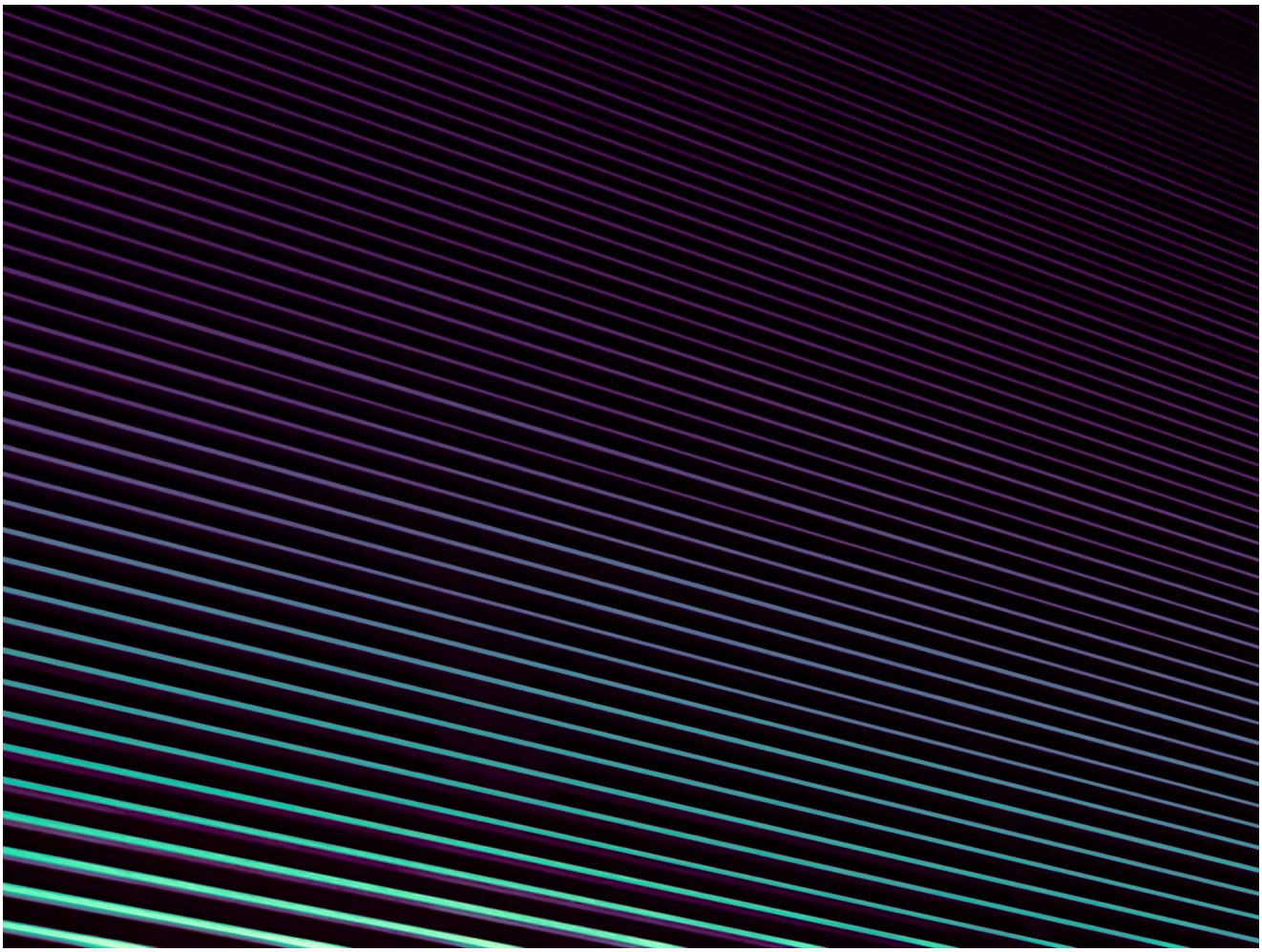
Smokeloader Analysis and More Family Detections

hatching.io/blog/tt-2020-08-27/



2020-08-27

triage

Written by
Pete Cowman

In this week's Triage Thursday blog, we'll cover a number of minor updates to family classification introduced in the past week, and @Casperinous goes under-the-hood with recent changes observed in SmokeLoader samples.

Over the past few days we have released another batch of smaller detection updates, affecting several families. The main focus has been on ransomware and stealers, adding family-specific detection for samples recently seen in the wild.

Read on below for more information on each of these topics.

Not signed up yet? Head over to https://tria.ge/ and register right away!

SmokeLoader Analysis

Smokeloader is a downloader/backdoor which has been active since 2011. Over the years it has evolved both its capabilities and the variety of malware it downloads to the infected host. In this post we will have a look at what's changed since the most recent analysis by Checkpoint and present the new features introduced in 2020.

Smokeloader Analyses:

- 200827-m1jren2nas
- 200827-6x7fdlj8y2
- 200827-v6tcrvw9es

New Anti-VM methods

Detection of unsigned drivers

Smokeloader introduced 2 new anti-VM checks closely associated with the gaming community.

The first one checks if the executable's path contains the string `[A-F0-9]{4}.vmt`. Also, if the architecure of the system is 64-bit, `NtQuerySystemInformation` is called with the first argument set to `0x67` (`SystemCodeIntegrityInformation`). After the call, `ESI` points to the `SYSTEM_CODEINTEGRITY_INFORMATION`. The check `[ESI+4]` confirms if the struct's `CodeIntegrityOptions` member is equal to `0x2`. Based on some public information it is assumed that this check is intended to detect the Driver Signing Policy of the infected host - if the value is indeed equal with `0x2` an unsigned kernel driver can be installed, a common configuration for sandboxes.

The check is not well implemented - instead of comparing if the variable is equal with `0x2`, it should be using a `TEST` instruction to figure out if the `0x2` flag is used.

```
0040200C    C745 FC 01000000    mov dword ptr ss:[ebp-4],1
00402013    8B5D 08             mov ebx,dword ptr ss:[ebp+8]
00402016    31FF                xor edi,edi
00402018    66:8CE8             mov ax,gs
0040201B    66:85C0             test ax,ax
0040201E    74 27               je smokey_loader.402047
00402020    8D75 F0             lea esi,dword ptr ss:[ebp-10]
00402023    C706 08000000       mov dword ptr ds:[esi],8
00402029    57                  push edi
0040202A    6A 08               push 8
0040202C    56                  push esi
0040202D    6A 67               push 67
0040202F    FF93 88000000       call dword ptr ds:[ebx+88]
00402035    85C0                test eax,eax
00402037    75 0E               jne smokey_loader.402047
00402039    8B46 04             mov eax,dword ptr ds:[esi+4]
0040203C    85C0                test eax,eax
0040203E    74 05               je smokey_loader.402045
00402040    83F8 02             cmp eax,2
00402043    75 02               jne smokey_loader.402047
00402045    EB 21               jmp smokey_loader.402068
00402047    8D75 F8             lea esi,dword ptr ss:[ebp-8]
0040204A    893E                mov dword ptr ds:[esi],edi
0040204C    57                  push edi
0040204D    6A 04               push 4
0040204F    56                  push esi
00402050    6A 07               push 7
00402052    6A FF               push FFFFFFFF
00402054    FF93 8C000000       call dword ptr ds:[ebx+8C]
0040205A    85C0                test eax,eax
0040205C    75 08               jne smokey_loader.402066
```

Detection of loaded DLLs

Smokeloader also extended the list of loaded DLLs that it checks for. Going by previous analyses Smokeloader was only checking for sbiedll, but it was observed that in 2020 it is also looking for:

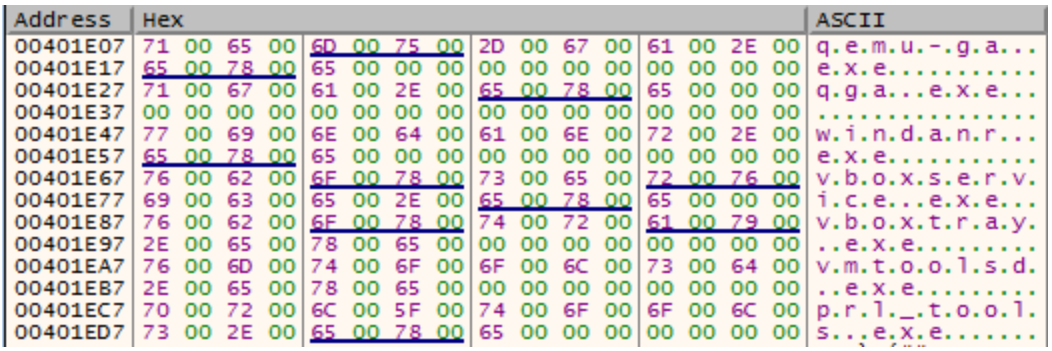
- aswhook
- snxhw

```
Address   Hex                                              ASCII
004020EB  73 62 69 65 64 6C 6C 00 61 73 77 68 6F 6F 6B 00  sbiedll.aswhook.
004020FB  73 6E 78 68 6B 00 00 00 00 5E 80 3E 00 74 11 56  snxhk....^.>.t.V
```

Detection of processes associated with virtualization software

Something that is common in various packers/loaders is checking the running processes against an array of predefined strings, in order to check virtualized environments. Smokeloader has implemented the same check, by calling `NtQuerySystemInformation` with the first parameter set to `0x5` (`SystemProcessInformation`) in order to get all the running processes. Then there is a loop where every process is converted to lowercase and is checked with `wcsstr` to see if it contains the following strings:

- L"qemu-ga.exe”
- L"qga.exe”
- L"windanr.exe”
- L"vboxservice.exe”
- L"vboxtray.exe”
- L"vmtoolsd.exe”
- L"prl_tools.exe”

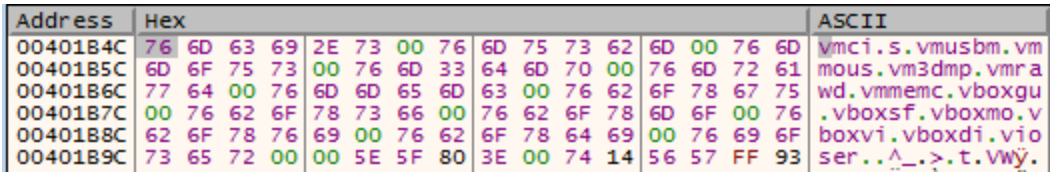


Address	Hex	ASCII
00401E07	71 00 65 00 6D 00 75 00 2D 00 67 00 61 00 2E 00	q.e.m.u.-.g.a...
00401E17	65 00 78 00 65 00 00 00 00 00 00 00 00 00 00 00	e.x.e...........
00401E27	71 00 67 00 61 00 2E 00 65 00 78 00 65 00 00 00	q.g.a...e.x.e...
00401E37	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
00401E47	77 00 69 00 6E 00 64 00 61 00 6E 00 72 00 2E 00	w.i.n.d.a.n.r...
00401E57	65 00 78 00 65 00 00 00 00 00 00 00 00 00 00 00	e.x.e...........
00401E67	76 00 62 00 6F 00 78 00 73 00 65 00 72 00 76 00	v.b.o.x.s.e.r.v.
00401E77	69 00 63 00 65 00 2E 00 65 00 78 00 65 00 00 00	i.c.e...e.x.e...
00401E87	76 00 62 00 6F 00 78 00 74 00 72 00 61 00 79 00	v.b.o.x.t.r.a.y.
00401E97	2E 00 65 00 78 00 65 00 00 00 00 00 00 00 00 00	..e.x.e.........
00401EA7	76 00 6D 00 74 00 6F 00 6F 00 6C 00 73 00 64 00	v.m.t.o.o.l.s.d.
00401EB7	2E 00 65 00 78 00 65 00 00 00 00 00 00 00 00 00	..e.x.e.........
00401EC7	70 00 72 00 6C 00 5F 00 74 00 6F 00 6F 00 6C 00	p.r.l._.t.o.o.l.
00401ED7	73 00 2E 00 65 00 78 00 65 00 00 00 00 00 00 00	s...e.x.e.......

Detection of files associated with virtualization software

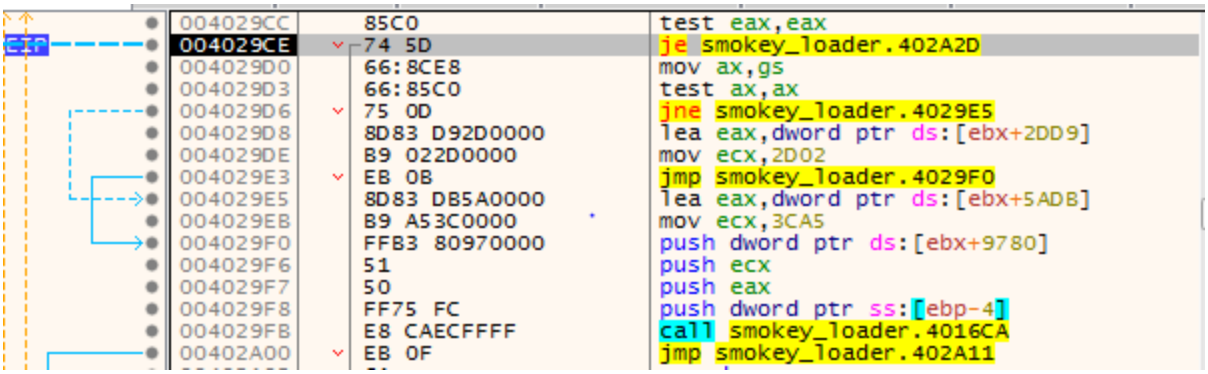
Another technique employed by Smokeloader is checking the `System32` folder for files that are associated with virtualization software. This is again done by calling `NtQuerySystemInformation` with the first argument `0xB` (`SystemModuleInformation`). Then, following the previous logic, there is a loop where every file in the aforementioned location is converted to lowercase and checked by calling `strstr` if it contains the following strings:

- “vmci.s”
- “vmusbm”
- “vmmous”
- “vm3dmp”
- “vmrawd”
- “vmmemc”
- “vboxgu”
- “vboxsf”
- “vboxmo”
- “vboxvi”
- “vboxdi”
- “vioser”



Address	Hex	ASCII
00401B4C	76 6D 63 69 2E 73 00 76 6D 75 73 62 6D 00 76 6D	vmci.s.vmusbm.vm
00401B5C	6D 6F 75 73 00 76 6D 33 64 6D 70 00 76 6D 72 61	mous.vm3dmp.vmra
00401B6C	77 64 00 76 6D 6D 65 6D 63 00 76 62 6F 78 67 75	wd.vmmemc.vboxgu
00401B7C	00 76 62 6F 78 73 66 00 76 62 6F 78 6D 6F 00 76	.vboxsf.vboxmo.v
00401B8C	62 6F 78 76 69 00 76 62 6F 78 64 69 00 76 69 6F	boxvi.vboxdi.vio
00401B9C	73 65 72 00 00 5E 5F 80 3E 00 74 14 56 57 FF 93	ser..^_.>.t.VWÿ.

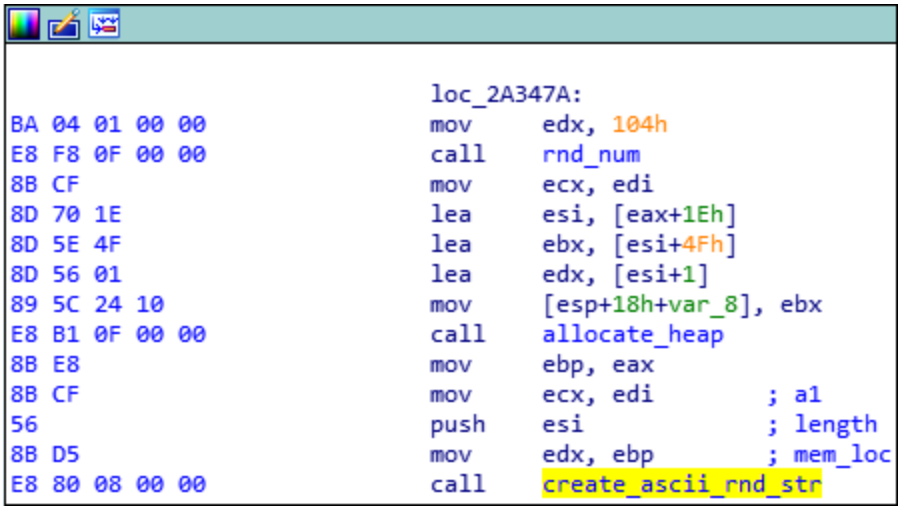
After successfully passing the aforementioned checks, Smokeloader must determine the system’s architecture. This is done by using the `gs` register and a test instruction. For our own convenience, we patched the check in order for Smokeloader to decompress the 32-bit payload and continue the analysis. While it was common for Smokeloader to utilize Propagate to inject the payload in `explorer.exe` , in the 2020 version it is still injecting into this process but it using a more typical combination of `NtCreateSection`, `NtMapViewOfSection` and `RtlCreateUserThread` to start the execution.



Changes in the payload

Increased size of random data buffer

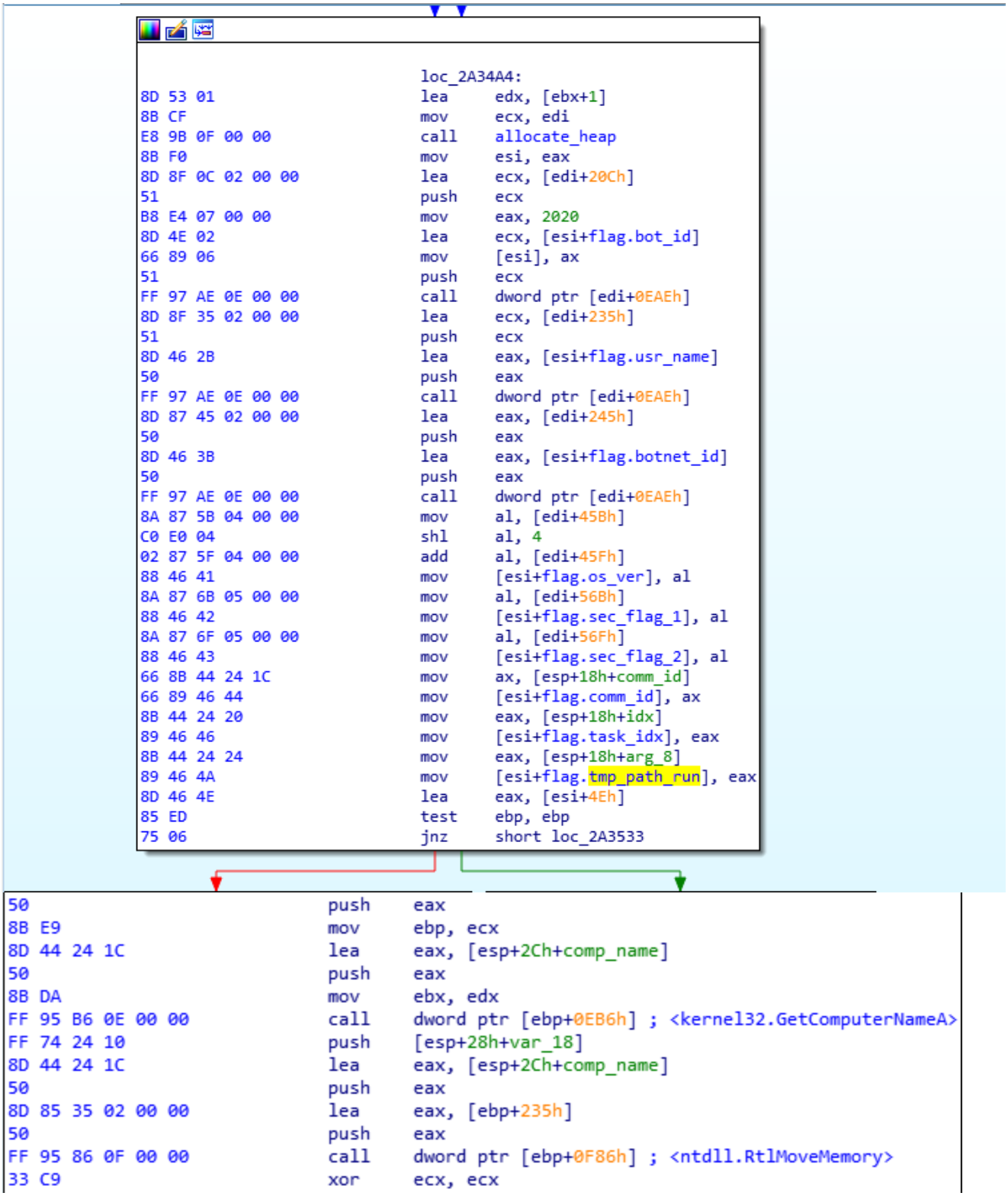
Smokeloader introduced the usage of randomly generated data in 2019, possibly in order to fool IDS/IPS systems. The size of the buffer is calculated randomly but is set to be at most `0x104`. Then, the number is used to allocate heap space and fill it with randomly generated lowercase letters. The generated string is appended at the end of the packet structure.



Change in communication traffic

As was discovered in early March, the communication packet structure of Smokeloader has been extended by `0x10` bytes. In the new struct, after the `bot_id` member, there is a new field allocated to hold the name of the infected host. There is also now a check to either append the random data or the additional data at the end of the `pkc` struct. The new struct is now defined like this:

```
struct pkc {
     WORD magic
     BYTE[40] bot_id
     BYTE[16] comp_name
     BYTE[6] botnet_id
     BYTE os_ver
     BYTE sec_flag_1
     BYTE sec_flag_2
     WORD comm_id
     DWORD task_idx
     DWORD tmp_path_run
     BYTE[n] extra_data
}
```



In some cases SmokeLoader was observed to be using decoy C2 to put off analysts. In these instances the sample stored a fake value using its standard encryption technique which would be dumped by static extractors, and the actual C2 was simply stored as a plaintext string. Triage can now distinguish between the fake and real C2 strings and only reports the legitimate ones in the report. This analysis is a good example of this behaviour.

Ransomware Support

Ransomware is extremely active these days and new variants and families are constantly being released, with even relatively basic ones sometimes managing to achieve infections in the wild. This week we've added support for a number of these which have gained attention over recent weeks.

LockBit and BigLock Analysis:

> 200827-dmry7lp4cs

The sample referenced above came to our attention recently as a slightly unusual case. It drops multiple families, including 2 different ransomware - Lockbit and BigLock. Lockbit is run first, encrypting files with it's distinctive `.lockbit` extension, then another re-encrypts the files with a second layer.

For Lockbit, ransom note extraction has been improved to now also dump details like Telegram contacts, and we have fixed an issue that was preventing some URLs being dumped from certain variants of the note.

We have also added support for BigLock, a family we previously did not have family classification for. The note and family tag should now be correctly displayed in the report.

Along with this, we have improved/added detection and ransom note support for:

- DarkSide ransomware
- Conti ransomware
  - 200826-jdzf5d33aa
  - 200826-k8ykljftvn
- JackPot Ransomware

  > 200826-3jfzxsp9yx
- DeathRansom

  > 200803-bktwtzlfze

Infostealers

We have added a number of yara rules and other detections for a few infostealer families. Where possible we have also used behaviour to identify them, but often one infostealer's actions look much like another, so our focus has generally been on static techniques.

404Keylogger

Infostealer which has been exploiting COVID-19 related lures to gain infections. First appeared around August 2019.

Analyses:

- 200818-t1jk5m8sc6
- 200624-gbxe29kehe

Kutaki

Keylogger with some other basic infostealer functionality like taking screenshots and harvesting data on the clipboard. Includes a range of anti-VM and anti-analysis techniques, although mostly a bit dated.

Analyses:

- 200805-k11vh8yarj
- 200805-arnebas9fa

XpertRAT

Backdoor/stealer which can carry out a wide range of operations on an infected machine depending on the instructions received. Can also act as a dropper for other families.

Analyses:

- 200624-3pqyjfy64j
- 200817-h4pjdtget2