APT10: sophisticated multi-layered loader Ecipekac discovered in A41APT campaign

By GReAT

Published: 2021-03-30 · Archived: 2026-04-06 00:56:36 UTC

Why is the campaign called A41APT?

In 2019, we observed an APT campaign targeting multiple industries, including the Japanese manufacturing industry and its overseas operations, that was designed to steal information. We named the campaign A41APT (not APT41) which is derived from the host name “DESKTOP-A41UVJV” from the attacker’s system used in the initial infection. The actor leveraged vulnerabilities in Pulse Connect Secure in order to hijack VPN sessions, or took advantage of system credentials that were stolen in previous operations.

```
2019-10-    15:30:28 - VPN Tunneling: Session started for user with IPv4 address 192.168.X.X, hostname ホスト名
2019-10-    15:30:28 - VPN Tunneling: User with IP 192.168.X.X connected with SSL transport mode.
2019-10-    15:30:28 - Closed connection to TUN-VPN port 443 after 6 seconds, with 0 bytes read (in 1 chunks) and 221 bytes written (in 6 chunks)
2019-10-    15:30:28 - VPN Tunneling: User with IP 192.168.X.X connected with ESP transport mode.
2019-10-    15:30:28 - Key Exchange number 1 occurred for user with NCIP 192.168.X.X
2019-10-    15:30:28 - VPN Tunneling: Session ended for user with IPv4 address 192.168.X.X
2019-10-    15:30:28 - Closed connection to 192.168.X.X after 0 seconds, with 0 bytes read and 0 bytes written
2019-10-    15:30:28 - VPN Tunneling: Session started for user with IPv4 address 192.168.X.X, hostname DESKTOP-A41UVJV
2019-10-    15:30:28 - Connected to TUN-VPN port 443
2019-10-    15:30:28 - Key Exchange number 1 occurred for user with NCIP 192.168.X.X
2019-10-    15:30:29 - Remote address for user <ドメイン/ユーザ名> changed from ユーザのリモートIPアドレス to 151.80.241.108.
```

Log of the hijacking VPN session from DESKTOP-A41UVJV

A41APT is a long-running campaign with activities detected from March 2019 to the end of December 2020. Most of the discovered malware families are fileless malware and they have not been seen before. One particular piece of malware from this campaign is called Ecipekac (a.k.a DESLoader, SigLoader, and HEAVYHAND). It is a very sophisticated multi-layer loader module used to deliver payloads such as SodaMaster (a.k.a DelfsCake, dfls, and DARKTOWN), P8RAT (a.k.a GreetCake, and HEAVYPOT) and FYAnti (a.k.a DILLJUICE stage2) which loads QuasarRAT.

In November and December 2020, Symantec and LAC both published blogposts about this campaign. A month later, we discovered new activities from A41APT that utilized modified and updated payloads, and that’s what we cover in this blog.

In February 2021, a GReAT security expert and his friends gave a presentation on the A41APT campaign at the GReAT Ideas event. You can download the slides here. Further information about A41APT is available to customers of the Kaspersky Intelligence Reporting service. Contact intelreports@kaspersky.com

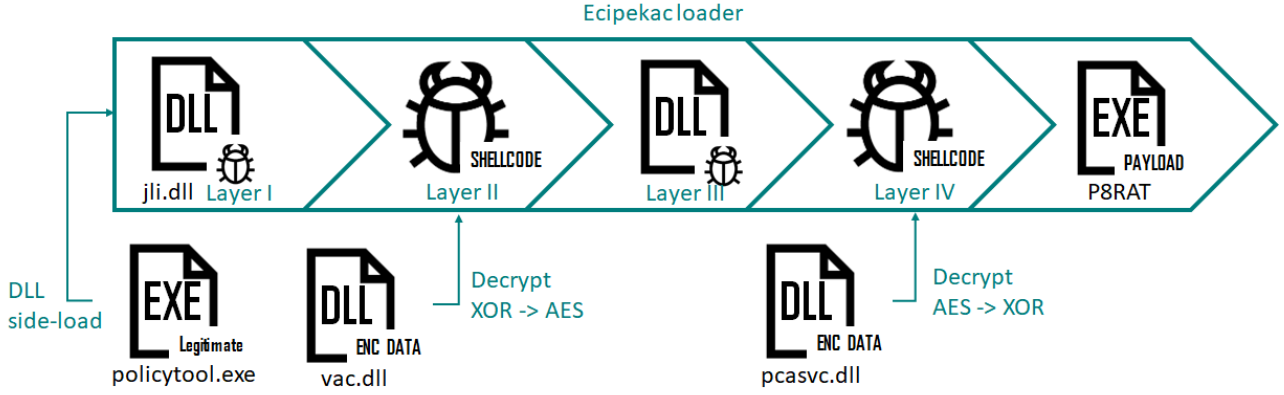
Technical analysis: Ecipekac

We observed a multi-layer x64 loader used exclusively by this actor and dubbed Ecipekac after a unique string found in the second layer of the Ecipekac loader. The string is “Cake piece” in reverse (with a typo).

```
xor     edi, edi
lea     r15, aEcipekac  ; "ecipekac"
mov     edx, edi
mov     ecx, edi

                        ; CODE XREF: sub_11B0A20+6C↓j
cmp     byte ptr [rcx+r15], 65h ; 'e'
jnz     short loc_11B0A80
cmp     byte ptr [rcx+r15+1], 63h ; 'c'
jnz     short loc_11B0A80
cmp     byte ptr [rcx+r15+2], 69h ; 'i'
jnz     short loc_11B0A80
cmp     byte ptr [rcx+r15+3], 70h ; 'p'
jnz     short loc_11B0A80
cmp     byte ptr [rcx+r15+4], 65h ; 'e'
jnz     short loc_11B0A80
cmp     byte ptr [rcx+r15+5], 6Bh ; 'k'
jnz     short loc_11B0A80
cmp     byte ptr [rcx+r15+6], 61h ; 'a'
jnz     short loc_11B0A80
cmp     byte ptr [rcx+r15+7], 63h ; 'c'
jz      short loc_11B0A90

                        ; CODE XREF: sub_11B0A20+26↑j
                        ; sub_11B0A20+2E↑j ...
inc     edx
```

The hardcoded unique string “ecipekac”

Ecipekac uses a new, complicated loading schema: it uses the four files listed below to load and decrypt four fileless loader modules one after the other to eventually load the final payload in memory.



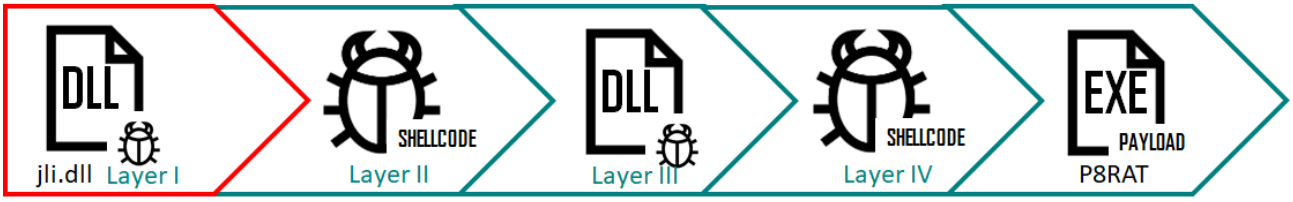
Ecipekac infection flow

The files are:

Filename	MD5 Hash	Description
policytool.exe	7e2b9e1f651fa5454d45b974d00512fb	Legitimate exe for DLL side-loading
jli.dll	be53764063bb1d054d78f2bf08fb90f3	Ecipekac Layer I loader
vac.dll	f60f7a1736840a6149d478b23611d561	Encrypted Ecipekac Layer II loader (shellcode)
pcasvc.dll	59747955a8874ff74ce415e56d8beb9c	Encrypted Ecipekac Layer IV loader (shellcode)

Please note that the Ecipekac Layer III loader module is embedded in the encrypted Layer II loader.

Ecipekac: Layer I loader



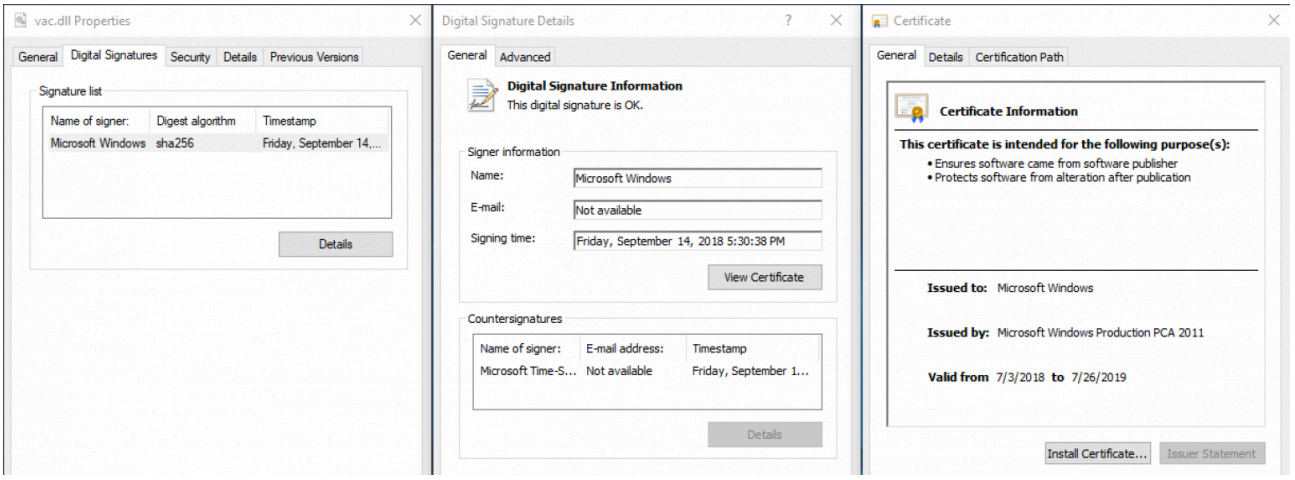
Layer I of Ecipekac infection flow

The Ecipekac Layer I loader abuses policytool.exe, a legitimate application that is normally packaged in the IBM Development Package for Eclipse, to load a malicious DLL named ‘jli.dll’ in the current directory via the DLL side-loading technique. The ‘jli.dll’ file acts as the first layer of the Ecipekac loader. This DLL file has a number of export functions; however, all of them refer to a similar function that carries the main loading feature. The loader reads 0x40408 bytes of data from the end of another DLL – ‘vac.dll’ (where the data section starts at the offset of 0x66240). The data size of 0x40408 is derived from a hardcoded value, 0x40405, incremented until it’s divisible by eight.

MD5	f60f7a1736840a6149d478b23611d561
SHA1	5eb69114b2405a6ea0780b627cd68b86954a596b
SHA256	3b8ce709fc2cee5e7c037a242ac8c84e2e00bd597711093d7c0e85ec68e14a4c
Link time	2033-11-13 08:50:03
File type	PE32+ executable (DLL) (GUI) x86-64, for MS Windows
Compiler	Linker Version: 14.13, OS Version: 10.0
File size	681544 (666KB)

File name	vac.dll
Embedded data at 0x66240 (size:0x40405)	00066240: febe d990 66de 1bc9 75b7 dc2c 3e1f 3ef2 00066250: 78d0 0005 5c27 a511 c122 bdf4 15e7 052c 00066260: af72 7e08 064c f7b9 70f0 57bf 250a 3b4d [..skipped..] 000a6630: ee4b b1f2 294d eea1 290e aba2 6954 130f 000a6640: 1267 9ab3 f800 0000

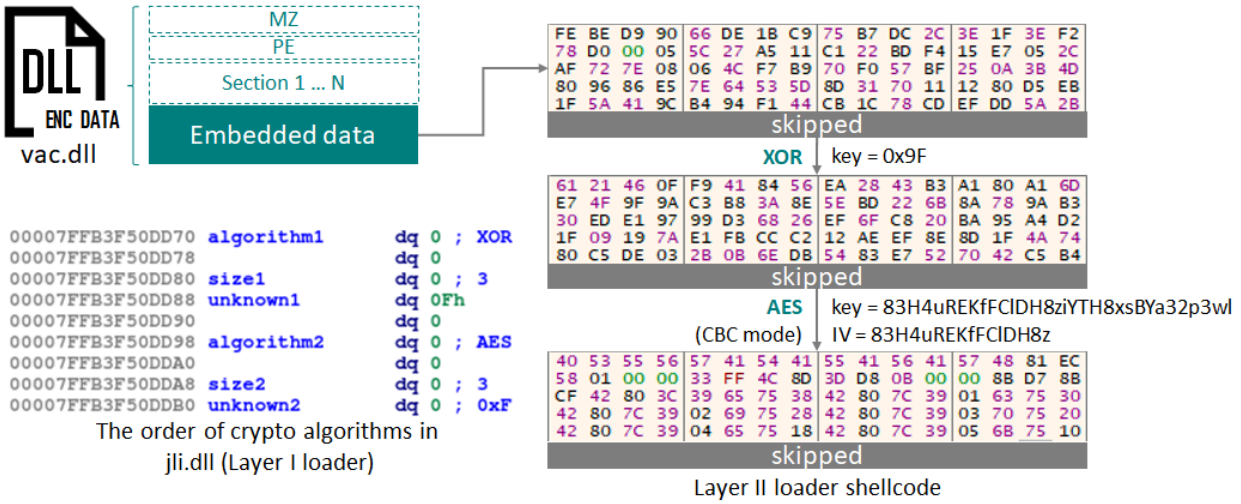
The ‘vac.dll’ DLL file is signed with a valid, legitimate digital signature, although the file has been tampered with. At first glance, the fact that its digital signature is valid would suggest the file has not been manipulated after being digitally signed.



The signed digital certificate of vac.dll

However, what happened was that the actor resized the Certificate Table in the digitally signed ‘vac.dll’ and inserted their own data in the Certificate Table so it doesn’t affect the digital signature. This technique was published at BlackHat 2016 as MS13-098.

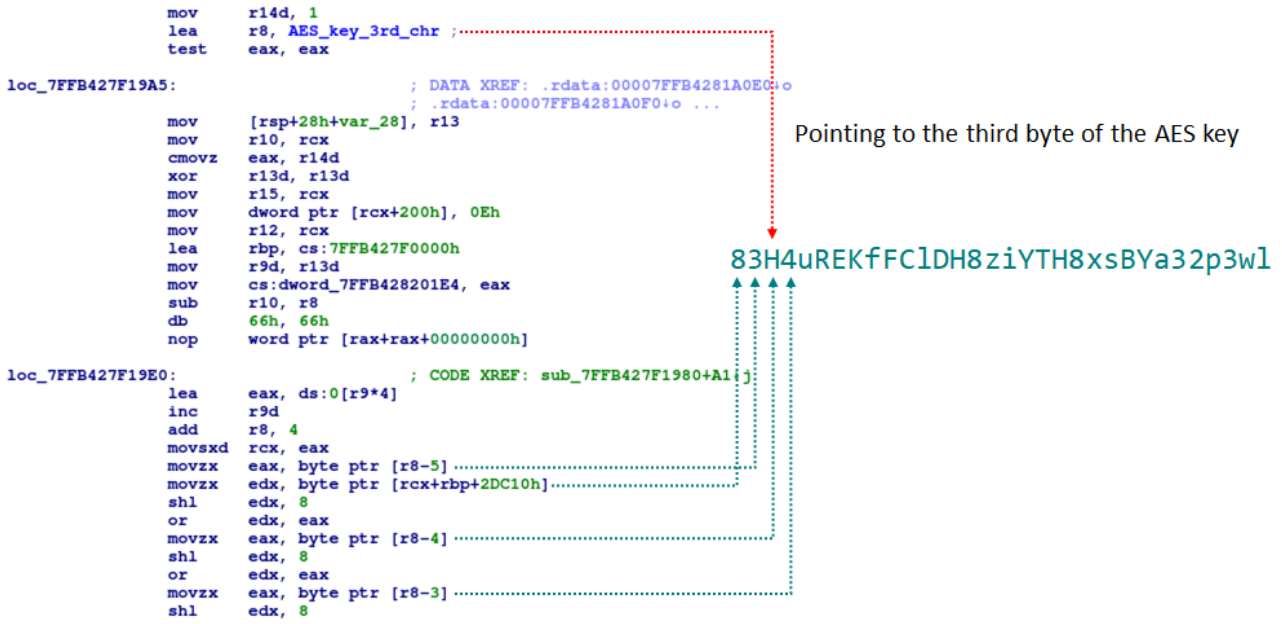
The layer I loader decrypts the layer II loader shellcode from the embedded data in ‘vac.dll’. Several crypto algorithms are used, such as XOR, AES and DES. The order and combination of algorithms, as well as the decryption keys, are different from one sample to another.



Decryption flow in first loader

For example, in the sample shown above, the order of crypto algorithms was a one-byte XOR using the hardcoded key of '0x9F', followed by an AES CBC mode decryption with the AES key '83H4uREKfFClDH8ziYTH8xsBYa32p3wl' and the IV key '83H4uREKfFClDH8z'.

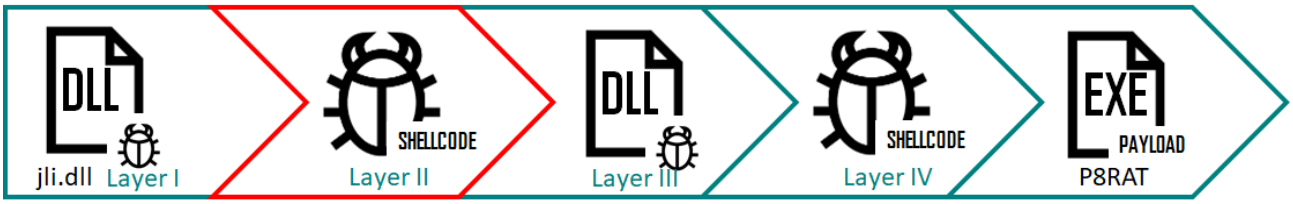
One interesting characteristic of Ecipekac is that the attackers implemented these cryptographic algorithms in their own code instead of using the Windows API. The attackers have also made slight modifications compared to the original implementation. For instance, in the function related to the AES algorithm, they intentionally referenced the third byte of the AES key as shown in the following code.



A small modification in the AES function

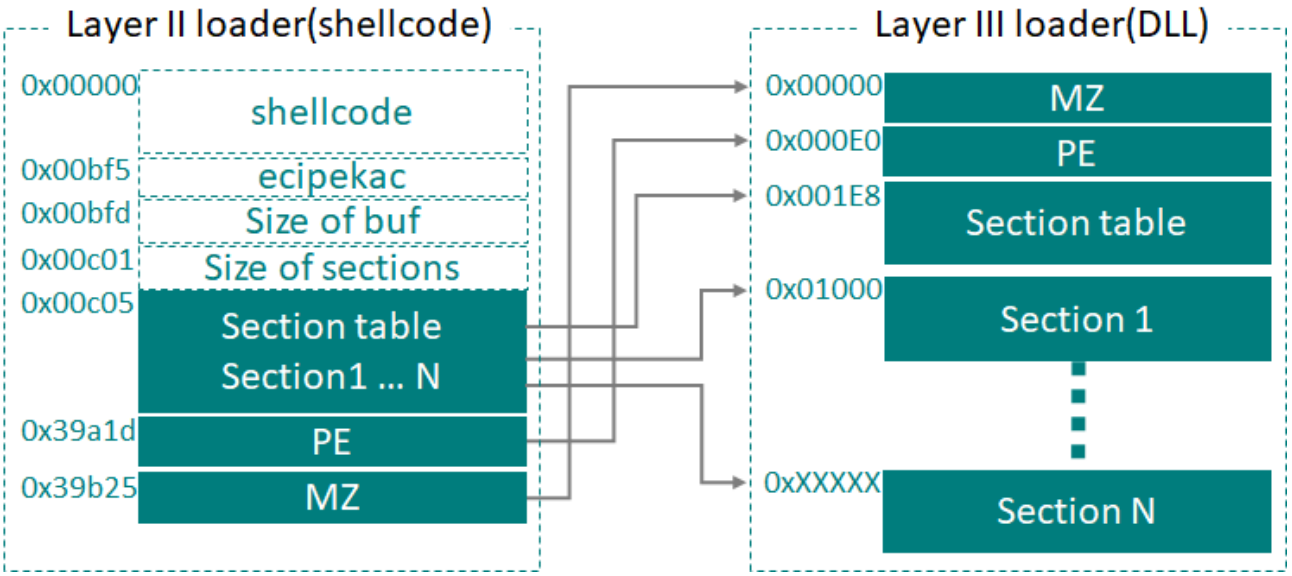
Apart from the AES algorithm mentioned, the attackers have also modified the DES algorithm.

Ecipekac: Layer II loader shellcode



Layer II of infection flow using Ecipekac

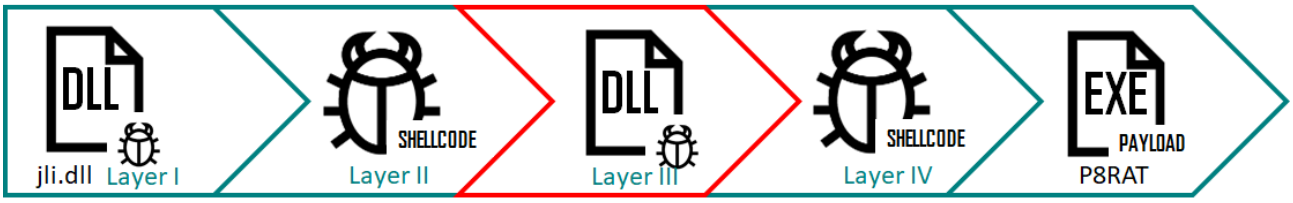
The Ecipekac Layer II loader is a simple shellcode which contains the data of the next layer DLL in disordered parts. At first, this shellcode checks for the magic string “ecipekac” in this data set. Then it reconstructs and loads each part of the embedded data into allocated memory in the correct order to create the original code of the DLL as shown below.



Reconstruction for the divided PE BLOB in memory

Then it calls the entry point of the loaded DLL which is the third layer of Ecipekac. Based on our investigation, the magic string used in this module is not exclusively “ecipekac” in all instances. We also observed “9F 8F 7F 6F” and “BF AF BF AF” being used in several samples instead.

Ecipekac Layer III loader DLL



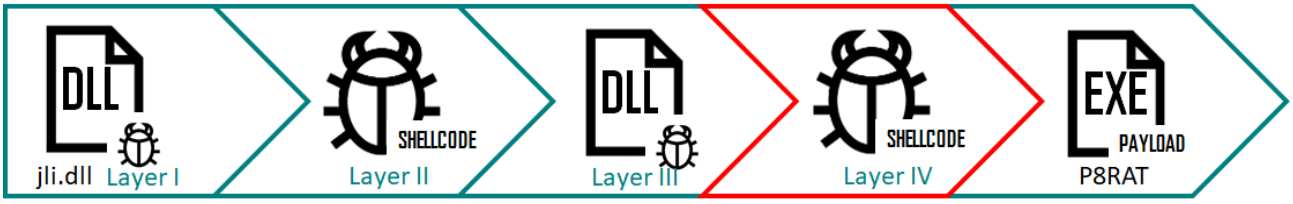
Layer III of infection flow using Ecipekac

The third layer’s method of loading the next layer resembles the first layer. It reads encrypted data from the end of ‘pcasvc.dll’, which is signed using a digital certificate as is the case with ‘vac.dll’.

MD5	59747955a8874ff74ce415e56d8beb9c
SHA1	0543bfebff937039e304146c23bbde7693a67f4e
SHA256	a04849da674bc8153348301d2ff1103a7537ed2ee55a1588350ededa43ff09f6
Link time	2017-02-24 15:47:04
File type	PE32+ executable (DLL) (console) x86-64, for MS Windows
Compiler	Linker Version: 14.13, OS Version: 10.0
File size	733232 (717KB)
File name	pcasvc.dll
Embedded data at 0x87408 (size:0x2BC28)	00087408: 98e4 1def 8519 d194 3c70 4e84 458a e34c 00087418: b145 74da c353 8cf8 1d70 d024 8a54 8bde [..skipped..] 000b3010: 2c1b 6736 8935 d55d 8090 0829 5dfc 7352 000b3020: 44bd c35b 9b23 1cb6 0000 0000 0000 0000

The crypto algorithms are again one-byte XOR and AES CBC mode, this time to decrypt the fourth loader shellcode from the embedded data of 'pcasvc.dll'. However, the sequence of algorithms is in reverse order compared to the first layer. The hardcoded keys are also different: "0x5E" is used as the XOR key, while the AES key and IV are "K4jcj02QSLWp8lK9gMK9h7W0L9iB2eEW" and "K4jcj02QSLWp8lK9" respectively.

Ecipekac: Layer IV loader shellcode



Layer IV of infection flow using Ecipekac

During our research, we found three different types of shellcode used as the fourth layer of Ecipekac.

Layer IV loader shellcode – first type

The first shellcode type's procedure acts the same way as the Ecipekac Layer II shellcode, with the only difference being the embedded PE, which is the final payload of Ecipekac in this case. The payload of the first type shellcode is either "P8RAT" or "FYAnti loader". An analysis of these payloads is provided in the later sections of this report.

Layer IV loader shellcode – second type

The second type of shellcode is totally different from the other loader types. This shellcode has a unique data structure shown in the table below.

Offset	Example Data	Description
0x000	90 90 90 90 90 90 90 90	magic number to check before proceeding to data processing.
0x008	0x11600	size of encrypted data
0x00C	A9 5B 7B 84 9C CB CF E8 B6 79 F1 9F 05 B6 2B FE	16 bytes RC4 key
0x01C	C7 36 7E 93 D3 07 1E 86 23 75 10 49 C8 AD 01 9F 6E D0 9F 06 85 97 B2 [skipped]	Encrypted payload (SodaMaster) by RC4

This shellcode confirms the presence of the magic number “90 90 90 90 90 90 90 90” at the beginning of this data structure, before proceeding to decrypt a payload at offset 0x01C using RC4 with a 16-byte key of “A9 5B 7B 84 9C CB CF E8 B6 79 F1 9F 05 B6 2B FE”. The decrypted payload is “SodaMaster”, and is described later in this report.

Layer IV loader shellcode – third type

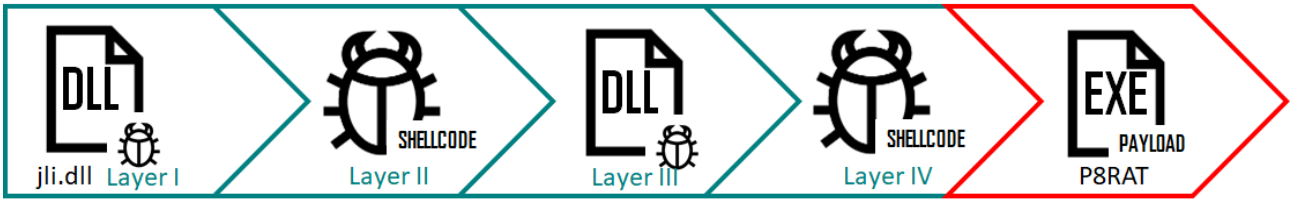
The last type of shellcode is a Cobalt Strike stager. We have confirmed the use of several different Cobalt Strike stager shellcodes since October 2019. In addition, some of the observed Cobalt Strike stager samples included a setting in the HTTP header of their malicious communications to disguise them as common jQuery request in order to evade detection by security products.

```
loc_1A4:                                ; CODE XREF: sub_10C+1D↑j
                call    sub_12B
; ---------------------------------------------------------------------------
aJquery332SlimM db '/jquery-3.3.2.slim.min.js',0
                dq 2BF74A74899DD0E3h
                dq 491D2C8E02A0A23Eh
                dq 4D4EF3B8BB658F92h
                dq 0C13F23F31D0E583Ah
                dq 0BC71CC77A00B5ED2h
                dq 8ECFD4F348461C93h
                dd 7525A8E3h
                db  9Ah
                db    0
aAcceptTextHtml db 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*'
                db ';q=0.8',0Dh,0Ah
                db 'Accept-Language: en-US,en;q=0.5',0Dh,0Ah
                db 'Host: code.jquery.com',0Dh,0Ah
                db 'Referer: http://code.jquery.com/',0Dh,0Ah
                db 'Accept-Encoding: gzip, deflate',0Dh,0Ah
                db 'User-Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) li'
                db 'ke Gecko',0Dh,0Ah,0
                dq 0E3D54C2DBD970227h, 0B00E484C62E1CF6h, 0AFD6E9777402A0Fh
```

Hardcoded HTTP header to impersonate jQuery request

The actual hardcoded C2 used in the HTTP header for the C2 communication impersonating JQuery requests was “51.89.88[.]126” with the respective port 443.

Payloads of Ecipekac

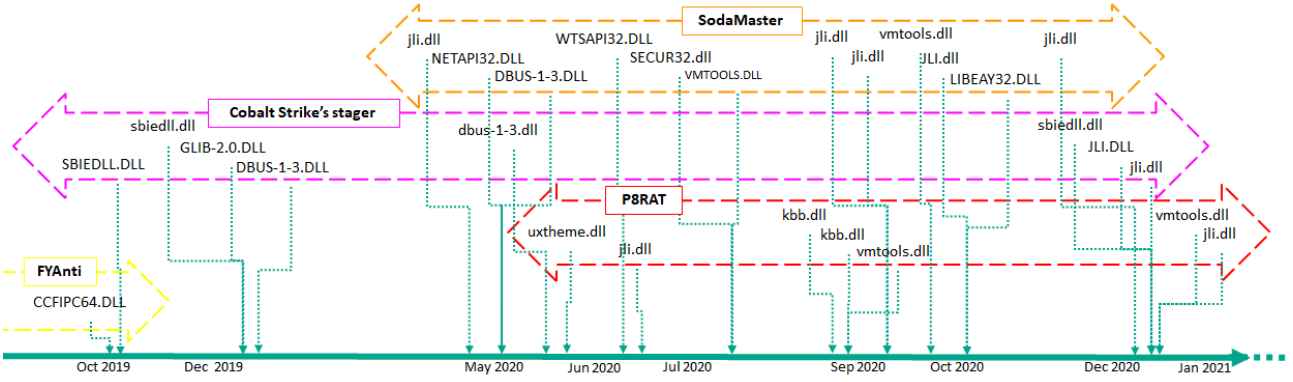


Payload of Infection flow using Ecipekac

As mentioned previously, apart from the Cobalt Strike stager, we observed three types of final payload implanted by the Ecipekac loader during this long-running campaign.

- P8RAT
- SodaMaster
- FYAnti loader for QuasarRAT

The following timeline shows samples of the Ecipekac loader together with their respective filename and payload type based on a compilation timestamp of the first layer DLL:



Timeline of the Ecipekac loader files and payloads

Cobalt Strike’s stager has been used throughout the research period. FYAntiLoader for QuasarRAT was monitored in October 2019, and has not been observed since then. Instead of this, SodaMaster and P8RAT were monitored from May 2020.

P8RAT

One of Ecipekac’s payloads is a new fileless malware which we call P8RAT (a.k.a GreetCake). P8RAT has the following unique data structure used to store the C2 communication configuration. We collected several samples of P8RAT during our research and found no C2 address of P8RAT that was used more than once. In total we found 10 backdoor commands in all the collected P8RAT samples. The most recent P8RAT sample, with the compilation

timestamp of December 14, 2020, shows a new backdoor command with the code number of “309” implemented. The command “304”, which was present in earlier samples and carries similar functionality, was removed.

Command	Description	Compilation time of P8RAT		
		2020-03-30	2020-08-26	2020-12-14
300	Closing socket	Enabled	Enabled	Enabled
301	Creating a thread for executing/loading a downloaded PE file	Enabled	Enabled	Enabled
302	No functionality	Enabled	Removed	Removed
303	Sending randomly generated data	Enabled	Enabled	Enabled
304	Executing/loading downloaded PE/shellcode	Enabled	Removed	Removed
305	Setting value of “Set Online Time” (the string was hardcoded in the P8RAT version compiled on 2020-03-30 and removed from the P8RAT version compiled on 2020-08-26).	Enabled	Enabled	Enabled
306	Setting value of “Set Reconnect TimeOut” (the string was hardcoded in the P8RAT version compiled on 2020-03-30 and removed from the P8RAT version compiled on 2020-08-26).	Enabled	Enabled	Enabled
307	Setting value of “Set Reconnect times” (the string was hardcoded in the P8RAT version compiled on 2020-03-30 and removed from the P8RAT version compiled on 2020-08-26).	Enabled	Enabled	Enabled
308	Setting value of “Set Sleep time” (the string was hardcoded in the P8RAT version compiled on 2020-03-30 and removed from the P8RAT version compiled on 2020-08-26).	Enabled	Enabled	Enabled
309	Creating thread for executing downloaded shellcode was implemented from the P8RAT version compiled on 2020-12-14.	Not implemented	Not implemented	Enabled

The main purpose of P8RAT is downloading and executing payloads (consisting of PE and shellcode) from its C2 server. However, we were unable to obtain any sample of the subsequent payloads for this malware.

In the P8RAT sample from March 2020, hardcoded strings such as “Set Online Time”, “Set Reconnect TimeOut”, “Set Reconnect Times” and “Set Sleep Time” were used in regard to backdoor commands “305” to “308”, which point to the possible purpose of these commands. Based on these strings, which were removed from the P8RAT samples in August 2020, we speculate that these commands are possibly used to control the intervals of the C2 communication by defining sleep time, reconnect time and reconnect timeout in order to blend C2 communication with normal network traffic of the system.

In another update, the P8RAT sample from August 2020 looks for two process names (“VBoxService.exe” and “vmtoolsd.exe”) on the victim’s system, to detect VMware or VirtualBox environments at the beginning of its main malicious function.

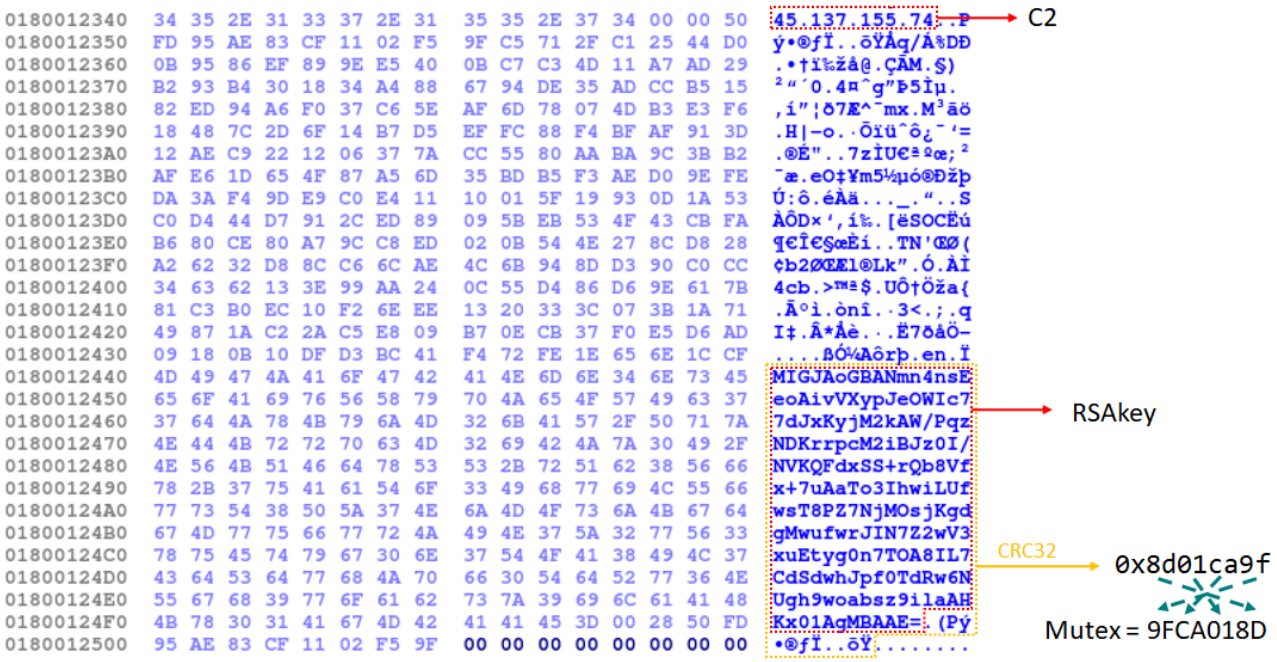
```
char detectVMenv_1121594()
{
  char v0; // bl
  HANDLE hObject; // rdi
  PROCESSENTRY32 pe; // [rsp+20h] [rbp-148h]

  pe.dwSize = 304;
  memset(&pe.cntUsage, 0, 0x12Cui64);
  v0 = 0;
  hObject = CreateToolhelp32Snapshot(2u, 0);
  Process32First(hObject, &pe);
  while ( (unsigned int)Process32Next(hObject, &pe) )
  {
    if ( !(unsigned int)lstrcmpA(pe.szExeFile, "VBoxService.exe")
      || !(unsigned int)lstrcmpA(pe.szExeFile, "vmtoolsd.exe") )
    {
      v0 = 1;
      break;
    }
  }
  if ( hObject != (HANDLE)-1i64 )
    CloseHandle(hObject);
  return v0;
}
```

Hardcoded file names to detect VMware and VirtualBox

Interestingly the attacker made some modifications to P8RAT in December 2020, shortly after the publication of the two blogposts from Symantec on November 17, 2020, and LAC on December 1, 2020 (in Japanese). We strongly believe that this actor had examined these security vendors’ publications carefully and then modified P8RAT accordingly.

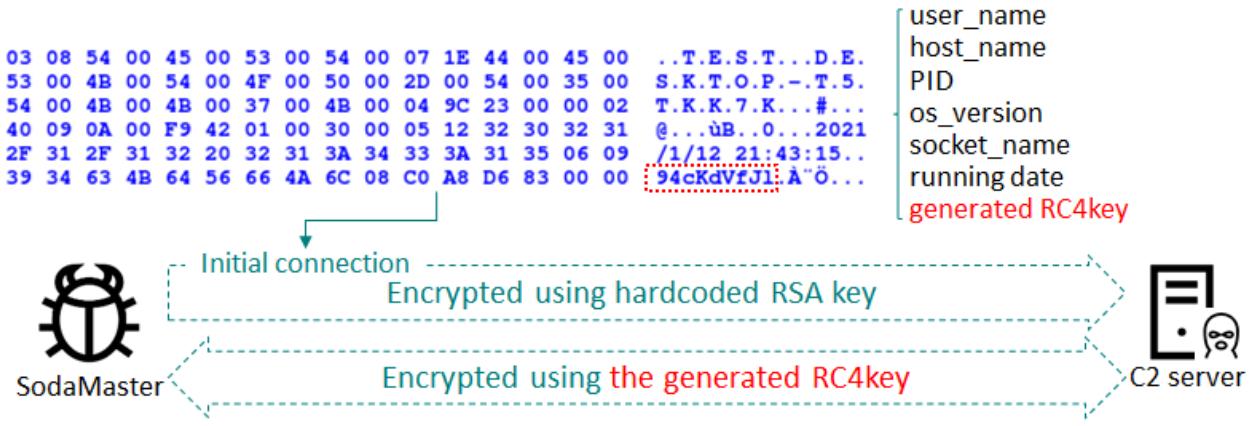
SodaMaster

Another payload of the Ecipekac loader, which we call SodaMaster (a.k.a DelfsCake), is also a new fileless malware. In our research we found more than 10 samples of SodaMaster. All the collected samples of this module were almost identical, with the offsets and hex patterns of all functions perfectly matching. The only differences were in the configuration data, including a hardcoded C2, an encoded RSA key and additional data for calculating a mutex value.



Configuration of SodaMaster

When execution of this malware begins, it creates a mutex with a name in the reverse order of the CRC32 checksum calculated from the encoded RSA key and its following additional data. Then the malware randomly generates a value as an RC4 key for C2 communication. The first data block sent to the C2 server includes the user_name, the host_name, PID of the malware module, OS_version, socket_name, the generated RC4 key and the malware's elapsed running time.



C2 communication of SodaMaster

We confirmed four backdoor commands, coded as "d", "f", "l" and "s", in the recent SodaMaster sample. In addition, we also discovered an old SodaMaster sample which has only two commands. A description of each command is shown in the following table.

Command	Description	Compilation time of SodaMaster

		2019-01-07	**2019-06-10**
d	Create thread for launching downloaded DLL and call export function of the DLL.	Enabled	Enabled
f	Set value as RC4 key for the encrypted C2 communication	Not implemented	Enabled
l	Set value as sleep time	Not implemented	Enabled
s	Create thread for executing downloaded shellcode	Enabled	Enabled

Based on the analysis of the backdoor features of the SodaMaster module, the purpose of this malware is also to download and execute payloads (DLL or shellcode), like P8RAT. Unfortunately, we have not been able to obtain these payloads yet.

The SodaMaster module also shows an anti-VM feature. The malware looks for the presence of the registry key "HKEY_CLASSES_ROOT\\Applications\\VMwareHostOpen.exe" on the victim's system before proceeding to its main functionality. This registry key is specific to the VMware environment.

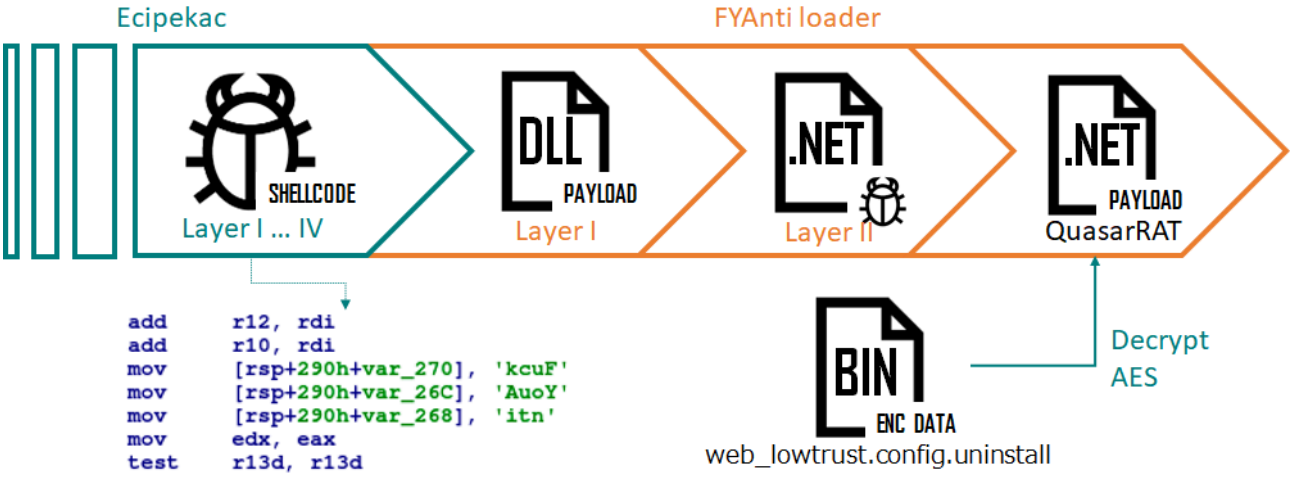
```
mov     eax, '\'
lea     r8, [rbp+phkResult] ; phkResult
lea     rdx, [rbp+Applications_VMwareHostOpen_exe] ; lpSubKey
                    ; "Applications\VMwareHostOpen.exe"
mov     [rbp+Applications_VMwareHostOpen_exe+18h], ax ; \
mov     eax, 'V'
xor     ebx, ebx
mov     [rbp+Applications_VMwareHostOpen_exe+1Ah], ax ; V
mov     eax, 'M'
mov     rcx, HKEY_CLASSES_ROOT ; hKey
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe], 700041h
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe+4], 6C0070h
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe+8], 630069h
mov     [rbp+Applications_VMwareHostOpen_exe+1Ch], ax ; M
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe+0Ch], 740061h
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe+10h], 6F0069h
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe+14h], 73006Eh
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe+1Eh], 610077h
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe+22h], 650072h
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe+26h], 6F0048h
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe+2Ah], 740073h
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe+2Eh], 70004Fh
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe+32h], 6E0065h
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe+36h], 65002Eh
mov     dword ptr [rbp+Applications_VMwareHostOpen_exe+3Ah], 650078h
mov     [rbp+Applications_VMwareHostOpen_exe+3Eh], bx
call    cs:RegOpenKeyW
```

SodaMaster anti-VM check

Another characteristic of SodaMaster is the use of a common obfuscation technique known as "stackstrings" to create the registry key in double-byte characters. We observed the same obfuscation technique used for a process name and DLL name in other SodaMaster samples.

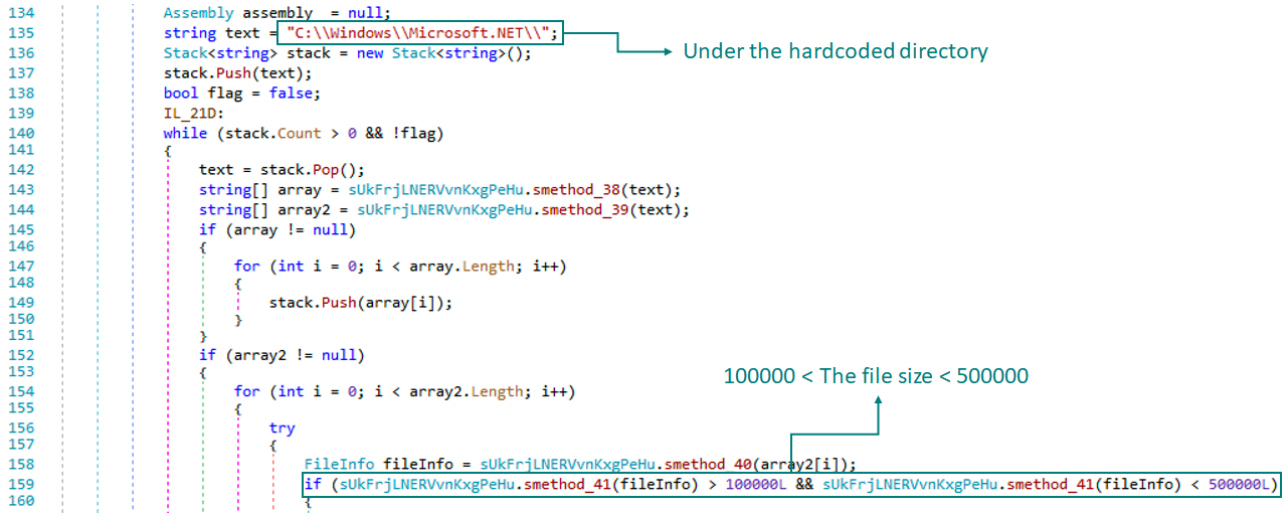
FYAnti loader for QuasarRAT

The last observed type of payload deployed by Ecipekac is a loader module named FYAnti loader. In the Ecipekac loader malware of the fourth layer, the DLL is loaded into memory and an export carrying the name "F**kY**Anti" is called. We named this loader "FYAnti" because of this distinct string. The execution flow of the FYAnti has two additional layers to implement the final stage, which is a QuasarRAT (a.k.a xRAT).



Infection flow of FYAnti loader

The first layer of the FYAnti loader decrypts an embedded .NET module and executes it using the CppHostCLR technique. The .NET module is packed using "ConfuserEx v1.0.0" and acts as yet another loader that searches for a file in the "C:\Windows\Microsoft.NET\" directory with file sizes between 100,000 and 500,000. The unpacked code is shown in the screenshot below.



Unpacked code of the second layer loader of FYAnti to search a file

In this instance, an encrypted file named "web_lowtrust.config.uninnstall" is found and used as the next stage module. The .NET module loads and decrypts this file using AES CBC mode. The decrypted payload is another .NET module named Client.exe which is QuasarRAT, a popular open-source remote administration tool. The

configuration data is stored in the binary with most of this data being encrypted by AES CFB mode and base64. The AES key is generated using the hardcoded string “KCYcz6PCYZ2VSiFyu2GU”in the configuration data.

Malware configuration of QuasarRAT

All loader modules and payloads are decrypted and executed in memory only.

Attribution

Based on our investigations, we assess with high confidence that the APT10 threat actor is behind the A41APT campaign. This attribution is based on the following points:

First, the hardcoded URL “www.rare-coisns[.]com” from an x86 SodaMaster sample was mentioned in the report from ADEO regarding APT10’s activity targeting the finance and telecommunication sectors of Turkey, also fitting the geolocation of the VirusTotal submitter.

Second, the similarity of the A41APT campaign with APT10 activities described in a Cylance blogpost. These include the Ecipekac Loader, FYAnti loader’s unique export name “F**kY**Anti”, using the CppHostCLR technique and QuasarRAT used as the FYAnti’s final payload. Moreover, as stated in the Symantec blogpost, the FYAnti loader, the export name “F**kY**Anti”, CppHostCLR technique for injection of .NET loader and the QuasarRAT were similar to the activities of the APT10 group discovered by the BlackBerry Cylance threat research team.

Last but not least, there are some similarities and common TTPs to those outlined in our previous TIP report on APT10 activities:

- Implementation of the hashing or crypto algorithms done manually by the malware developers instead of using Windows APIs, with some modifications;
- Use of calculated hash values (fully or partially) for some features like crypto keys, part of the crypto keys, key generation, mutex names and so on;
- Using the DLL side-loading technique to run a payload in memory;

- Using PowerShell scripts for persistence and also for lateral movement;
- Using exe for removing logs in order to hide their activities;
- Sending victim machine data such as username, hostname, PID, current time and other specifics – though this point is not unique to APT10 backdoors and is quite common in most backdoor families;
- Modifying implants shortly after security researchers publish their analysis of the actor's activities and TTPs;
- Targeting mainly Japan, alongside associated overseas branches or organizations related to Japan.

However, we observed some interesting differences in the A41APT campaign and previous activities:

- P8RAT and SodaMaster did not contain a malware version number as opposed to the previous malware instances used by APT10 such as LilimRAT, Lodeinfo and ANEL;
- As for the infection vector, we could not identify any spear-phishing email in this A41APT campaign, which is quite common in APT10 attacks.

Overall, APT10 is considered a large APT group running multiple simultaneous campaigns and, understandably, TTPs differ from one campaign to another. We believe the differences mentioned here for the A41APT campaign represent a normal variation of TTPs that can be expected in the case of such a large APT group.

Conclusions

We consider the A41APT campaign to be one of APT10's long-running activities. This campaign introduced a very sophisticated multi-layer malware named Ecipekac and its payloads, which include different unique fileless malware such as P8RAT and SodaMaster.

In our opinion, the most significant aspect of the Ecipekac malware is that, apart from the large number of layers, the encrypted shellcodes were being inserted into digitally signed DLLs without affecting the validity of the digital signature. When this technique is used, some security solutions cannot detect these implants. Judging from main features of the P8RAT and SodaMaster backdoors, we believe that these modules are downloaders responsible for downloading further malware that, unfortunately, we have not been able to obtain so far in our investigation.

We see the activity outlined in this report as a continuation of the activity we previously reported in our Threat Intelligence Portal, where, in 2019, this threat actor began targeting overseas offices of Japanese associations or organizations using the ANEL malware. The operations and implants of the campaign described in this report are remarkably stealthy, making it difficult to track the threat actor's activities. The main stealth features are the fileless implants, obfuscation, anti-VM and removal of activity tracks.

We will continue to investigate and track the activities of the APT10 actor, which are expected to keep improving its covertness with each year.

Appendix I – Indicators of Compromise

Note: *The indicators in this section are valid at the time of publication. Any future changes will be directly updated in the corresponding .ioc file.*

File Hashes (malicious documents, trojans, emails, decoys)

Ecipekac loader

be53764063bb1d054d78f2bf08fb90f3 jli.dll P8RAT
cca46fc64425364774e5d5db782ddf54 vmtools.dll SodaMaster
dd672da5d367fd291d936c8cc03b6467 CCFIPC64.DLL FYAnti loader

Encrypted Ecipekac Layer II, IV loader (shellcode)

md5 filename payloads

f60f7a1736840a6149d478b23611d561 vac.dll P8RAT
59747955a8874ff74ce415e56d8beb9c pcasvc.dll P8RAT
4638220ec2c6bc1406b5725c2d35edc3 wiaky002_CNC1755D.dll SodaMaster
d37964a9f7f56aad9433676a6df9bd19 c_apo_ipoib6x.dll SodaMaster
335ce825da93ed3fdd4470634845dfea msftedit.prf.cco FYAnti loader
f4c4644e6d248399a12e2c75cf9e4bdf msdtcuiu.adi.wdb FYAnti loader

Encrypted QuasarRAT

md5 filename payloads

019619318e1e3a77f3071fb297b85cf3 web_lowtrust.config.uninstall QuasarRAT

Domains and IPs

151.236.30[.]223
193.235.207[.]59
45.138.157[.]83
88.198.101[.]58
www.rare-coisns[.]com

Appendix II – MITRE ATT&CK Mapping

This table contains all the TTPs identified in the analysis of the activity described in this report.

Tactic	Technique	Technique Name
Initial Access	T1133	**External Remote Services** Uses vulnerabilities in Pulse Connect Secure to hijack a VPN session.
	T1078	**Valid Accounts** Uses stolen credentials to connect to the enterprise network as initial infection.
Execution	T1059.001	**Command and Scripting Interpreter: PowerShell**

		Uses PowerShell to download implants and remove logs.
	T1053.005	**Scheduled Task/Job: Scheduled Task** Creates a task for running a legitimate EXE with Ecipekac (malicious DLL) using DLL side-loading technique.
Persistence	**T1574.001**	**Hijack Execution Flow: DLL Search Order Hijacking** Uses a legitimate EXE file which loads Ecipekac (malicious DLL) in the current directory.
	T1574.002	**Hijack Execution Flow: DLL Side-Loading** Uses a legitimate EXE file which loads Ecipekac (malicious DLL) in the current directory.
	T1053.005	**Scheduled Task/Job: Scheduled Task** Creates a task for running a legitimate EXE with Ecipekac (malicious DLL) using DLL side-loading technique.
	T1078	**Scheduled Task/Job: Scheduled Task** Uses stolen credentials to connect to the enterprise network as initial infection.
Privilege Escalation	**T1574.001**	**Hijack Execution Flow: DLL Search Order Hijacking** Uses a legitimate EXE file which loads Ecipekac (malicious DLL) in the current directory.
	T1574.002	**Hijack Execution Flow: DLL Side-Loading** Uses a legitimate EXE file which loads Ecipekac (malicious DLL) in the current directory.
	T1053.005	**Scheduled Task/Job: Scheduled Task** Creates a task for running a legitimate EXE with Ecipekac (malicious DLL) using DLL side-loading technique.
	T1078	**Scheduled Task/Job: Scheduled Task**

		Uses stolen credentials to connect to the enterprise network as initial infection.
Defense Evasion	**T1574.001**	**Hijack Execution Flow: DLL Search Order Hijacking** Uses a legitimate EXE file which loads Ecipekac (malicious DLL) in the current directory.
	T1574.002	**Hijack Execution Flow: DLL Side-Loading** Uses a legitimate EXE file which loads Ecipekac (malicious DLL) in the current directory.
	T1053.005	**Scheduled Task/Job: Scheduled Task** Creates a task for running a legitimate EXE with Ecipekac (malicious DLL) using DLL side-loading technique.
	T1078	**Scheduled Task/Job: Scheduled Task** Uses stolen credentials to connect to the enterprise network as initial infection.
	T1070.003	**Indicator Removal on Host: Clear Command History** Removes Powershell execution logs using Wevtutil.exe.
	T1036	**Masquerading** Encrypted shellcode of Ecipekac was embedded in the legitimate DLL.
	T1497.001	**Virtualization/Sandbox Evasion: System Checks** Payloads of Ecipekac check a registry key and process names to identify VM environment.
Discovery	**T1057**	**Process Discovery** Checks the process of VMware and VirtualBox to identify the VM environment.
	T1082	**System Information Discovery**

		SodaMaster sends system information such as user_name, the host_name, PID of the malware module, OS_version, etc.
	T1012	**Query Registry** Checks a registry key of VMware to identify the VM environment.
Lateral Movement	**T1210**	**Exploitation of Remote Services** Uses vulnerabilities in Pulse Connect Secure to hijack a VPN session.
Command and Control	**T1071.001**	**Application Layer Protocol: Web Protocols** Cobalt Strike's stager uses HTTP protocol for communication with C2 server to disguise as a common jQuery.
	T1132.002	**Data Encoding: Non-Standard Encoding** SodaMaster uses an original data structure and RSA for the first communication, then uses RC4 for encryption.
