Telsy

Cyber Reports

REMCOS and Agent Tesla loaded into memory with Rezer0 loader

15/09/2021

INDEX

1. Introduction

During the last month, Telsy encountered a new phishing campaign, with banking and payment lure, targeting email of Italian government (*.gov.it) and companies of some industries. The threat actor's goal was to install ***Remcos*** –a remote control tool– or ***Agent Tesla*** –information-stealing Trojan- on the victims' computers. The attackers initially sent fake emails that appeared to be from several legitimate companies or in some cases from emails compromise, moreover some binaries were hosted on compromised web sites.

Remcos is a remote control and surveillance software developed and distributed by an organization called *Breaking Security*[1] [2]. Since it first appeared on the market, *Remcos* has gained popularity among cyber-attackers and even made it into the arsenal of APT actors like the *Gorgon Group*[3]. ***Agent Tesla***[4] is an information-stealing Trojan that has been around since 2014. It's in active development, constantly being updated and improved with new features, obfuscation, and encryption methods, also used by the APT *SilverTerrier* group[5].

In the attack we observed, the malware used several evasion techniques to ensure its success. Among the most interesting are the following:

- Mapping DLLs into the address space and resolving functions in the mapped file instead of the conventional LoadLibrary + GetProcAddress function calls
- Multiple layers of code injection to hide malicious actions behind seemingly legitimate processes
- Anti-reverse-engineering tricks to force a human malware analyst to spend more time on the sample.

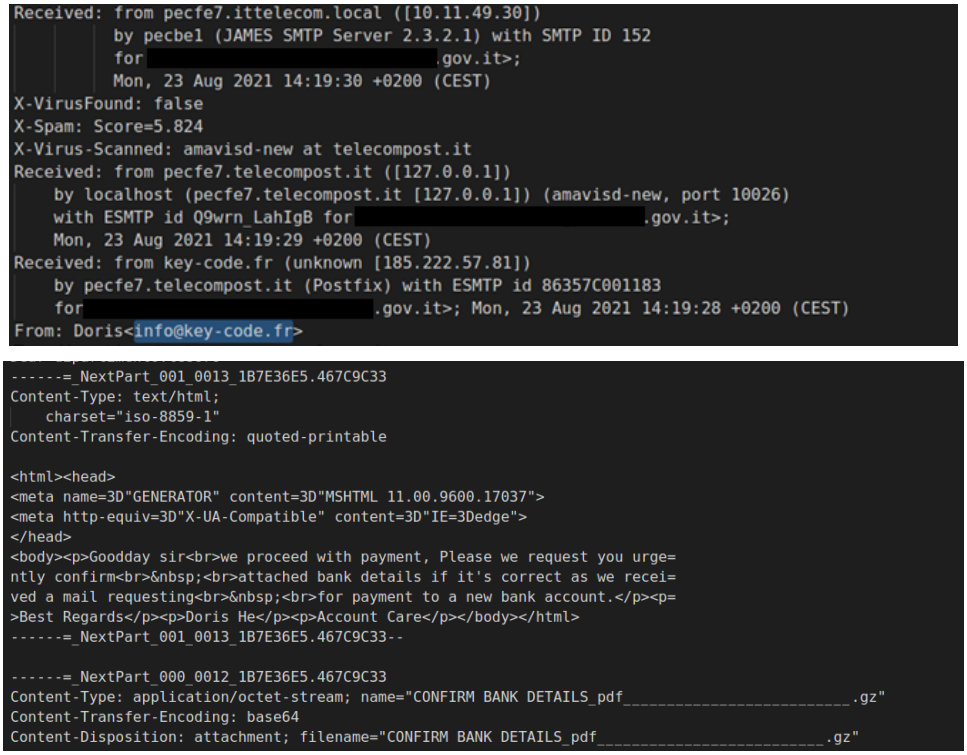
In some cases the emails contain an archive with an executable inside while in others a document (RTF) that releases an executable. All executables are written in *dotNET* and have multiple stages where the payload, usually a DLL, is loaded from the resources.

All emails belong to the same phishing campaign as the malicious executables were hosted by the same compromised website, and all have the same infection chain. In fact, all malicious executables load and run the same library in memory, called *SafeLSAPolicy.dll*.

[1] https://attack.mitre.org/software/S0332/
[2] https://breaking-security.net/remcos/
[3] https://attack.mitre.org/groups/G0078/
[4] https://attack.mitre.org/software/S0331/
[5] https://attack.mitre.org/groups/G0083/

The latter has the task of loading into memory a library called *Uint16.dll* which has the purpose of performing the actual final stage.

2. Analysis



```
Received: from pecfe7.ittelecom.local ([10.11.49.30])
          by pecbe1 (JAMES SMTP Server 2.3.2.1) with SMTP ID 152
          for [redacted].gov.it>;
          Mon, 23 Aug 2021 14:19:30 +0200 (CEST)
X-VirusFound: false
X-Spam: Score=5.824
X-Virus-Scanned: amavisd-new at telecompost.it
Received: from pecfe7.telecompost.it ([127.0.0.1])
    by localhost (pecfe7.telecompost.it [127.0.0.1]) (amavisd-new, port 10026)
    with ESMTP id Q9wrn_LahIgB for [redacted].gov.it>;
    Mon, 23 Aug 2021 14:19:29 +0200 (CEST)
Received: from key-code.fr (unknown [185.222.57.81])
    by pecfe7.telecompost.it (Postfix) with ESMTP id 86357C001183
    for [redacted].gov.it>; Mon, 23 Aug 2021 14:19:28 +0200 (CEST)
From: Doris<info@key-code.fr>
```

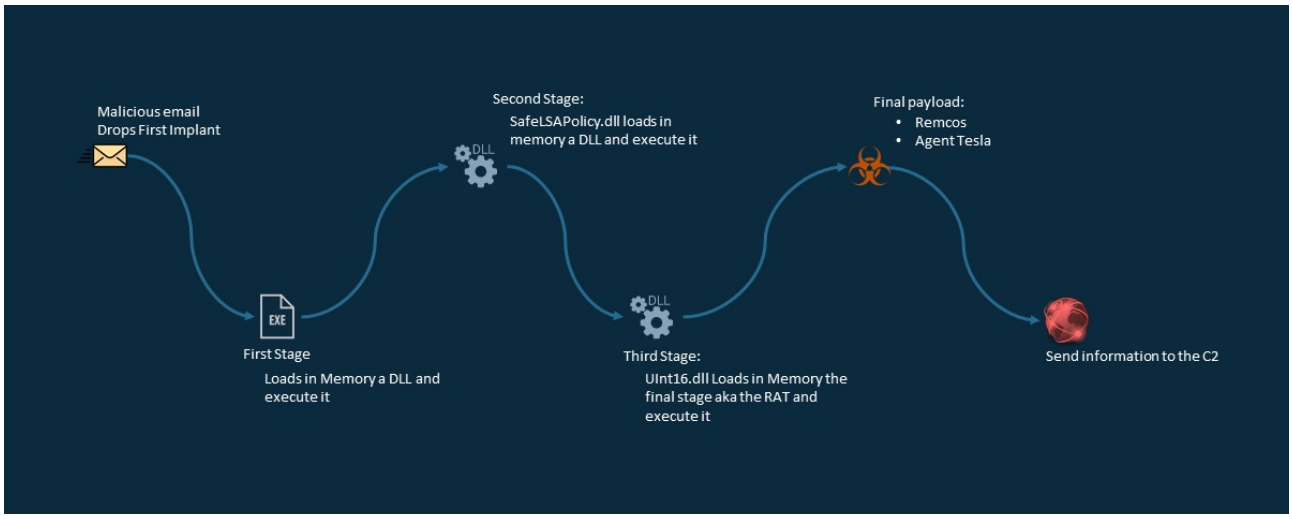
```
------=_NextPart_001_0013_1B7E36E5.467C9C33
Content-Type: text/html;
    charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

<html><head>
<meta name=3D"GENERATOR" content=3D"MSHTML 11.00.9600.17037">
<meta http-equiv=3D"X-UA-Compatible" content=3D"IE=3Dedge">
</head>
<body><p>Goodday sir<br>we proceed with payment, Please we request you urge=
ntly confirm<br> <br>attached bank details if it's correct as we recei=
ved a mail requesting<br> <br>for payment to a new bank account.</p><p=
>Best Regards</p><p>Doris He</p><p>Account Care</p></body></html>
------=_NextPart_001_0013_1B7E36E5.467C9C33--

------=_NextPart_000_0012_1B7E36E5.467C9C33
Content-Type: application/octet-stream; name="CONFIRM BANK DETAILS_pdf_____.gz"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="CONFIRM BANK DETAILS_pdf_____.gz"
```

Sample Email Delivering Remcos/Agent Tesla

In this case analyzed the phishing starts with an email comes from a compromise email: *info@key-code.fr* and having an archive as attachment that contains an executable.

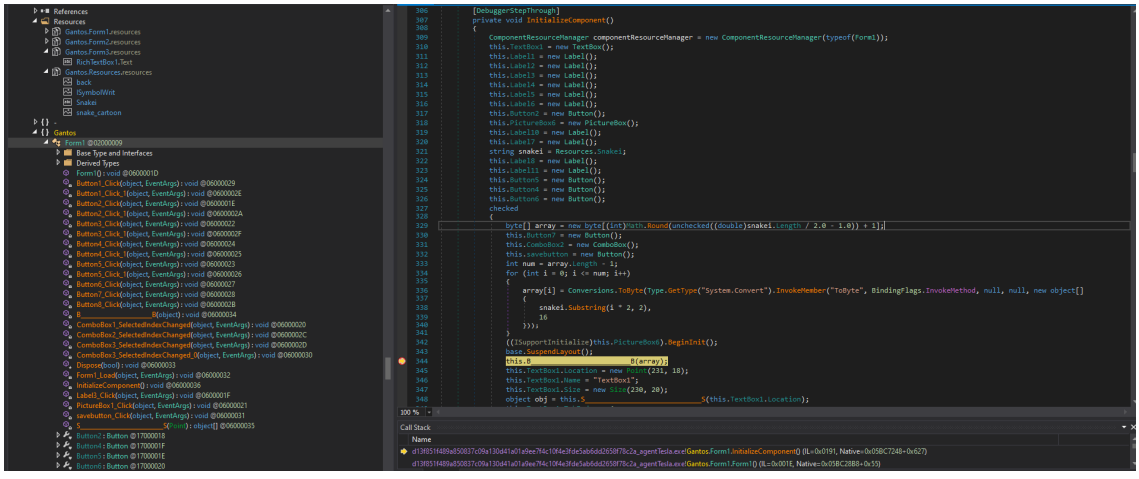
The executable loads in memory the first stage library, i.e. *SafeLSAPolicy.dll*, that is exactly the same library in every case examined, that in turn load in memory from resources a PNG image that it is parsed and decrypted to obtain the second stage.

The second stage is a library executed in memory and is the *ReZer0* loader version 4 in all the phishing cases analyzed. Finally, the second stage loads in memory the final stage which, in some case is an *Agent Tesla* while in the others is the RAT *Remcos*.

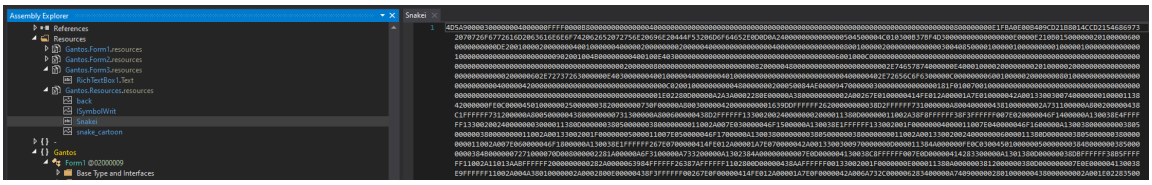


In the previous schema are resumed the similarities in the infection chain, moreover a great evidence that makes possible definitely to link all the cases to the same campaing is that all the first implants load in memory the same *SafeLSAPolicy.dll*. This library is written in dotNET and its hash in every analyzed case is: *e8d0b6339e94192eaaca32c812f914e60576dca6*.

The SafeLSAPolicy.dll library is loaded in different ways in the image below it is loaded directly from the resources, as it is possible to see in lines: 321, 329.

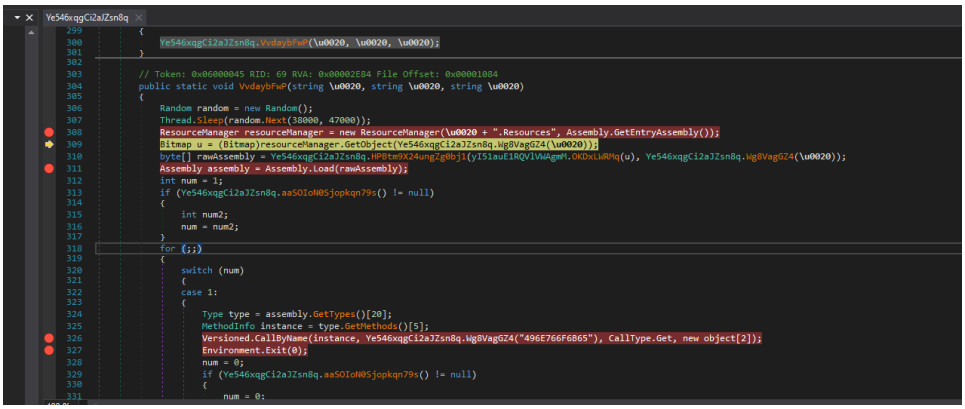


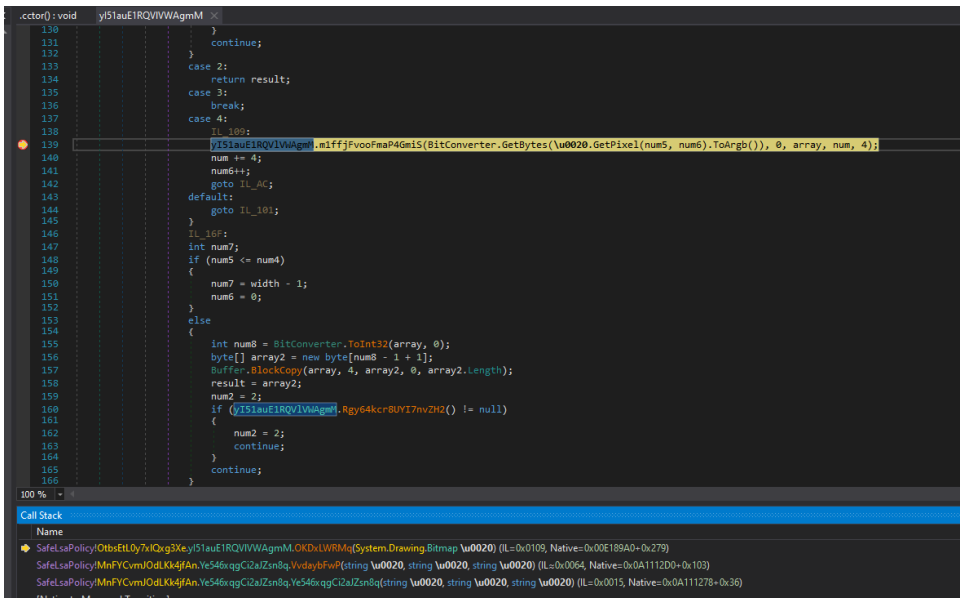
In particular the library is stored in the resource in hexlified format.



On the other hand in the second stage, *SafeLSAPolicy.dll* loads in memory *ReZer0V4* loader, named *UInt16.dll*, in the same way, i.e. takes from the main executable implant a PNG image from the resources and after a parse and a decryption obtain UInt16.dll and runs it.

The image resource is read, then each pixel starting from the top left angle to the bottom right angle is taken in RGBA format then each value of the pixel self is stored in the output array.





The first 4 bytes obtained are the length of the PE. Then the encrypted PE, i.e. array[4:], is decrypted with a xor operation.

PE_plain[i] = PE_enc[i] ^ key_wide[i%len(key_ascii)] ^ PE_enc[len(PE_enc) – 1]

The third stage is the stage that has the purpose to load in memory the final payload, i.e. the real malware used to infect the system.

In this stage, *UInt16.dll*, according to its configuration and code similarities has been identified as ***ReZer0V4*** Loader. *ReZer0V4* Loader is a dotNET configurable loader, that allows to load a malware from URL or to load it from resources and execute it using the process hollowing technique. These working-mode are specified through the configuration embedded in the loader self. Moreover this loader can check the existence of the SandBox or Virtual environment.

The features of the loader are resumed below:

- Download&Exec from URL
- Persistence through sctasks
- SandBox Detection
- VM Detection
- Disable Windows Defender Check

The configuration has 36 options. Some of these are not used some other implies a different behaviour of the loader self.

Each configuration value is contained in the same string and the separator between two configuration values is “||”. So, splitting the string by the “||”an array with all the configuration values is obtained. Some values are not set, based on the choices of the attacker.

A configuration string example:
"0||1||0||0||0||||||0||0||0||0||||||||||||||0||0||0||0||0||0||0||0||v4||2||3518||1||0||||||0||0||1||40||0||"

Below a table that explain every field.

Index configuration array	***Value&Purpose***
configuration[0]	*== 4 load the plugin from the resource into a new self process memory else != 4 inject the plugin from the resource into the system process to execute*
configuration[1]	*register scheduled tasks*
configuration[2]	*not used*
configuration[3]	*not used*
configuration[4]	*download and execute file*
configuration[5]	*URL to download file*
configuration[6]	*file downloaded in %appdata%filename*
configuration[7]	*detect Virtual Environment*
configuration[8]	*detect Sandobox*
configuration[9]	*if user is in role Administrator disable Windows Defender*
configuration[10]	*not used*
configuration[11]	*not used*
configuration[12]	*not used*
configuration[13]	*not used*
configuration[14]	*not used*
configuration[15]	*not used*
configuration[16]	*not used*
configuration[17]	*not used*
configuration[18]	*not used*
configuration[19]	*not used*
configuration[20]	*not used*
configuration[21]	*not used*
configuration[22]	*not used*
configuration[23]	*not used*
configuration[24]	*not used*
configuration[25]	*version major*
configuration[26]	*version minor*
configuration[27]	*version versioning*
configuration[28]	*check if exists mutex named*
configuration[29]	*Show messageBox*
configuration[30]	*message box title*
configuration[31]	*message box description*
configuration[32]	*message box button id*
configuration[33]	*message box icon id*
configuration[34]	*sleep*
configuration[35]	*Sleep duration*

Even though the hash of the UInt16.dlls are different between the phishing cases analyzed, their code is basically the same. What changes is the payload, mutex name, configuration and the XOR key to decrypt the payload.

Further IoCs, Yara / Snort rules and a deeper report in PDF format are available by subscribing a Telsy Threat Intelligence service.

3. Indicators of compromise

3.1 URL-Download First Stage

hxxp://psm-ir.com/powers

hxxp://psm-ir.com/ghost

hxxp://psm-ir.com/gemni

3.2 First-Stage Payload

ef1aed43a38dc7d81d03642f875432befea53354

ff908cd8d0e895fa8a0649afbe08cdd101c3bd06

ee07dfda2c004d798e724f65b7e775c76078deb0

b8459e1f0a719c0adae628b0f1336837b3a5c9f5

ccaf562fb5e0a172bf3f9b78aae277893bb202fa

bd37c996b4da560ae97b2609458aa048826f5337

5083a08925984bc70e2e88a9c4f94ea2ecb53d7f

e8367e6f1668c29233eea8d5ca74b370c9f6b37a

16f6f9cd325650f42ae2fda3f0e05f1734e0f9bc

b471af6675d8d95a22525f21a4de5615836e41f7

3.3 Final-Payload

45afc0fd011e84df03100f100f040a5da2450b3f	AgentTesla
bd629941f27528e3a870e301411a7038f67fa061	AgentTesla
3cf0a486307eda000ab647258b78387d564b107f	AgentTesla
e98eaf017328b7f265fd818052870110e2756961	AgentTesla
5186e4c15e2a7f4bb1650b5e9e10471dcdb9956a	AgentTesla
f0ed14d49ee4ba8eff000ef57cb463d23a54939e	AgentTesla
7df5aae5f4150eb5575c7080c8f55a0576525200	AgentTesla
5c85317d3d2e67d282f64f0eaae62285ddd034b3	AgentTesla
cf94ba7dd59f73b8e08a95a32eb7d7d279c744c6	AgentTesla
4a0d3d121f261e91430892962b6aa7c0d70dd088	Remcos RAT

This report was produced by Telsy's "Cyber Threat Intelligence" team with the help of its CTI platform, which allows to analyze and stay updated on adversaries and threats that could impact customers' business.