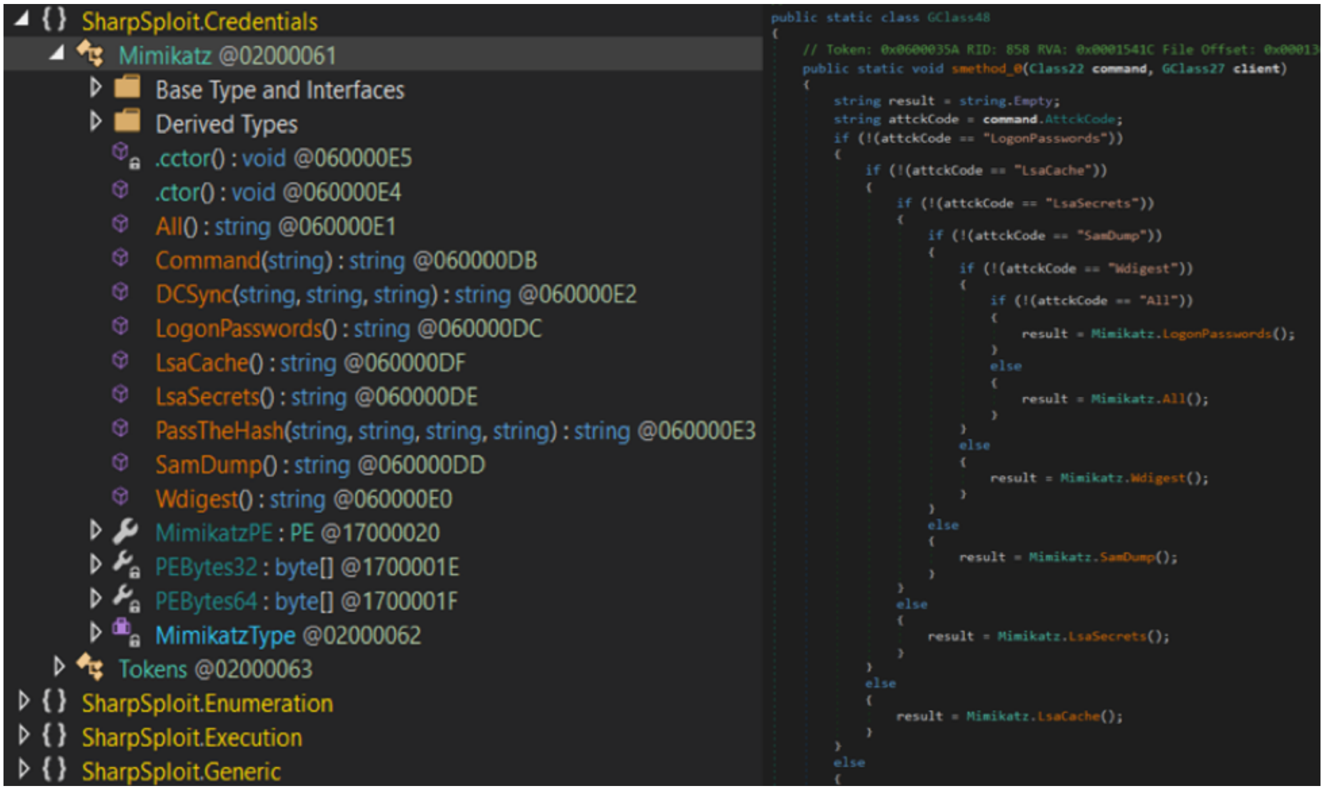
Uncovering New Activity By APT10

October 15, 2019



```
◢ { } SharpSploit.Credentials
    ◢ Mimikatz @02000061
        ▷ Base Type and Interfaces
        ▷ Derived Types
          .cctor() : void @060000E5
          .ctor() : void @060000E4
          All() : string @060000E1
          Command(string) : string @060000DB
          DCSync(string, string, string) : string @060000E2
          LogonPasswords() : string @060000DC
          LsaCache() : string @060000DF
          LsaSecrets() : string @060000DE
          PassTheHash(string, string, string, string) : string @060000E3
          SamDump() : string @060000DD
          Wdigest() : string @060000E0
        ▷ MimikatzPE : PE @17000020
        ▷ PEBytes32 : byte[] @1700001E
        ▷ PEBytes64 : byte[] @1700001F
        ▷ MimikatzType @02000062
    ▷ Tokens @02000063
▷ { } SharpSploit.Enumeration
▷ { } SharpSploit.Execution
▷ { } SharpSploit.Generic
```

```
public static class GClass48
{
    // Token: 0x0600035A RID: 858 RVA: 0x0001541C File Offset: 0x00013
    public static void smethod_0(Class22 command, GClass27 client)
    {
        string result = string.Empty;
        string attckCode = command.AttckCode;
        if (!(attckCode == "LogonPasswords"))
        {
            if (!(attckCode == "LsaCache"))
            {
                if (!(attckCode == "LsaSecrets"))
                {
                    if (!(attckCode == "SamDump"))
                    {
                        if (!(attckCode == "Wdigest"))
                        {
                            if (!(attckCode == "All"))
                            {
                                result = Mimikatz.LogonPasswords();
                            }
                            else
                            {
                                result = Mimikatz.All();
                            }
                        }
                        else
                        {
                            result = Mimikatz.Wdigest();
                        }
                    }
                    else
                    {
                        result = Mimikatz.SamDump();
                    }
                }
                else
                {
                    result = Mimikatz.LsaSecrets();
                }
            }
            else
            {
                result = Mimikatz.LsaCache();
            }
        }
        else
        {
```

FortiGuard Labs* Threat Analysis Report: This blog originally appeared on the enSilo website and is republished here for threat research purposes. enSilo was acquired by Fortinet in October 2019.*

Summary

In April 2019, we detected what we believe to be new activity by the Chinese cyber espionage group APT10. The discovered variants are previously unknown and deploy malware that is unique to the threat actor. These malware families have a rich history of being used in numerous targeted attacks against government and private organizations. The activity surfaced in Southeast Asia, a region where APT10 frequently operates.

Overview

Towards the end of April 2019, we tracked down what we believe to be new activity by APT10, a Chinese cyber espionage group. Both of the loader's variants, as well as the various payloads that we analyzed share similar Tactics, Techniques, and Procedures (TTPs) and code associated with APT10.

Although they deliver different payloads to a victim's machine, both variants drop the following files beforehand:

- jjs.exe - legitimate executable
- jli.dll - malicious DLL
- msvcrt100.dll - legitimate Microsoft C Runtime DLL
- svchost.bin - binary file

jjs.exe is a JVM-based implementation of a javascript engine that is part of the Java platform developed by Oracle, but in this case it served as a loader for the malware.

Among the payloads we found were PlugX and Quasar RATs. The former is well known to be developed in-house by the group with a rich history of being used in many targeted attacks against different government and private organizations. PlugX is a modular structured malware that has many different operational plugins, such as communication compression and encryption, network enumeration, files interaction, remote shell operations, and more.

The samples we analyzed originated from the Philippines. APT10 frequently targets the Southeast Asia region.

In this article we examine both versions of the loader along with their payloads, TTPs, and Command and Control (C&C) server information.

Loader

Abusing a Legitimate Executable

The loader starts out by running a legitimate executable, which is abused to load a malicious DLL instead of a legitimate one on which it depends. The method is known as DLL Side-Loading.

In both variants, the abused executable is jjs.exe, which loads jli.dll. The DLL exports the following functions:

- JLI_CmdToArgs
- JLI_GetStdArgs
- JLI_GetStdArgc
- JLI_MemAlloc
- JLI_Launch

The first function called by jjs.exe is *JLI_CmdToArgs*, which is implemented by the malware author and behaves differently in each variant.

Running The Payload

The malicious DLL maps the data file, svchost.bin, to memory and decrypts it. The decrypted content is a shellcode that is injected into svchost.exe and contains the actual malicious payload.

The decryption process resembles previous versions used by the group in PlugX\RedLeaves.

Figure 1: Decryption of the binary file

The injection flow is rather simple and is done by creating a process in suspended state, allocating memory with *VirtualAllocEx*, writing the shellcode with *WriteProcessMemory*, and running it using *CreateRemoteThread*. The complete execution flow is visualized in Figure 2, below.

Figure 2: Loader's execution flow

Variant 1

The first variant uses a service as its persistence method. It installs itself (jjs.exe) as the service and starts. When running in the context of the service, it performs the decryption and injection described above.

We have observed the variant delivering both Quasar and PlugX, which we'll discuss later on.

Among the different samples we tested, we encountered the following service names being registered:

- WxUpdateServiceInfo
- HxUpdateServiceInfo
- WinDefendSec
- Web_Client
- clr_optimization_v4.0.30319_31
- clr_optimization_v4.0.30319_37

Variant 2

Unlike the first variant, this variant uses the Run registry key for the current user under the name "*Windows Updata*" to ensure its persistence, rather than installing a service. This variant delivered the same PlugX DLL as the first loader.

Payloads

Modified Quasar RAT

The injected shellcode reflective loads in-memory an executable it reconstructs from data that it is bundled with it. The reflective load code is obfuscated, as function calls are made by dynamically resolving their addresses according to hashed values.

The executable tries to run conhost.exe from C:\Users\Public\Documents, and in case it doesn't exist, it turns to *ffca[.]caibi379[.]com* to download it. The code that sends the HTTP request seems to be buggy. While *wininet!InternetOpenW* (the Unicode version of the function) is used, an ascii value is provided as the User-Agent, so instead of "RookIE/1.0", the request headers will include inconsistent and meaningless values, as can be seen in Figure 3.

Figure 3: .NET downloader's HTTP request

In our analysis, the conhost.exe that was downloaded is itself another downloader written in .NET and disguised as a legitimate system executable.

This extremely simplified downloader only has the ability to download and execute a base-64 encoded executable from the hardcoded address using a simple System.Net.WebClient HTTP request: *ffca[.]caibi379[.]com/rwjh/qtinfo.txt*.

The downloaded payload is a modified Quasar RAT. This version contains the addition of SharpSploit to extract passwords from a victim's machine using the framework's built-in mimikatz capabilities.

Figure 4: Quasar Assembly information

Figure 5: Embedded SharpSploit code

Examining the sample's configuration, we found the following:

- C&C server: cahe.microsofts.org:443
- Mutex name: "QSR_MUTEX_rSifQNOVTwHrsBs2nd"
- A self-signed certificate issued to "MSGQ Server CA" (as seen in Figure 6)

Figure 6: Quasar's embedded certificate

PlugX

Following the injection to svchost.exe by the loader, the shellcode decrypts another part of itself and use RtlDecompressBuffer API to further unpack the PlugX DLL. The DOS and NT header's magic values, MZ and PE respectively, were replaced with VX, a typical behavior for PlugX payloads. This is meant to prevent security products and automated tools from identifying the executable headers when performing memory scans.

Like previous versions of PlugX, it collects information about the infected machine, such as the computer name, username, OS version, RAM usage, network interfaces, and resources.

In an attempt to generate noise around allocation and release of memory by the malware, the authors wrapped it with dummy calls to the *GetForegroundWindow* API function, as can be seen in Figure 7.

Figure 7: Dummy calls to GetForegroundWindow

This sample shares some similarities with the Paranoid PlugX variant. For example, it goes a long way to completely remove any sign of McAfee's email proxy service from the infected machine. Besides killing the process, it also makes sure to delete any related keys in the registry, and recursively deletes any related files and directories on the machine. The same behavior was observed by in the paranoid variant as part of a VBScript that the dropper runs.

Typically, APT10 tends to employ a namesquatting scheme in their domains that aims to confuse the observer by posing as a legitimate domain. In the configuration bundled to the samples we found the following:

- The sample in the first loader communicates with *update[.]microsoft**s**[.]org* with DNS over TCP.
- The sample in the second loader communicates with *update[.]kaspr**es**k**s**y[.]com* over HTTPS.

Threat Intelligence

When examining the first loader variant's domain (ffca[.]caibi379[.]com), we discovered that it resolved to the following IP addresses according to VirusTotal:

Date Resolved	IP
2019-04-24	27.102.128.157
2019-03-23	27.102.127.80
2019-01-30	27.102.127.75

While all the IP address ranges are listed under the name of an organization located in South Korea, the domain itself was registered in Hong Kong.

Reverse lookup on the IP addresses shows that some of them used to be resolved from another domain - **[.]microsofts[.]org*. As mentioned before, *cahe.microsofts.org* is the command and control server for the Quasar payload, and *update.microsofts.org* is used for the PlugX payload delivered by the first loader variant.

The PlugX's domains resolve to the following addresses:

Date Resolved	IP
2019-04-27	27.102.66.67
2019-02-19	27.102.115.249
2019-03-15	27.102.127.80

While looking at the other subdomains of kaspresksy[.]com we discovered that:

- download[.]kaspresksy[.]com resolves to 27.102.113.118
- api[.]kaspresksy[.]com resolves to 27.102.114.246

27.102.114.246 used to previously resolve from the following domains:

- smsapi[.]tencentchat[.]net
- onedrive[.]miscrosofts[.]com

We noticed a password-protected zip named "*Chrome_Updata*" being associated with the *download[.]kaspresksy[.]com* domain. The zip contained a sample of the Poison Ivy malware, which is also known to be used by APT10. The same executable was also seen communicating with 27.102.115.249, which also appeared to be mapped to *update[.]kaspresksy[.]com*.

All of the overlaps in the network infrastructure make it very reasonable to assume that the same group is operating both variants.

Conclusion

Both variants of the loader implement the same decryption and injection mechanism.

Looking at the history of APT10, one can notice major similarities in the details we highlighted in this post:

- A bundle of legitimate executables are used to sideload a custom DLL, along with storing the payload in a separate, encrypted file.
- Use of typosquatting domain names similar to real, legitimate tech companies.
- Unique malware families both developed by, and associated with, the group.
- Using C&C servers located in South Korea.

Some of the mentioned domain mappings were recently updated. Also, the certificate embedded in the Quasar sample was issued at 22.12.2018, which correlates with the file's compilation date. This may indicate that these samples are a part of a testing environment or a short-lived attack that is already finished. Either way, it's safe to say that the threat actor behind APT10 is still active and that we have yet to see the last of the group.

IOCs

Loader v1:

41542d11abf5bf4a18332e9c4f2c8d1eb5c7e5d4298749b610d86caaa1acb62c (conhost.exe downloader jli.dll)
29b0454db88b634656a3fc7c36f318b126a83ae8fb7f73fe9ff349a8f8536c7b (conhost.exe downloader svchost.bin)
02b95ef7a33a87cc2b3b6fd47db03e711045974e1ecf631d3ba9e076e1e374e9 (PlugX jli.dll)
e0f91da52fdc61757f6a3f276ae77b01d2d1cc4b3743629c5acbd0341e5de80e (PlugX svchost.bin)

Loader v2:

f13536685206a94a8d3938266f100bb2dffa740a202283c7ea35c58e6dbbb839 (PlugX jli.dll)
c8d86e9f486d23285b744279812ef9047a0908e39656c2ea4cdf3e182f80e11d (PlugX svchost.bin)

.NET Downloader (conhost.exe):

96649c5428c874f2228c77c96526ff3f472bc2425476ad1d882a8b55faa40bf5

Quasar RAT:

0644e561225ab696a97ba9a77583dcaab4c26ef0379078c65f9ade684406eded

Domains:

update[.]kaspresksy[.]com
download[.]kaspresksy[.]com
api[.]kaspresksy[.]com
ffca[.]caibi379[.]com
update[.]microsofts[.]org
ppit[.]microsofts[.]org
cahe[.]microsofts[.]org

IP Addresses:

27.102.128.157
27.102.127.80
27.102.127.75
27.102.66.67
27.102.115.249

Solutions

The FortiEDR platform is capable of detecting the this threat on both pre-execution and post-execution.

The Webfiltering service categorized all the network IOCs as malicious.

Learn more about FortiGuard Labs threat research and the FortiGuard Security Subscriptions and Services portfolio. Sign up for the weekly Threat Brief from FortiGuard Labs.

Learn more about Fortinet's free cybersecurity training initiative or about the Fortinet Network Security Expert program, Network Security Academy program, and FortiVet program.