changeip.com
Registry Tech ID: Not Available From Registry
Tech Name: changeip operations
Tech Organization: changeip.com
Tech Street: 1200 brickell ave
Tech City: miami
Tech State/Province: florida
Tech Postal Code: 33131
Tech Country: US
Tech Phone: +1.800791337
Tech Phone Ext:
Tech Fax:
Tech Fax Ext:
Tech Email: noc@changeip.com
Name Server: ns1.changeip.com
Name Server: ns2.changeip.com
Name Server: ns3.changeip.com
Name Server: ns4.changeip.com
Name Server: ns5.changeip.com
DNSSEC: Unsigned
Registrar Abuse Contact Email: abuse-contact@publicdomainregistry.com
Registrar Abuse Contact Phone: +1.2013775952
URL of the ICANN WHOIS Data Problem Reporting System: http://wdprs.internic.net/

Relationships

infonew.dubya.net	Connected_From	0d0ccfe7cd476e2e2498b854cef2e6f959df817e52924b3a8bcdae7a8faaa686

Description

svchost.dll (6627918d989bd7d15ef0724362b67edd) attempts to connect to the following domain.

Relationship Summary

4a0688baf9...	Used	363ea096a3f6d06d56dc97ff1618607d462f366139df70c88310bbf77b9f9f90
363ea096a3...	Used_By	4a0688baf9661d3737ee82f8992a0a665732c91704f28688f643115648c107d4
363ea096a3...	Connected_To	cnaweb.mrslove.com
363ea096a3...	Connected_To	210.68.69.82
cnaweb.mrslove.com	Connected_From	363ea096a3f6d06d56dc97ff1618607d462f366139df70c88310bbf77b9f9f90

210.68.69.82	Connected_From	363ea096a3f6d06d56dc97ff1618607d462f366139df70c88310bbf77b9f9f90
0d0ccfe7cd...	Connected_To	infonew.dubya.net
infonew.dubya.net	Connected_From	0d0ccfe7cd476e2e2498b854cef2e6f959df817e52924b3a8bcdae7a8faaa686

Mitigation

```
alert tcp 210.68.69.82 any <> $HOME_NET any (msg:" Malicious traffic "; sid:#########; rev:1; classtype:tcp-event;)

alert tcp 156.238.3.162 any <> $HOME_NET any (msg:"Malicious traffic"; sid:#########; rev:1; classtype:tcp-event;)

alert udp any any 53 <> $HOME_NET any (msg:"Attempt to connect to malicious domain"; content:"|03|www|07|infonew|05|dubya|03|net|00|"; sid:#########; rev:1;)

alert udp any any 53 <> $HOME_NET any (msg:"Attempt to connect to malicious domain"; content:"|03|www|06|cnaweb|07|mrslove|03|com|00|"; sid:#########; rev:1;)
```

Note: At the time of analysis, one of the domains resolved to the IP address 156.238.3.162.

Recommendations

CISA recommends that users and administrators consider using the following best practices to strengthen the security posture of their organization's systems. Any configuration changes should be reviewed by system owners and administrators prior to implementation to avoid unwanted impacts.

- Maintain up-to-date antivirus signatures and engines.
- Keep operating system patches up-to-date.
- Disable File and Printer sharing services. If these services are required, use strong passwords or Active Directory authentication.
- Restrict users' ability (permissions) to install and run unwanted software applications. Do not add users to the local administrators group unless required.
- Enforce a strong password policy and implement regular password changes.
- Exercise caution when opening e-mail attachments even if the attachment is expected and the sender appears to be known.
- Enable a personal firewall on agency workstations, configured to deny unsolicited connection requests.
- Disable unnecessary services on agency workstations and servers.
- Scan for and remove suspicious e-mail attachments; ensure the scanned attachment is its "true file type" (i.e., the extension matches the file header).
- Monitor users' web browsing habits; restrict access to sites with unfavorable content.
- Exercise caution when using removable media (e.g., USB thumb drives, external drives, CDs, etc.).
- Scan all software downloaded from the Internet prior to executing.

- Maintain situational awareness of the latest threats and implement appropriate Access Control Lists (ACLs).

Additional information on malware incident prevention and handling can be found in National Institute of Standards and Technology (NIST) Special Publication 800-83, **"Guide to Malware Incident Prevention & Handling for Desktops and Laptops".**

Contact Information

CISA continuously strives to improve its products and services. You can help by answering a very short series of questions about this product at the following URL: https://www.cisa.gov/forms/feedback/

Document FAQ

What is a MIFR? A Malware Initial Findings Report (MIFR) is intended to provide organizations with malware analysis in a timely manner. In most instances this report will provide initial indicators for computer and network defense. To request additional analysis, please contact CISA and provide information regarding the level of desired analysis.

What is a MAR? A Malware Analysis Report (MAR) is intended to provide organizations with more detailed malware analysis acquired via manual reverse engineering. To request additional analysis, please contact CISA and provide information regarding the level of desired analysis.

Can I edit this document? This document is not to be edited in any way by recipients. All comments or questions related to this document should be directed to the CISA at 1-888-282-0870 or CISA Service Desk ✉ .

Can I submit malware to CISA? Malware samples can be submitted via three methods:

- Web: https://malware.us-cert.gov
- E-Mail: submit@malware.us-cert.gov ✉
- FTP: ftp.malware.us-cert.gov (anonymous)

CISA encourages you to report any suspicious activity, including cybersecurity incidents, possible malicious code, software vulnerabilities, and phishing-related scams. Reporting forms can be found on CISA's homepage at www.cisa.gov.

Revisions

August 3, 2020: Initial Version

August 3, 2020: Corrected Snort rules

August 31, 2020: Updated

This product is provided subject to this Notification and this Privacy & Use policy.

Was this document helpful? Yes | Somewhat | No