Taking Advantage of PE Metadata, or How To Complete your Favorite Threat Actor's Sample Collection

Daniel Lunghi
daniel_lunghi@trendmicro.com

Trend Micro

Abstract. In this paper, we will show some common techniques that we leveraged in a real case investigation that started with one SysUpdate sample found in December 2020, and ended with dozens of samples from the same malware family, dating back to March 2015. SysUpdate is a malware family that has been attributed to the Iron Tiger threat actor in the past. Other malware families from the same threat actor were found, and the result of the investigation has been published in the Trend Micro blog [4]. The goal of the current paper is to discuss some of the techniques that proved useful to gather related samples, with detailed examples. It is complementary with the investigation presented at SSTIC in 2020 [9].

This investigation started when Talent Jump[1] company handed us a malware sample that they found in December 2020 in a gambling company. That gambling company had already been targeted in July 2019, in what we called Operation DRBControl [5]. Our goals were to find if the sample belonged to a known malware family, a known threat actor, and if it was related to the previous campaign.

1 Sample analysis

Four files were initially sent to us, and the code analysis showed that a fifth file was involved:

— `dlpumgr32.exe`, a legitimate signed file that is part of the DESlock+ product

— `DLPPREM32.DLL`, a malicious DLL sideloaded [1] by `dlpumgr32.exe` that loads and decodes `DLPPREM32.bin`

— `DLPPREM32.bin`, a shellcode that decompresses and loads a launcher in memory

— `data.res`, an encrypted file decoded by the launcher that contains two SysUpdate versions: one for a 32-bit architecture and another for a 64-bit architecture

1. `http://www.talent-jump.com/EN_index.html`

— `config.res`, an encrypted file decoded by the launcher which contains the SysUpdate command-and-control (C&C) address

The `config.res` file was not directly available to us, but we could confirm by analyzing a process dump that it contained the C&C IP address. The code analysis showed that the file is deleted right after being read.

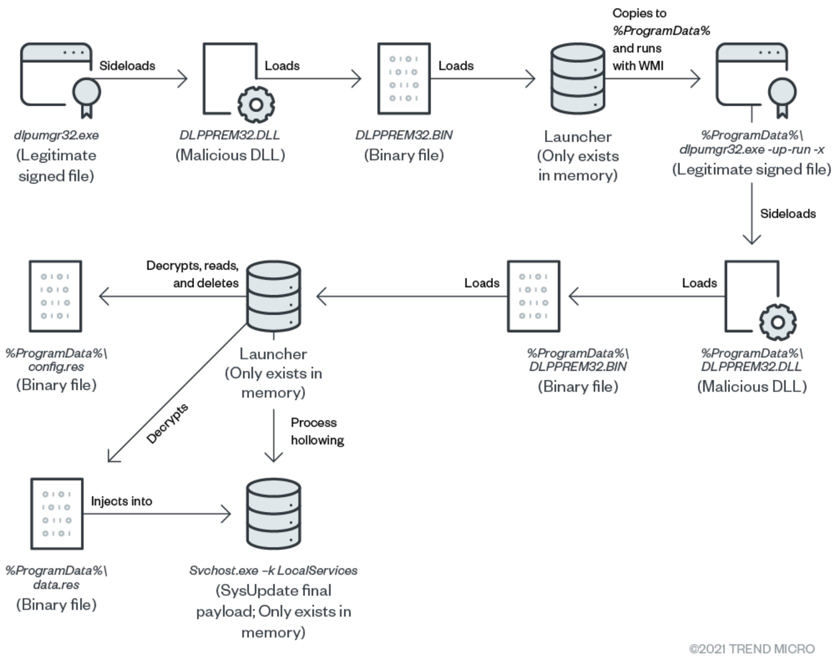
Sideloads
Loads
Loads
Copies to %ProgramData% and runs with WMI
dlpumgr32.exe (Legitimate signed file)
DLPPREM32.DLL (Malicious DLL)
DLPPREM32.BIN (Binary file)
Launcher (Only exists in memory)
%ProgramData%\ dlpumgr32.exe -up-run -x (Legitimate signed file)
Sideloads
Decrypts, reads, and deletes
Loads
Loads
%ProgramData%\ config.res (Binary file)
Launcher (Only exists in memory)
%ProgramData%\ DLPPREM32.BIN (Binary file)
%ProgramData%\ DLPPREM32.DLL (Malicious DLL)
Decrypts
Process hollowing
Injects into
%ProgramData%\ data.res (Binary file)
Svchost.exe -k LocalServices (SysUpdate final payload; Only exists in memory)
©2021 TREND MICRO

Fig. 1. SysUpdate sample loading process

A quick analysis of the unobfuscated code reveals some interesting RTTI class names, which help figuring some of the malware features:

— CMShell
— CMFile
— CMCapture
— CMServices
— CMProcess
— CMPipeServer
— CMPipeClient

2 Malware family identification

The process of obtaining unobfuscated code from the packed files is off-topic, but as seen on figure 1, the unobfuscated code is only present in memory. After dumping the unobfuscated code, our goal was to identify the malware family, if known. We found a hardcoded user-agent, `Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.116 Safari/537.36`, which was mentioned in an article written by Dell SecureWorks [17] in February 27, 2019 on the Bronze Union threat actor, also known as Emissary Panda, APT27, LuckyMouse or Iron Tiger [12]. The article mentions two malware families, HyperBro [2] and SysUpdate. We were familiar with HyperBro, as we encountered it during our investigation on Operation DRBControl [6] in 2019, so we could discard it. We found another article [14] listing the features of the SysUpdate malware family, and they matched the behavior of the in-memory launcher displayed in figure 1. Thanks to it, we confidently assumed that this malware belonged to the SysUpdate family.

3 Pivoting on metadata

The two previously listed articles [14, 17] provided indicators of compromise (IOC) of old SysUpdate samples that we could retrieve. A quick analysis showed that their loading process was quite similar, involving a legitimate signed file loading a malicious DLL, which unpacks a binary file in-memory. However, no launcher nor additional files were involved.

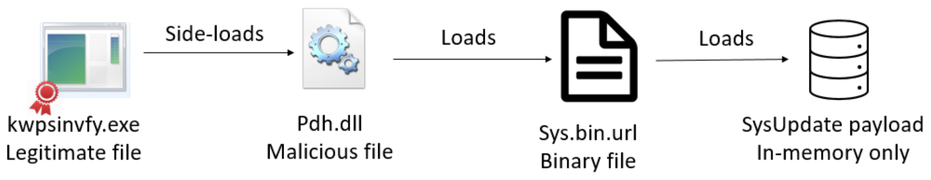

Fig. 2. Loading process of a SysUpdate sample found in NCC blog [14]

3.1 Filename

We noticed the `sys.bin.url` filename has been used in both cases. This was our first pivot, and searching for this filename followed by relevant

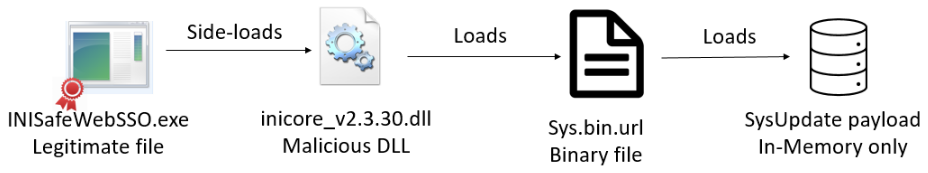

Fig. 3. Loading process of a SysUpdate sample found in Dell SecureWorks blog [17]

keywords (APT, malware) returned sandbox results, but no new samples. Issuing the same query [2] in the Virus Total platform returned 7 results, some of which were unknown to us. Searching for their MD5 hashes in search engines, we found another article [8] written in Farsi language which described the same malware, and contained further IOCs.

In that list, we noticed another legitimate binary being abused by the attacker for side-loading a malicious DLL: `PYTHON33.dll`. In that case, the DLL unpacked a binary file named `PYTHON33.hlp`. A new search [3] on this filename returned four samples and a new technical analysis [13] of our malware. We also found a security bulletin [3] from the UAE national CERT dated June 13, 2019 which mentioned a malware family named HyperSSL. After analysis, it turned out it was the same malware family, and thus HyperSSL could be considered an alias of SysUpdate.

In addition to pivoting on filenames for packed payloads, which can be changed by the attacker at any moment, we noticed that the threat actor tended to reuse the same legitimate executables to side-load its malicious DLL libraries. As the attacker did not control the code of such signed legitimate binaries, he could not change the names of the side-loaded libraries, and thus we could leverage that behavior to hunt for malicious DLL libraries based on their name. As an example, we could search for libraries named `PYTHON33.dll`, but we had to filter the results, because a simple query returned 166 results. We removed corrupt samples, but there were still 99 samples left. After analyzing malicious DLL files from the attacker, we noticed their size was always less than 100 Kb, whereas the official Python library was heavier than a megabyte. By adding a filter on the file size [4], we obtained only five files, which we could analyze manually. It turned out they were all related to the SysUpdate malware family.

2. `https://www.virustotal.com/gui/search/name:sys.bin.url/files`
3. `https://www.virustotal.com/gui/search/name:python33.hlp/files`
4. `https://www.virustotal.com/gui/search/name:PYTHON33.dllsize:100Kb-NOTtag:corrupt/files`

At this point, simple queries on filenames learned us that our threat actor has targeted Middle-East countries in the past (Iran, UAE), among which governmental entities, and we expanded our IOC list, not only with samples but also with infrastructure. We also saw multiple reports of malware families and TTPs related to our threat actor.

3.2 Imphash

"Imphash" or "Import hashing" is a method invented by FireEye and published [11] in 2014. The general idea is that the Import Address Table (IAT), which is built at compilation time, changes depending on which order the functions are placed in the source code. Thus, when a significant amount of functions are imported and called in a malware's source code, its IAT has a fingerprint unique enough to allow for correlations. Multiple tools [5] exist to generate this hash. In our case, this method worked pretty well on the malicious DLLs compiled by our threat actor. For example, a search [6] query on the imphash `509a3352028077367321fbf20f39f6d9` returned three files related to Iron Tiger. Other platforms, such as Malware Bazaar, allow for similar search queries [7] through their API.

It is also possible to build a Yara rule that matches on files with a specific imphash:

```
import "pe"
rule sysupdate_dll_imphash
{
  meta:
    author = "Daniel Lunghi"
    description = "Matches Iron Tiger's SysUpdate DLLs from 2019"
    purpose = "Show an example of imphash Yara rule for SSTIC 2021
        conference"
  condition:
    uint16(0) ==0x5a4d and // "MZ" header
    pe.imphash()=="509a3352028077367321fbf20f39f6d9"
}
```

3.3 PE RICH Header

The RICH header is added by the Microsoft compiler to files in the Portable Executable (PE) format. It has been first documented [15] in 2010. It embeds many information, such as the compiler's version, up to

5. `https://github.com/Neo23x0/ImpHash-Generator`
6. `https://www.virustotal.com/gui/search/imphash:509a3352028077367321fbf20f39f6d9/files`
7. `https://bazaar.abuse.ch/api/#imphash`

the build number. Researchers have taken advantage [10] of this header for the last couple of years, because it sometimes bring useful correlations. The general idea is that a similar building environment should produce a similar header, which could help finding binaries that have been compiled in the same machine. Some public tools[8] generate a MD5 hash of the RICH header, which can then be used in a Yara rule, or in malware repositories that support it. As an example, searching for the hash `5503d2d1e505a487cbc37b6ed423081f` in Virus Total returns three files, which are all related to our threat actor.

The following Yara rule matches those samples:

```
import "pe"
import "hash"
rule sysupdate_richheader
{
  meta:
    author = "Daniel Lunghi"
    description = "Matches Iron Tiger's SysUpdate DLLs from 2018"
    purpose = "Show an example of RICH header Yara rule for SSTIC
        2021 conference"
  condition:
    uint16(0) ==0x5a4d and // "MZ" header
    hash.md5(pe.rich_signature.clear_data)=="5503
        d2d1e505a487cbc37b6ed423081f"
}
```

However, it is important to note that this header is ignored when running an executable, and thus, it can be altered without any impact on the code execution. In 2018, it has been proven [7] that a threat actor purposefully modified the RICH header of an executable to match the header of another threat actor.

3.4 Stolen certificates

PE executables can be signed using the Microsoft Authenticode technology. The purpose is to identify the publisher of the code and guarantee the code integrity. Authenticode relies on x509 certificates and certification authorities. Some threat actors compromise software companies to steal the private key that allows them to generate a valid signature for their own malicious executables. This might confuse some security solutions or young analysts that might flag signed code as non malicious. From a defender stand point, it is useful to check for compromised certificates, as usually a threat actor will sign more than one executable with a stolen certificate. In the case of Iron Tiger, we found that DLL `inicore_v2.3.30.dll` reported

8. `https://gist.github.com/fr0gger/44ef948d5f129a183b4d44d3e867e097`

by Palo Alto [16] was signed by a certificate stolen from the company Kepware Technologies. Searching for its serial number [9] or thumbprint [10] in Virus Total returns six results, which are all related to the SysUpdate malware family. Malware Bazaar provides two keywords related to certificates, serial_number and issuer_cn, which are self-explanatory.

The following Yara rule matches samples signed by the stolen certificate:

```
import "pe"
rule sysupdate_compromised_kepware_certificate
{
  meta:
    author = "Daniel Lunghi"
    description = "Matches Iron Tiger's files from 2018 signed by a
        stolen certificate from Kepware Technologies"
    purpose = "Show an example of a Yara rule handling certificates
        for SSTIC 2021 conference"
  condition:
    uint16(0) ==0x5a4d and // "MZ" header
    for any i in (0 .. pe.number_of_signatures) : (
      // thumbprint and serial must be lower case
      pe.signatures[i].serial == "0d:02:f1:24:c1:64:96:22:57:06:f4:
          ed:d3:8d:a6:96"
      or pe.signatures[i].thumbprint == "
          fa89c0cbfce8d745ef3b2f72312077799e69df72"
    )
}
```

Note that the stolen certificate has been revoked by the certification authority, so the malicious files' signature is not valid anymore.

3.5 TLSH

Multiple "fuzzy hashing" algorithms exist that intend to match similar files automatically. They usually split the original file in blocks of variable length, and then make a hash of the different blocks. The most popular fuzzy hashing algorithm is SSDeep. However, results on compiled code are usually not good. Of all the fuzzy hashing algorithms that we looked at, TLSH was the one that gave better results for correlation. Results should still be taken with caution. As an example, TLSH hash T112F21A0172A28477E1AE2A3424B592725D7F7C416AF040CB3F9916FA9FB16D0DA3C367 returned 223 results in Virus Total, and while many of them were related to our threat actor, many of them were not, and simply

9. https://www.virustotal.com/gui/search/signature:0D02F124C16496225706F4EDD38DA696/file

10. https://www.virustotal.com/gui/search/signature:FA89C0CBFCE8D745EF3B2F72312077799E69DF72/files

had a similar structure. On the other hand, searching for the hash `T17A634B327C97D8B7E1D97AB858A2DA12152F250059F588C9BF7043E70F2A6509E37F0E` returned only two results, and both were related to our threat actor. Note that Malware Bazaar also allows to search for TLSH hash in its API[11].

4 Conclusion

The goal of this short paper was to present some techniques that defenders can leverage when investigating a breach to gather additional information and IOC about the threat actor. It is complementary with the talk [9] presented at SSTIC in 2020, this time focusing on PE metadata.

By using these techniques in a recent investigation, we started from a single unknown sample and found more than thirty samples from the same malware family, around 15 C&C IP addresses, multiple reports discussing the same threat actor and detailing its targets and Tactics, Techniques and Procedures (TTPs). It shows the importance of public research containing IOC, which helped us to identify the malware family. We could also compare our sample to previous versions of the same malware and spot the structural changes.

We hope that by showing examples taken from a real case investigation, it will help young researchers to apply these techniques to their own investigations.

References

1. Mitre ATT&CK. DLL Side-Loading. `https://attack.mitre.org/techniques/T1073/`.
2. Mitre ATT&CK. HyperBro. `https://attack.mitre.org/software/S0398/`.
3. CERT.ae. ADV-19-27-Advanced Notification of Cyber Threats against Family of Malware Giving Remote Access to Computers. `https://www.tra.gov.ae/assets/mTP39Tp6.pdf.aspx`, 2019.
4. K. Lu D. Lunghi. Iron Tiger APT Updates Toolkit With Evolved SysUpdate Malware. `https://www.trendmicro.com/en_us/research/21/d/iron-tiger-apt-updates-toolkit-with-evolved-sysupdate-malware-va.html`, 2021.
5. K. Lu J. Yaneza D. Lunghi, C. Pernet. Uncovering a Cyberespionage Campaign Targeting Gambling Companies in Southeast Asia. `https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/operation-drbcontrol-uncovering-a-cyberespionage-campaign-targeting-gambling-companies-in-southeast-asia`, 2020.

11. `https://bazaar.abuse.ch/api/#tlsh`

6. K. Lu J. Yaneza D. Lunghi, C. Pernet. Uncovering DRBControl - Inside the Cyberespionage Campaign Targeting Gambling Operations. https://documents.trendmicro.com/assets/white_papers/wp-uncovering-DRBcontrol.pdf, 2020.
7. GReAT. The devil's in the Rich header. https://securelist.com/the-devils-in-the-rich-header/84348/, 2018.
8. Kamiran. APT27 HackerTeam Analyse. https://www.kamiran.asia/documents/APT27_HackerTeam_Analyse.pdf, 2019.
9. D. Lunghi. Pivoter tel Bernard, ou comment monitorer des attaquants négligents. https://www.sstic.org/2020/presentation/pivoter_tel_bernard_ou_comment_monitorer_des_attaquants_ngligents/, 2020.
10. P. Kálnai M. Poslušný. Rich Headers: leveraging this mysterious artifact of the PE format. https://www.virusbulletin.com/virusbulletin/2020/01/vb2019-paper-rich-headers-leveraging-mysterious-artifact-pe-format/, 2019.
11. Mandiant. Tracking Malware with Import Hashing. https://www.fireeye.com/blog/threat-research/2014/01/tracking-malware-import-hashing.html, 2014.
12. Mitre. Threat Group-3390. https://attack.mitre.org/groups/G0027/, 2017.
13. Norfolk. Emissary Panda DLL Backdoor. https://norfolkinfosec.com/emissary-panda-dll-backdoor/, 2019.
14. N. Pantazopoulos. Emissary Panda – A potential new malicious tool. https://research.nccgroup.com/2018/05/18/emissary-panda-a-potential-new-malicious-tool/, 2018.
15. D. Pistelli. Microsoft's Rich Signature (undocumented). https://www.ntcore.com/files/richsign.htm, 2010.
16. T. Lancaster R. Falcone. Emissary Panda Attacks Middle East Government SharePoint Servers. https://unit42.paloaltonetworks.com/emissary-panda-attacks-middle-east-government-sharepoint-servers/, 2019.
17. Counter Threat Unit™ (CTU) researcher team. A Peek into BRONZE UNION's Toolbox. https://www.secureworks.com/research/a-peek-into-bronze-unions-toolbox, 2019.