Graphology of an Exploit

Hunting for exploits by looking for the author's fingerprints

CHECK POINT RESEARCH

Who are we?

ITAY COHEN
Malware Researcher
Co-Maintainer of Radare2 & Cutter
@megabeets_

EYAL ITKIN
Vulnerability Researcher
@EyalItkin

CHECK POINT RESEARCH

It all began with an incident response case

Tales of a mysterious binary

During a complicated attack we found a mysterious 64-bit binary:

1. The binary was **very small**
2. Unusual debug strings suggested an attempt to **exploit** a vulnerability
3. Leftover **PDB path**

S:\Work\Inject**cve-2019-0859**\x64\Release\CmdTest.pdb

A quick look at CVE-2019-0859

Reverse-engineering the exploit was pretty straight forward -

A **Use-After-Free** vulnerability in CreateWindowEx. Used to Elevate Privileges

```
mov     edx, 0x8002                ; LPCWSTR lpClassName
xor     ecx, ecx                   ; DWORD dwExStyle
call    qword [CreateWindowExW]    ; HWND CreateWindowExW(DWORD dwExStyle, LPCWSTR lpClassName, L...
test    rax, rax
jne     0x1400020b6

mov     dword [0x1400063dc], ebx
mov     r8, rsi                    ; LONG_PTR dwNewLong
lea     edx, [rax - 8]
mov     rcx, qword [0x1400064f8]   ; HWND hWnd
call    qword [SetClassLongPtrW]   ; ULONG_PTR SetClassLongPtrW(HWND hWnd, int nIndex, LONG_PTR d...
xor     r9d, r9d                   ; LPARAM lParam
mov     r8, r14                    ; WPARAM wParam
mov     edx, msg.MN_SETHMENU       ; UINT Msg
mov     rcx, r15                   ; HWND hWnd
call    qword [SendMessageW]       ; LRESULT SendMessageW(HWND hWnd, UINT Msg, WPARAM wParam, LPA...
```

Script Kiddie?

~~Scrip~~ NO ~~ddie?~~

We couldn't find any **public resource** of this implementation

It wasn’t written by the attacker!

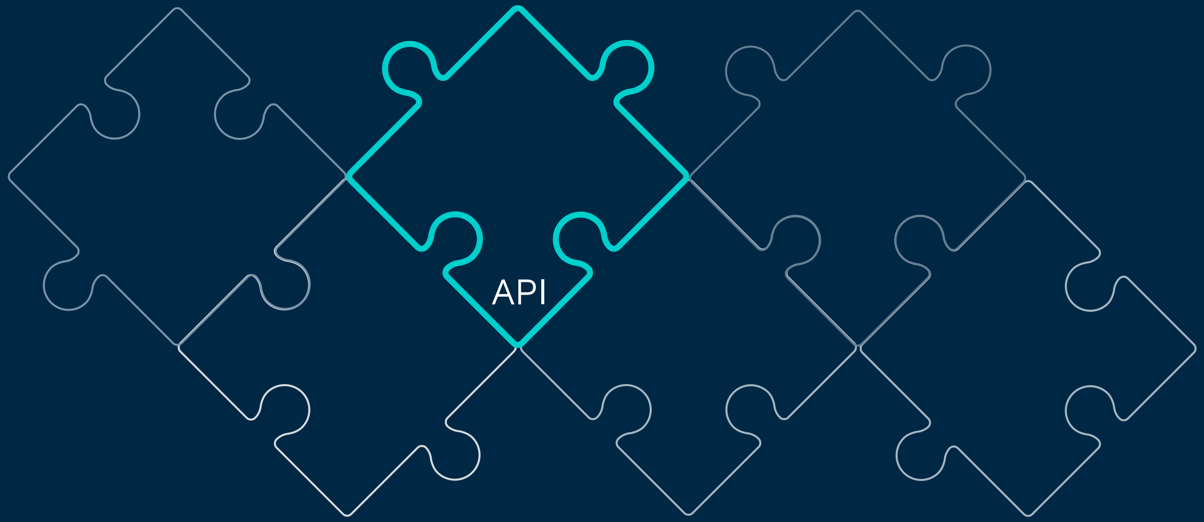
The exploit and the malware weren’t written by the same authors:

- Different code quality
- Lack of obfuscation
- Timestamps
- PDB paths

Exploit Distribution 101

Exploit distribution

The exploit is only a single piece of the puzzle



Acquiring exploits

Another team in the same organization

Another organization

Offensive Cyber companies

Exploit brokers

Underground forums

Publicly available exploits (Github, Metasploit)

Thinking like an exploit writer

Thinking like an exploit writer

An exploit is a **product** and not some PoC on Github.

It needs to support as many versions as possible:

1. 32-bit / 64-bit
2. Windows XP, Vista, 7, 8.0, 8.1, 10

Often we will need direct access to a given syscall:

- syscall gate (assembly)
- syscall numbers

A lot of the code is actually exploit agnostic, and can be reused!

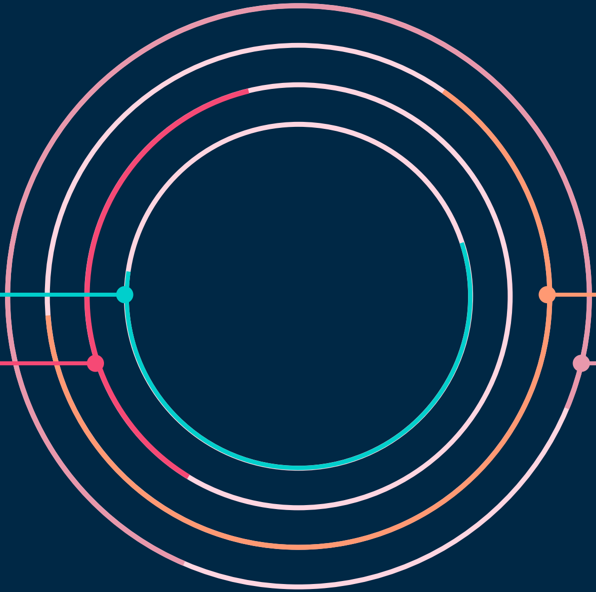
What are we looking for?

Unique Artifacts

- Hard coded values
- Data tables
- Strings
- PDB paths

Techniques & Habits

- Leaking
- Elevation
- Heap Spraying



Code Snippets

- Syscall wrappers
- Inline assembly
- Crypto

Framework

- Configurations
- Code structure
- Exploit flow
- API

Looking for clues

We have our 64-bit sample, let’s search for artifacts in it

Found some candidate, and did a basic search - a shot in the dark

- **Surprise:** we found the matching 32-bit sample :)

Looks promising, let’s start an extensive hunt with this rule

- Meanwhile, kept looking for more artifacts we could use

One day later, after we saw the results, we couldn’t believe what we found

949 Samples

(just from the initial hunt)

Identifying the author

Identifying the vulnerabilities

Identifying the vulnerabilities used in each exploit was a tedious task:

- Exploited as **0-Days** - Usually well documented in security reports
- Exploited as **1-Days** - Mostly nothing. Just good old RE and patch testing
- Sometimes we get lucky to have CVE-IDs in strings / PDBs

Some vulnerabilities were mislabeled by the author / clients :(

- CVE-2016-0165*

Some were exploited just from a patch-diff, without a clear CVE-ID

- CVE-2018-8641

The exploit writer

Volodimir (Volodya), a.k.a BuggiCorp

Developing exploits since 2015

Known clients include:

- Turla
- FIN8
- GandCrab

Exploits both 1-Days and 0-Days

Note: We focused on Windows local privilege escalations (LPEs)

CVE-2015-2546

CVE-2016-0040

CVE-2016-0165*

CVE-2017-0001

CVE-2018-8641

CVE-2019-1458

CVE-2016-0167

CVE-2016-7255

CVE-2017-0263

CVE-2019-0859

CVE-2019-1132

Identifying the fingerprints

We can’t pick an arbitrary code line and decide it is an “artifact”

- We need a control group to compare against

Our goal is to show that each exploit writer is unique:

- Had multiple implementation / exploitation decisions to make
- In each decision indeed faced multiple options
- Was consistent once chose a given decision

In order to do that, we reiterated our research method on REvil

- Embeds a 1-Day exploit for CVE-2018-8453

And once again, it worked!

Our control group

PlayBit, a.k.a luxor 2008

Developing exploits since 2013

Known clients include:

- REvil
- LockCrypt

Only exploits 1-Days

CVE-2013-3660

CVE-2015-0057

CVE-2015-1701

CVE-2016-7255

CVE-2018-8453

The author's fingerprints

Clue #1 – Sleep()

Yup, most* exploits start with a call to Sleep(200)

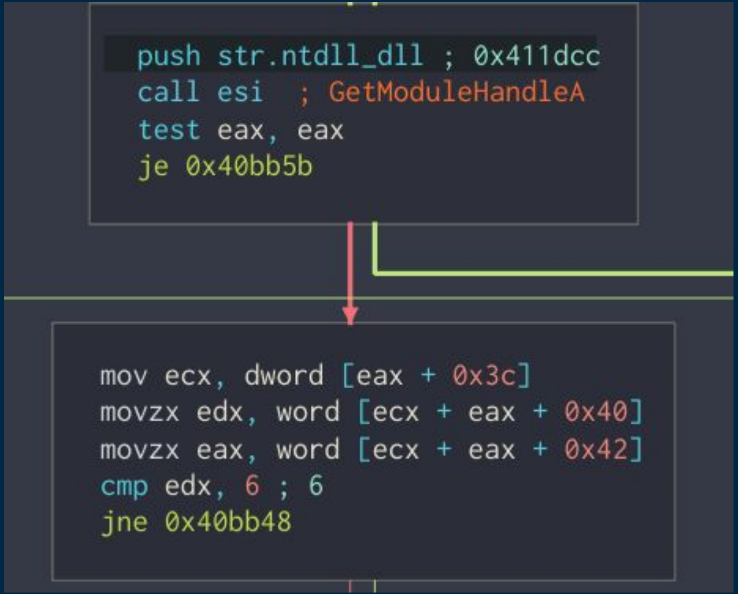
```
1077: exploit_bootstrap ();
sub     rsp, 0x68
mov     ecx, 200                          ; DWORD dwMilliseconds
call    qword [Sleep]                     ; VOID Sleep(DWORD dwMilliseconds)
xor     ecx, ecx                          ; LPCSTR lpModuleName
call    qword [GetModuleHandleA]          ; HMODULE GetModuleHandleA(LPCSTR lpModuleName)
lea     rcx, [0x1400065a0]                ; LPCRITICAL_SECTION lpCriticalSection
mov     qword [0x140006020], rax
call    qword [InitializeCriticalSection] ; VOID InitializeCriticalSection(LPCRITICAL_SECTION lpC...
lea     rcx, [0x140005268]                ; LPCSTR lpLibFileName
call    qword [LoadLibraryA]              ; HMODULE LoadLibraryA(LPCSTR lpLibFileName)
test    rax, rax
je      0x14000466e
```

We are not sure why is it there, but it is a distinct feature.

Clue #2 - OS Fingerprinting

Goal: Get the OS Major & Minor version numbers

The favorite method is directly parsing ntdll.dll's IMAGE_NT_HEADERS



Clue #3 - Token Swap

In order to elevate the target process (by PID) we need SYSTEM's token

The favorite method is scanning the pslist:

- Using arbitrary-read and arbitrary-write from user-mode
- Traversing the process list in search of both EPROCESS structs
- Updating target's EPROCESS to point at SYSTEM's token

However, this update requires delicate ref-count handling

Clue #3 - Token Swap

1. The token is an EX_FAST_REF object (lower ptr bits used as refcount)
2. There is an OBJECT_HEADER before the token, holding another refcount

On 32-bits, we found the following bug (On 64-bits it is calculated OK):

```
mov eax, dword [global_token_offset]
add eax, ebx
push eax
call arbitrary_read
mov esi, eax
mov ecx, esi
and ecx, 0xfffffff8
sub ecx, 0x18
push ecx
call arbitrary_read
add eax, 2
push eax
mov eax, esi
and eax, 0xfffffff0
sub eax, 0x18
push eax
call arbitrary_write
```

Evolution: Volodya’s learning curve

Worth mentioning

It is clear that Volodya was already quite **professional** from the first exploit -

CVE-2015-2546

From source code to compiled binaries

At start, Volodya used to sell the source-code of the exploits to the customers

1. Exploit was properly embedded in the binary
2. Source-level obfuscation was applied to both malware and the exploit
3. Elevation of current PID

Later, Volodya started to sell compiled exploits

1. The exploits are shown as separated binaries (or embedded PE)
2. They contain hard-coded instructions for the customers
3. Elevation of parent PID

Improvements in the exploits

1. More effective Arbitrary Read/Write primitives
   - Even a bug fix between CVE-2015-2546 and CVE-2016-0165*

2. Code modularity
   - Splitting large functions to modular sub-routines

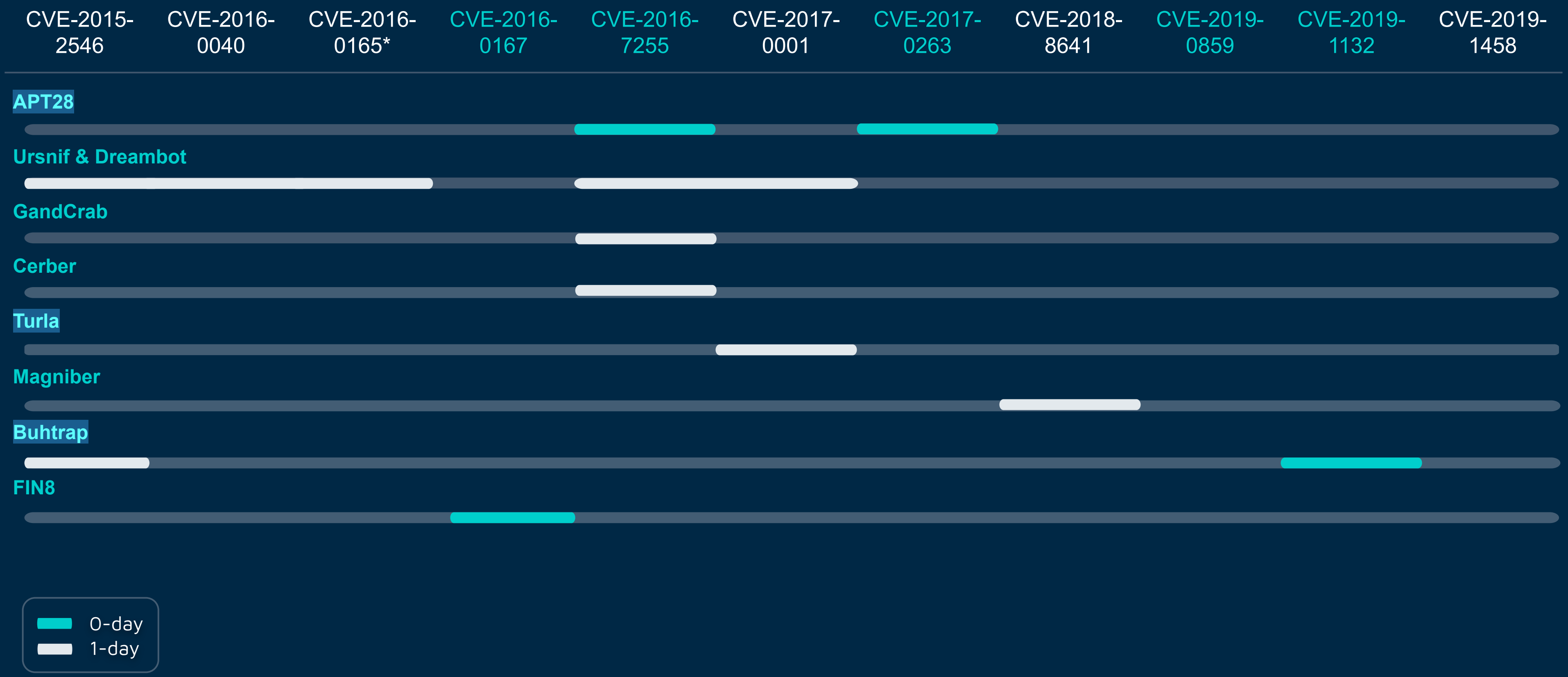
3. Dynamic search for the precise field offsets in various structs

4. Shift to distinguish between multiple Windows 10 versions

5. Exploits became more sophisticated

The Customers

The Customers



Conclusion

Research Methodology Worked

Fingerprinting an exploit writer and using these characteristics as unique hunting signatures.

Worked for both Volodya and PlayBit

16 Windows LPE Exploits

By two different developers between 2015-2019

A **significant** share of the exploitation market, specifically for **Windows LPE** exploits.

SURVIVORSHIP BIAS

How many more are out there?

Crimeware and APT

The customers were both **Crimeware** (especially Ransomware) and **nation-sponsored** groups.

You should try it too

THANK YOU

@megabeets_ @EyalItkin

cp<r>

CHECK POINT RESEARCH