



DeadlyKiss

Hit One to Rule Them All

// Summary

In the first days of September 2019, Telsy Threat Intelligence Division received a variant of a strange and initially mysterious malware from a stream of thousands of samples coming from a partner operating in the telecommunications and internet connectivity sector.

Although this sharing had not been accompanied by much information about it, it immediately seemed quite clear that the object under analysis was not something very common to be observed.

Very quickly, indeed, a clear picture emerged that led to the observation of an advanced, rare and extremely evasion-oriented malware, which implements effective stratified obfuscation techniques and adopts many solutions dedicated to operate “*under the radar*”.

The file in question was already present in the most common malware analysis and file evaluation platforms (this factor prompted us to publicly release a first analysis and assessment paper about it) but what immediately struck our attention has been a very low detection rate.

On September 13, 2019, in fact, our file could be detected only by one (1) engine with the generic signature “*Generic.mg.f010b0b7681ede24*”. Our internal machine learning based detection engine was around a risk rate of only 60%.

These evidences have obviously raised up our levels of attention towards that sample, coming to conclude, after few days and thanks to the collaboration with other partners and individual researchers (thank to *@daphiel*), that what we were observing was one of

the components of a much more complex suite and that very probably it is destined to support cyber operations directed towards a very limited number of targets.

Finding no publicly known evidence regarding the family of samples in question or regarding any threat actor that made use of it, we came to the conclusion that what we were observing belonged to a new case in the world of APTs (Advanced Persistent Threats) and the probable discovery of a still not publicly known threat actor.

We arrived at this assumption for the following reasons:

1. All the retrieved malware samples are structured and complex. They can be associated for sure with a development and maintenance cycle compatible with well organized actors.
2. We observed a multi-layered obfuscation techniques of the executables which is probably the result of a solution specifically designed for this purpose. This factor is also compatible with organized and structured threat actors.
3. All information we got about this threat confirm telecommunications companies and internet service providers as main targets. This specific sector in which the actor operates pushed us to classify this threat as "*persistent*".
4. The dates of submission of our first file to the most common malware analysis platforms are back to December 2018. Throughout this time the detection rate of the executable has practically settled at extremely low levels (1/66). The ability to remain

so unnoticed is typical of malware used in an extremely selective manner as well as of threat actors with high technical skills.

5. We observed some features of the backdoor which can allow to communicate with Command and Controls (CnC) at pre-established times and intervals (days/weeks/specific hours). These are likely implemented in order to evade network-based detection. This confirms to us the attention of the actor in operating silently.

For these and other reasons still under investigation, we have decided to refer to this as well as to all the malicious components related to it as "**DeadlyKiss**", due to the planned conjunction of two specific malicious modules during the kill chain.

// Loader

As already mentioned we started our investigation by a single dll PE file.

Almost immediately we realized that this component could not individually represent the total of the entire malicious suite. Having initially tried to explode this single file it was clear that it represented only a part of the threat.







Specifically we immediately cataloged this component as the probable "*loader*" of one or more additional components. It appears quite complex to explode manually;

Indeed, the actor made use of strong code obfuscation and string encryption in order to make analysis harder.

```
u3 = GetLastError() + 41;
if ( (signed int)u3 > 59 )
{
    if ( u3 == 616 )
    {
        div(913, 13);
    }
    else if ( u3 == 719 )
    {
        div(996, 8);
    }
    else
    {
        LABEL_35:
        div(72, 16);
    }
}
else if ( u3 == 59 )
{
    div(502, 87);
}
else
{
    switch ( u3 )
```

Example of obfuscated code

This technique is very effective in slowing down and discouraging any manual analysis, both static and dynamic, and inevitably leads to problems when creating code-based generic detection rules for this malware family. Anyway, we focused our attention at *DllRegisterServer()* by looking at the exported routine code.

 DllCanUnloadNow	1000D690	1
 DllGetClassObject	1000D4C0	2
 DllRegisterServer	1000D060	3
 DllUnregisterServer	1000D290	4
 ServiceMain	1000D7D0	5
 DllEntryPoint	1001E08C	[main entry]

Exported function of the primary dll

```
loc_1000CF2A:      ; CODE XREF: sub_1000CE00+1237j
                 call    ds:GetVersion
                 cmp     eax, 207h
                 jg     loc_1000CFC3
                 jz     short loc_1000CFAD
                 cmp     eax, 4           ; SWITCH/CASE
                 ja     loc_1000CFCA     ; jumptable 1000CF46 default case
                 jmp     ds:off_1000D040[eax*4] ; JUMP
; -----
```

Switch / Case code segment

regsvr32.exe, once executed, calls the routine *DllRegisterServer* exported by the loader that goes to start the infection cycle.

The loader is able to handle the installing, uninstalling and the reinstalling of the payload handling five (5) arguments provided during the installation phase.

Written in C++ and probably obfuscated at source-code level, this malicious library can work as in-process COM server or directly as service.

Indeed, we can note the presence of *ServiceMain* routine designed to support service-mode installations.

// Payload

The payload shared with us appears as an encrypted file which must be loaded by the first module to start the infection cycle.

Following a frame of payload as it appeared originally:

```

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
87 EA EB 11 CF 1D B8 C6 CA 97 21 2A BC 1F 43 36 †êë.Ï.,ÆÊ-!*¼.C6
77 2D D2 B5 AD 51 27 28 37 EE 11 47 CF 10 94 C1 w-òµ.Q'(7î.GÏ."Á
C8 A6 15 11 7B FB D8 C3 56 5A 4A 34 7C 82 24 5C È!..{ûøÃVZJ4|,$\
E4 43 8A 48 D5 AC FF 88 14 F9 E7 20 09 E1 6F 53 äCŠHÖ-ÿ^.ùç .áoS
1C 50 F4 89 FE 8C F5 B3 30 00 CA E4 48 78 A6 BA .Pô%þEö°0.ÉäHx|°
04 54 8D E8 52 54 DB 75 CE EC 4A 46 DE 1A C2 A3 .T.èRTÛuîìJFP.Âf
C1 64 9A EE 5B B7 09 3C FA 04 ED 79 5C 39 BF E6 Ádšî[·.<ú.íy\9;æ
1A 71 BE 02 B9 7B A6 43 D5 09 64 61 24 43 F0 B0 .q¾.¹{|CÖ.da$Cö°
28 57 96 9B 16 E4 0C A2 58 4A CB 50 77 83 D7 31 (W->.ä.φXJËPwf×1
1B 3E 44 CC 84 68 05 0A 41 54 71 83 F9 DA 00 15 .>DÏ,,h..ATqfùÛ..
FD 60 0B 17 6F E7 84 3C B3 11 C6 67 C9 8F 17 17 ý`. .oç,,<°.ÆgÉ...
CF EC AF 9F E9 92 89 67 1A 0D 58 0A 99 F7 11 C9 ìì~ÿé'¼g..X.™÷.É

```

hex frame of payload as it appeared originally

Its decryption routine uses AES-256 algorithm. The key is calculated from an hardcoded string and a part of the payload.

Once running, it performs the typical operations of a malware dedicated to reconnaissance and cyber-espionage operations.

Indeed, the malware includes features aimed at data exfiltration and remote control of the victims.

These are composed of:

1. The ability to exfiltrate system data and informations.
2. The ability to download and upload arbitrary files.
3. The ability to execute arbitrary commands, programs and libraries on infected machines.

4. The ability to set persistency mechanisms within the infected systems and to survive to a system reboot.
5. The ability to detect blacklisted running processes such as, for example, AV products.
6. The ability to modify timestamps related to the implanter and to any additional downloaded components. This is likely to make harder the incident response activities and / or post-incident forensic investigations.

The implant communicates to the outside world through HTTP requests.

As early mentioned, the threat actor has the ability to choose in detail the time intervals between one communication and the others in such a way as to reduce the possibility of a detection based on network anomalies.

The CnC servers are setted to provide further instructions and commands to be executed once the installation and persistence phase is complete.

Communications to command and control centers make use of both encoding and cryptographic algorithms in order to keep the exfiltrated information and the provided commands back private and in order to avoid a recognition based on communication payloads.

// Persistence

As mentioned, the implant can operate as an in-process COM or as a system service.

In a similar way, it can achieve persistence in the affected system in two different ways (as observed by dynamic analysis):

1. Through the adding of a new Scheduled Task:

A new scheduled task is created by the implanter and configured to operate at logon. Task is created with +h attribute.

2. Through the adding of new System Service:

A new service is registered by the implanter and configured to start at logon.

// Threat Assessment

All the information obtained regarding this threat, both through internal research and through relationships with third parties, has led to evaluate this malware as something extremely oriented to infect a very limited number of targets.

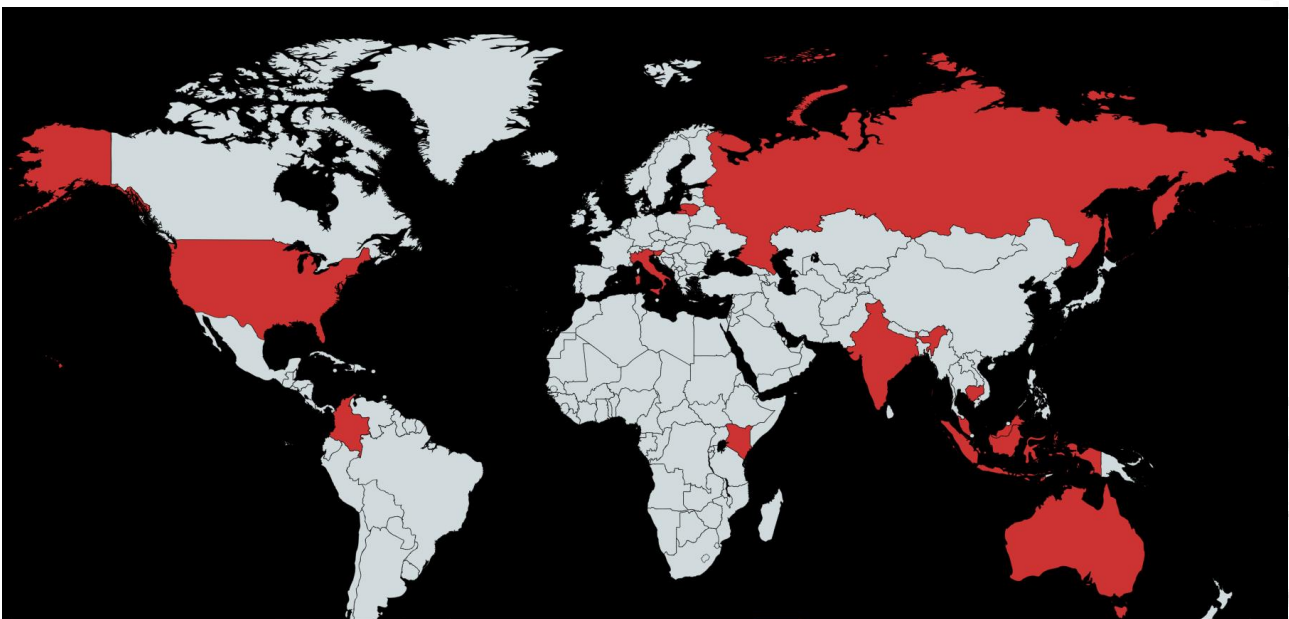
All the targets on which we have potential evidence of compromise belong to the world of Internet Service Providers (ISP). Although it is easy enough to imagine what the actor's final goals are judging by the functionality of the malicious toolkit, we cannot speculate on any broader scenarios as we currently do not even have the opportunity to assess the actor's possible behavior within the target networks.

However, the fact concerning the final targets remains; Internet Service Providers (ISP) are, indeed, an extremely strategic resource if we consider the cyber threat landscape.

An actor able to take possession of such networks could not only access information concerning the victim himself, but also those concerning all its clients, including simple users, institutions and organizations. Others minor evidences suggest interests in military and defence network as well.

As for the potential spread, all the information we obtained initially saw the Asian region as the main targets pool. However, we have evidence that could suggest targets also in other parts of the world, including US and RU.

The following is a geographical map that shows potential malicious activity related to the threat in question:

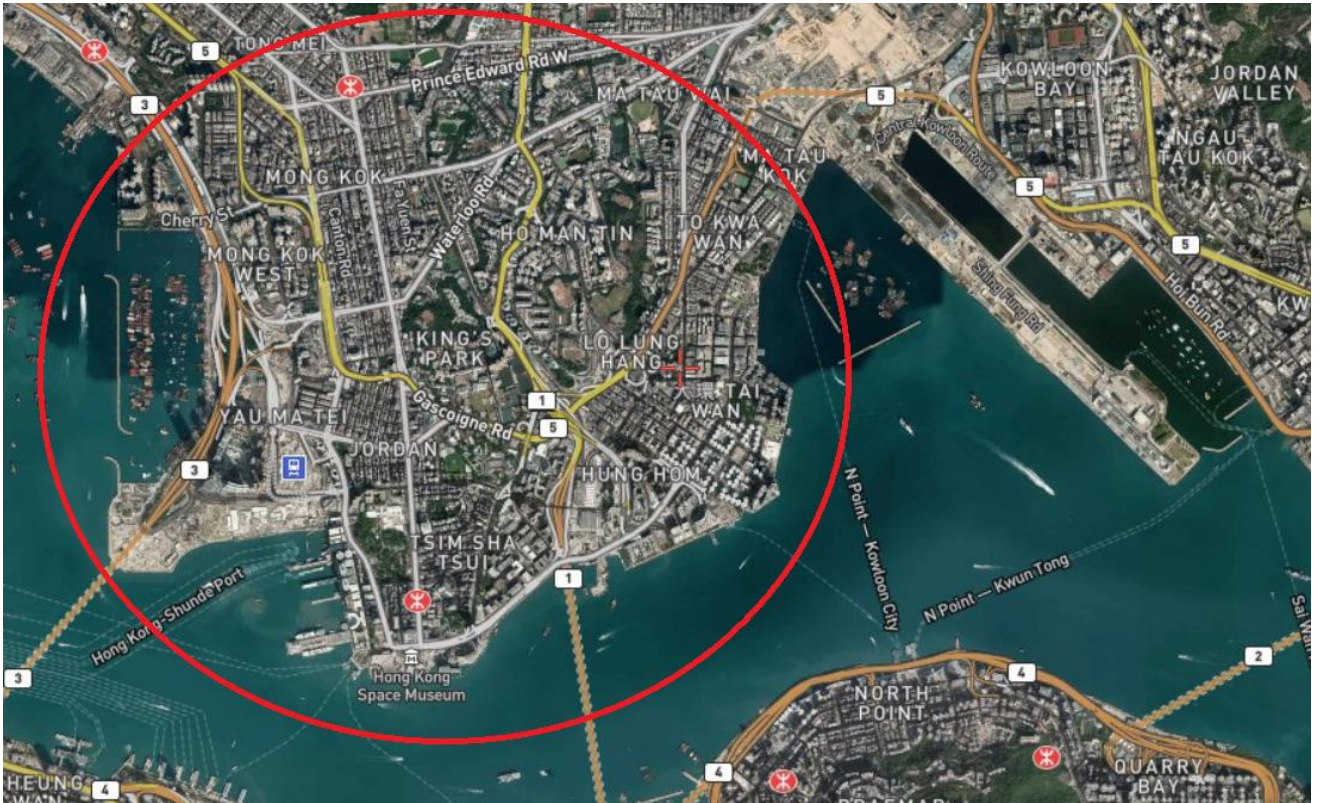


// Attribution

Although our analysis is still in progress, at this time we do not have enough evidences to assume an attribution. However we have high confidence that this is a very well structured and organized threat actor. In consideration of an extensive use of cryptography, obfuscation and techniques aimed at reducing the probability of detection, we feel we can catalog it also as an “*advanced*” one.

Although we are not in a position to provide even a hypothetical attribution of what we have observed, we can report some objective evidence regarding the operations and tools used:

1. Actor is operating at least since 2016 according to compilation timestamps and related infrastructures, succeeding to be unnoticed for a long time. One of domain name related with high confidence to the malicious infrastructure has been registered on 2015, suggesting a still earlier start of activities.
2. Written in C++, the malicious implanters show a very high level of accuracy in the development style. This suggests a dedicated development team behind it.
3. Historical information regarding some registered domain names refers to geographical areas located in China, more precisely in the south of Hong Kong, as showed by the following picture:



Over the time the information publicly obtainable from the network infrastructures has been reduced, perhaps to suggest an increase in the actor's attention to operational security.

However, it's important to note that these type of information can be easily falsified by the actor itself during the registration phase to mislead researchers and analysts.

To conclude, although we have a high confidence in the fact that this is a very well organized and efficient threat actor, we are not able to provide an attribution and / or a similarity rate with what has already been observed over time by other research teams or internally.

// Indicators of Compromise

Type	Observable
Domain	orionfile[.]com
Domain	tawaranmurah[.]com
URL	http://www[.]tawaranmurah[.]com/home/newitems.php
URL	http://www[.]orionfile[.]com/rssfeeds/lang1.php
IP	81[.]4[.]100[.]197
Artifact	479b9e6d7a5d35d8854756be845de34e270214d145ddb8f70b0c9755b4a62a8
Artifact	6373ccea42086db2ec0d7d801540206ad7cd16130f0fdc0bf1d5e20cca876d6
Artifact	98214a8ff23135a1e92e2ab029a4806cd1501d0a190798cf37bec90b2b20729e
Artifact	c0d70c678fcf073e6b5ad0bce14d8904b56d73595a6dde764f95d043607e639b
Artifact	d7ce022a6bad033fd22b76259ed4071b2d76f1ec547b2924411824aa7362e442
System	[install_path]\packages\stop.tmp

// Technical and Tactical Procedures

[+] Actor encrypts and obfuscates data before being exfiltrated in order to hide the information from detection.

[+] Actor uses Scheduled Task to achieve persistence on victim system.

[+] Actor creates a new service to set persistence or run payloads.

[+] Actor uses obfuscation in order to create code that is more difficult to understand.

[+] Actor uses AES to decrypt payloads.

[+] Actor refers to well known and commonly used software names in persistence techniques (tasks / services).

// Detection Rule

```
rule APT_DeadlyKiss_DLL_v1 : UNKNOWN ORIGIN THREAT ACTOR {
meta:
description = "Detects DeadlyKiss APT Loaders"
author      = "Emanuele De Lucia"
date        = "2019-09-18"
tlp         = "white"
strings:
/* Common exports in dlls dataset */
$export1 = "DllRegisterServer" ascii
$export2 = "DllCanUnloadNow" ascii
$export3 = "DllGetClassObject" ascii
/* SHELL */
$shell = "SHELL32.dll" ascii
/* STRINGS */
$name1 = "Java(TM) Platform SE 6" fullword wide
$name2 = "Intel(R) Chipset Device Software" fullword wide
/* Extracted matched clear common functions for generic catch */
$func1 = { 55 8B EC 8B 55 0C 8B 45 08 83 65 0C 00 D1 EA 85 C0 74 14 81 fA }
$func2 = { 55 8B EC 83 EC 4C 83 65 FC 00 EB 07 8B 45 FC 40 89 45 FC 83 7D }
/* XOR LOOP */
$xor = { C1 E9 02 32 48 FC 32 }
condition:
uint16(0) == 0x5a4d and filesize < 700KB and
((all of ($export*) and $shell and 1 of ($name*)) or
(all of ($func*) and $xor))
}
```