Acceleration of Statistical Detection of Zero-Day Malware in the Memory Dump Using CUDA-Enabled GPU Hardware

Igor Korkin, Ph.D.   Iwan Nesterow

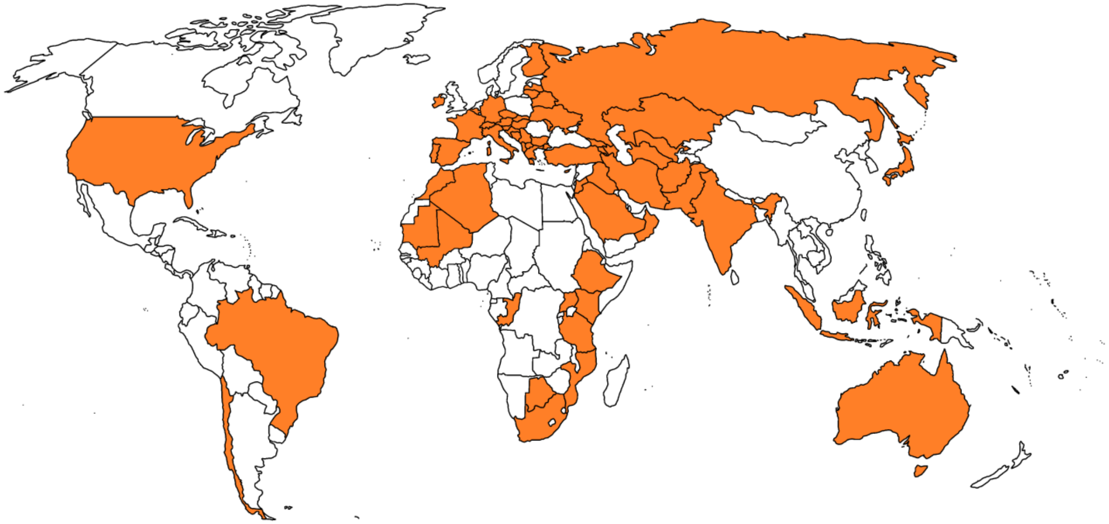
Independent Researchers

2016

Agenda

1. Motivation
2. Analysis of drawbacks of drivers detection
3. HighStem prototype
4. Drivers detection in the memory by separating code from data
5. GPU & CPU powered dump analysis

5-year cyber espionage attack



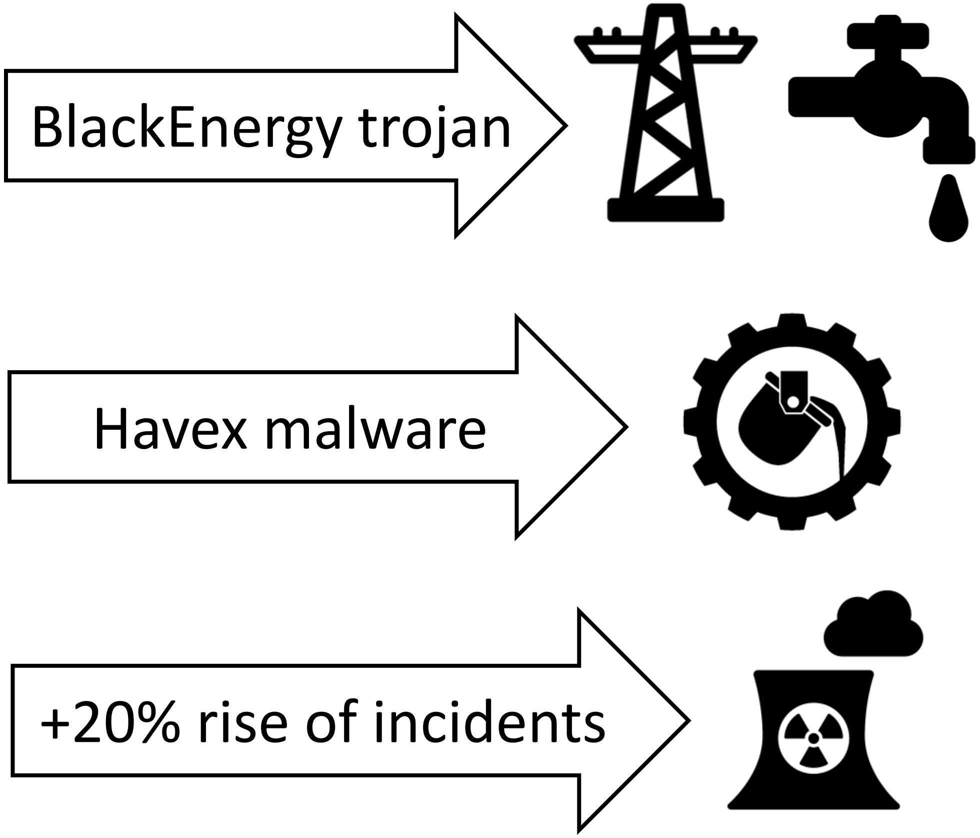
69 countries were attacked

🏛 🔬 🛢 🔥 ✈

Sensitive data were collected from hundreds of victims

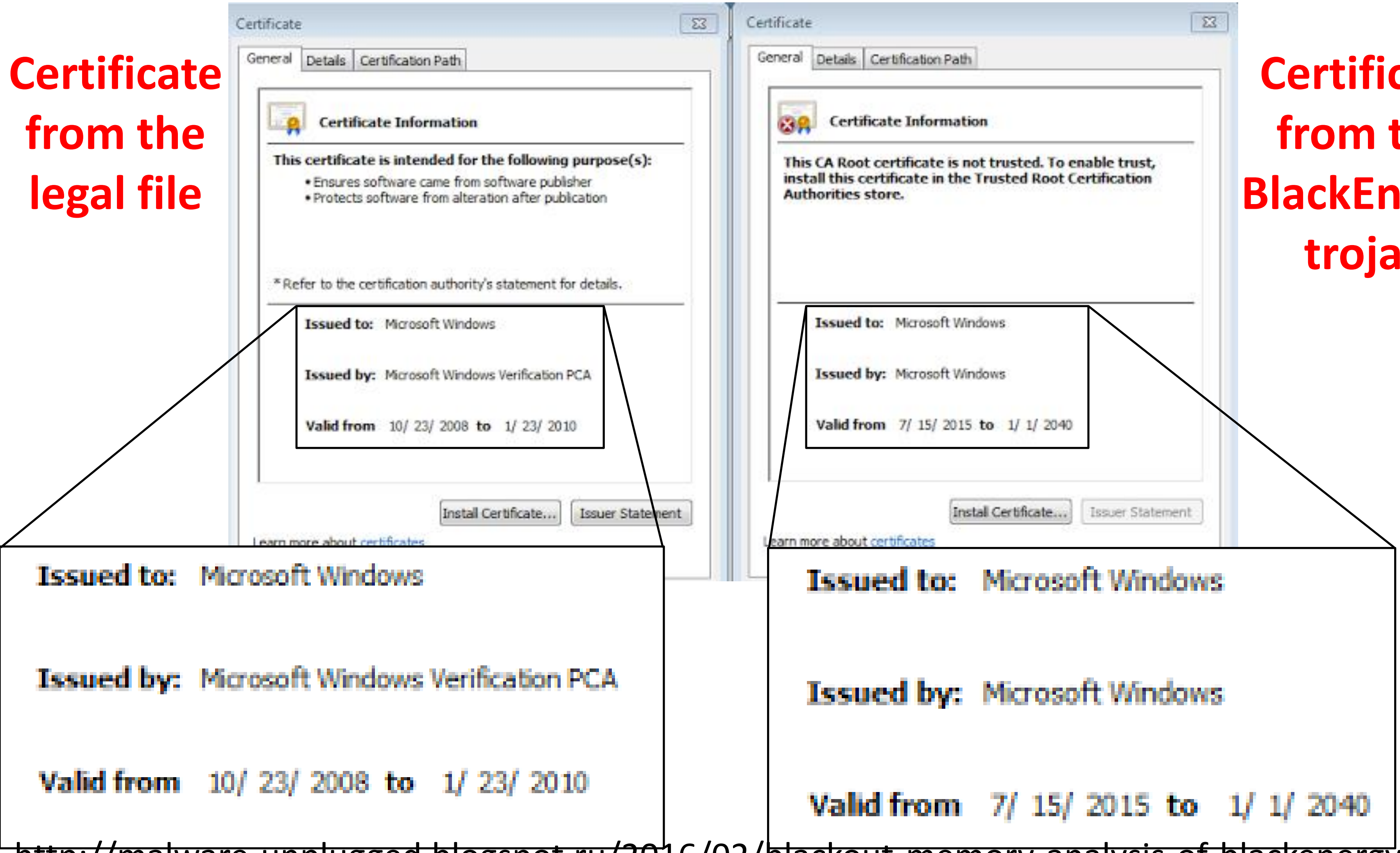
*https://securelist.com/blog/incidents/57647/the-red-october-campaign/

Modern malware in modern world

Well-targeted	Well-prepared
 BlackEnergy trojan Havex malware +20% rise of incidents	— Uses 0-day exploits — Contains fake digital certificates — Applies anti-forensics tricks

Detection becomes too time-consuming

BlackEnergy used fake digital certificates (on the right hand side) and also notice how the expiration date of the certificate is set to 2040



Certificate from the legal file

Certificate Information

This certificate is intended for the following purpose(s):

- Ensures software came from software publisher
- Protects software from alteration after publication

* Refer to the certification authority's statement for details.

Issued to: Microsoft Windows

Issued by: Microsoft Windows Verification PCA

Valid from 10/ 23/ 2008 **to** 1/ 23/ 2010

Issued to: Microsoft Windows

Issued by: Microsoft Windows Verification PCA

Valid from 10/ 23/ 2008 **to** 1/ 23/ 2010

Certificate from the BlackEnergy trojan

Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: Microsoft Windows

Issued by: Microsoft Windows

Valid from 7/ 15/ 2015 **to** 1/ 1/ 2040

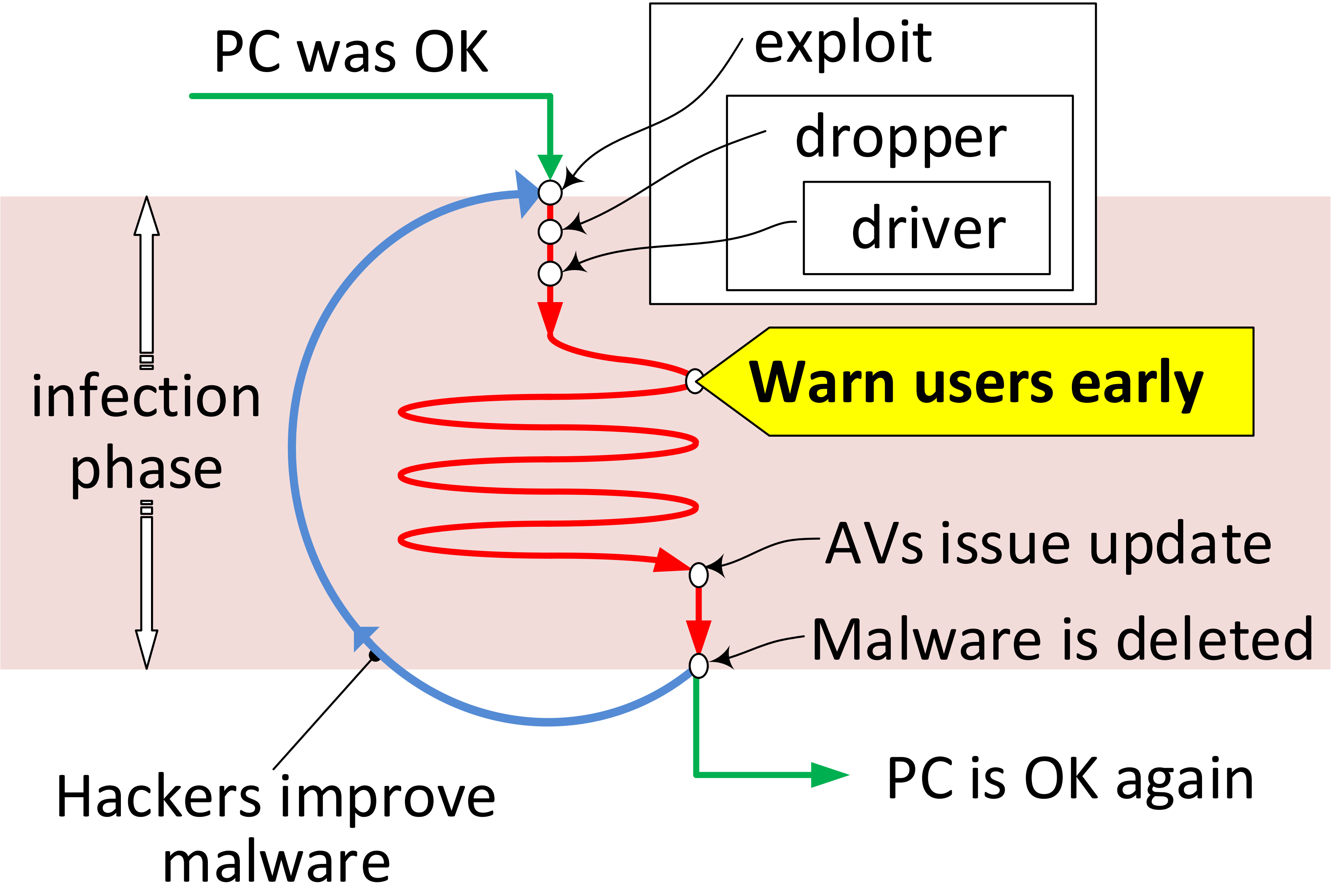
Issued to: Microsoft Windows

Issued by: Microsoft Windows

Valid from 7/ 15/ 2015 **to** 1/ 1/ 2040

http://malware-unplugged.blogspot.ru/2016/02/blackout-memory-analysis-of-blackenergy.html

Inside modern malware & its detection



Cross-view drivers detection

- Match the contents of two lists of drivers:

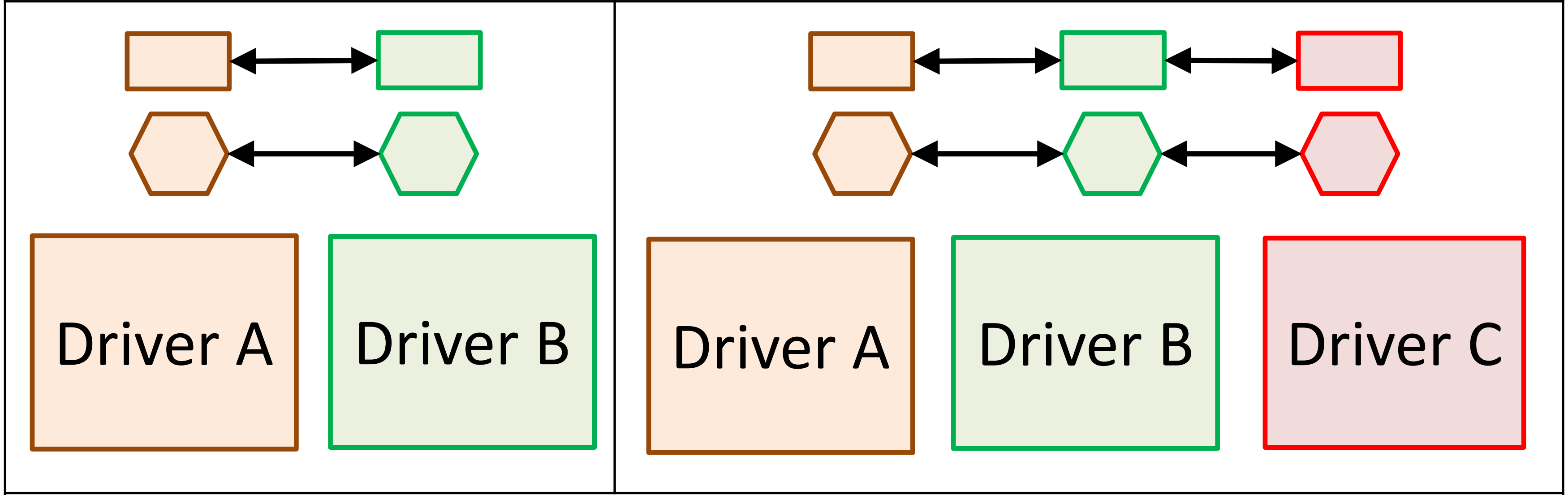
#	Drivers list made by	Example of the lists content	Vulnerable
1	A built-in tool, e.g. ZwQuerySystemInformation	A, B, C	yes
2	An expert	1, 2, 3, 4	Hope not

How to get the 2nd list of drivers?

Warn about a suspicious driver

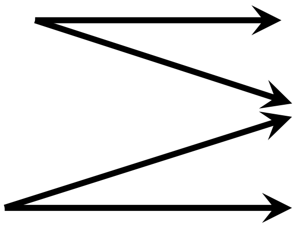
Using updates in the memory content as a source for drivers detection

Original memory | **After a new ‘driver C’ is loaded**



Two updates:

- new structures
- new file driver



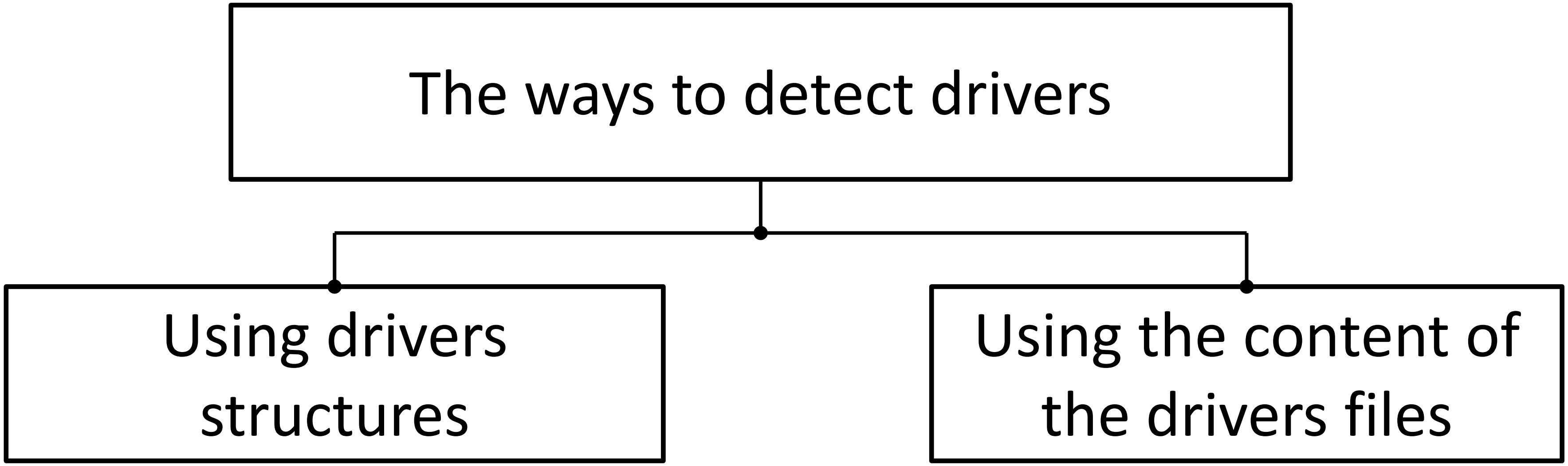
Ways to detect:

○ links

○ signatures

○ statistics

Classification of drivers detection

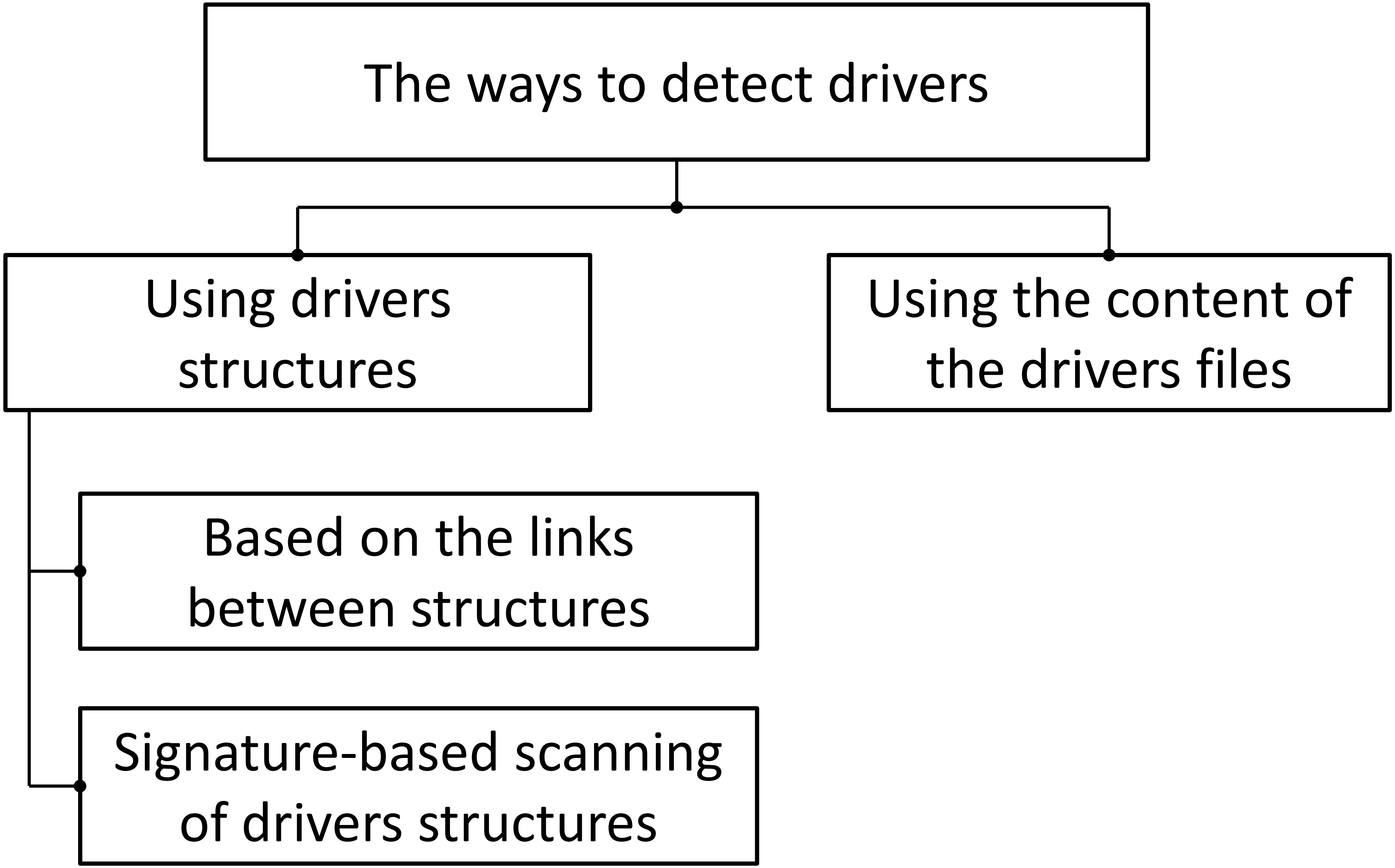


The ways to detect drivers

Using drivers structures

Using the content of the drivers files

Classification of drivers detection



The ways to detect drivers

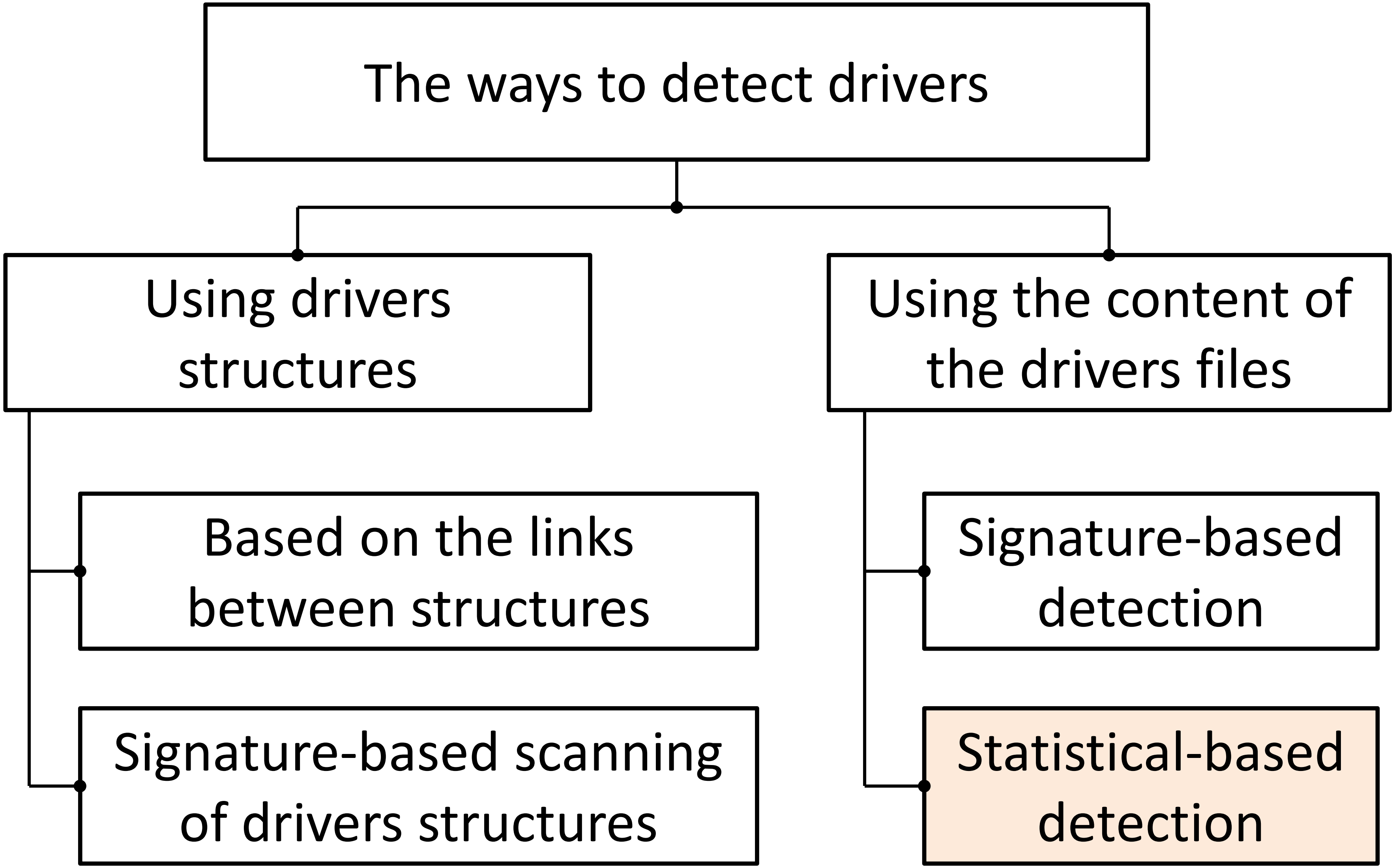
Using drivers structures

Using the content of the drivers files

Based on the links between structures

Signature-based scanning of drivers structures

Classification of drivers detection



The ways to detect drivers

Using drivers structures

Based on the links between structures

Signature-based scanning of drivers structures

Using the content of the drivers files

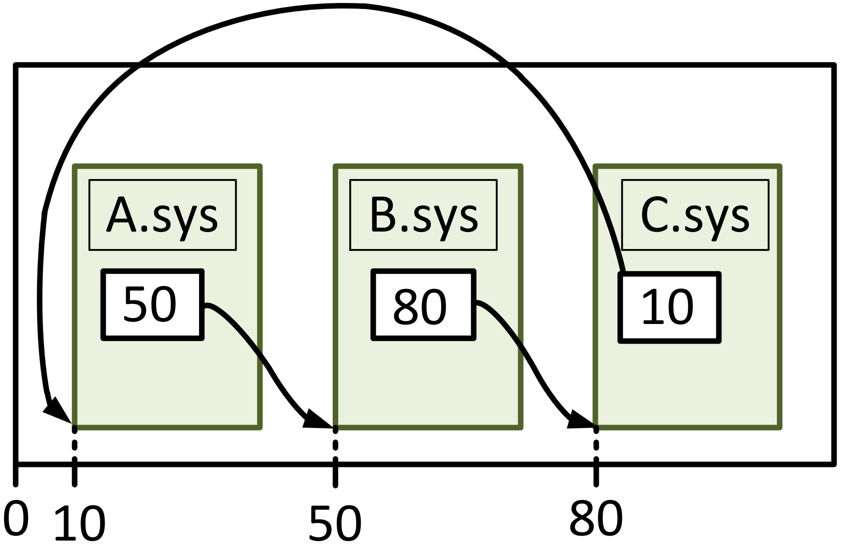
Signature-based detection

Statistical-based detection

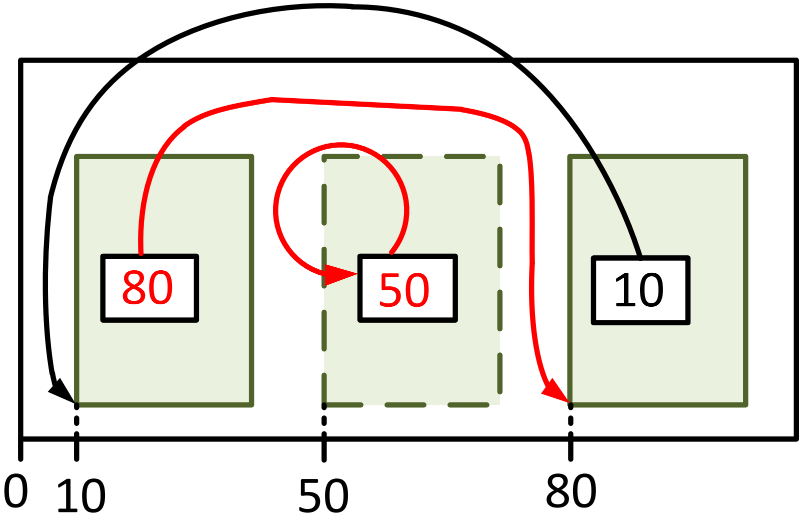
Detect drivers using drivers lists

Drivers list names		Name of structure
PsLoadedModuleList	➢	KLDR_DATA_TABLE_ENTRY
ObjectDirectory	➢	DRIVER_OBJECT
Service record list by SCM	➢	SERVICE_RECORD
Threads from 'System'	➢	ETHREAD
Recently unloaded drivers	➢	UNLOADED_DRIVERS

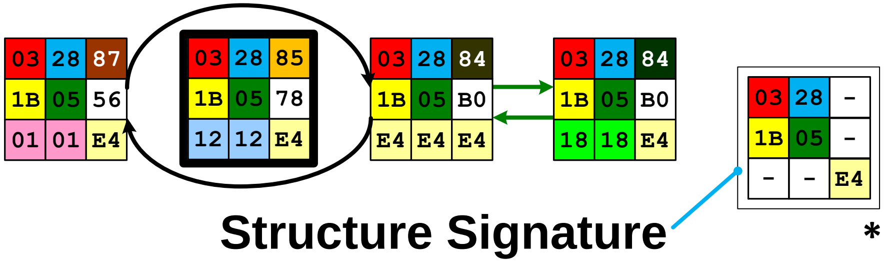
Get a drivers list using links



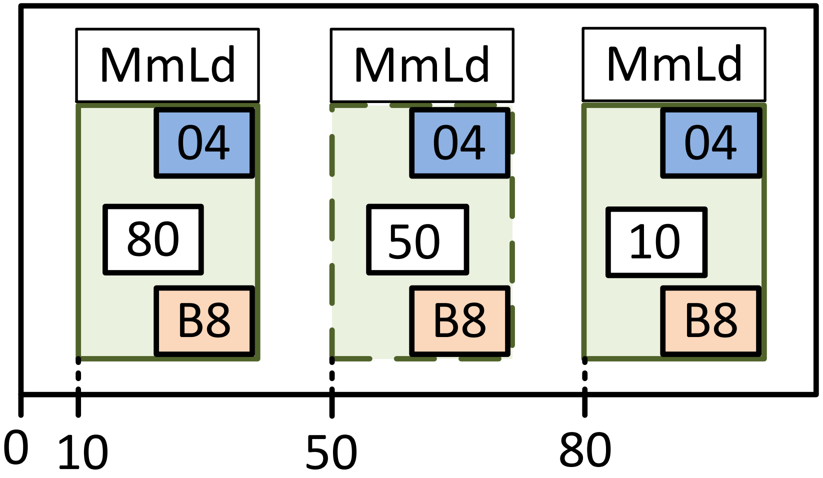
Overcome detection by unlinking



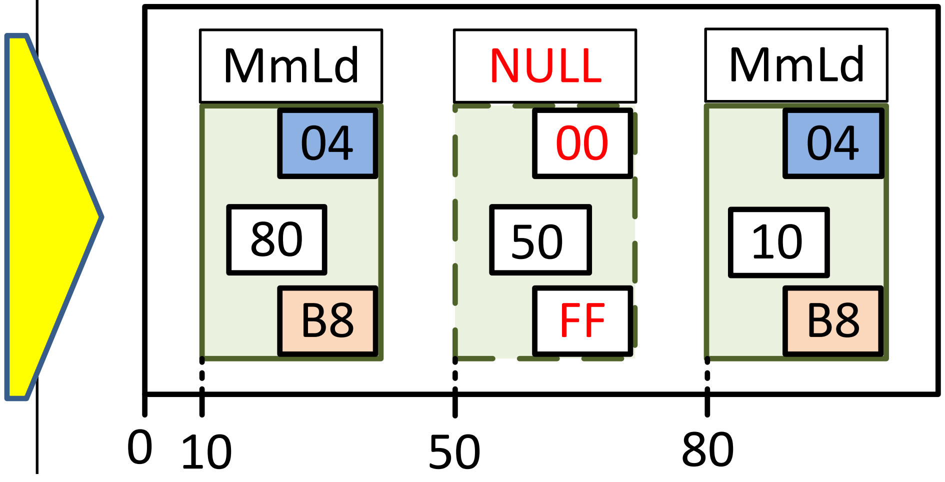
Apply byte-to-byte scanning using structures signatures to detect drivers

Type of structures signature	Examples of signatures and structures
Pool-tag by ExAllocatePoolWithTag	"MmLd" ➢ KLDR_DATA_TABLE_ENTRY "Driv" ➢ DRIVER_OBJECT "sErv" ➢ SERVICE_RECORD
Bytes fragments (fields-based signatures)	 03 28 87 1B 05 56 01 01 E4 03 28 85 1B 05 78 12 12 E4 03 28 84 1B 05 B0 E4 E4 E4 03 28 84 1B 05 B0 18 18 E4 03 28 – 1B 05 – – – E4 Structure Signature * ADFSL'14

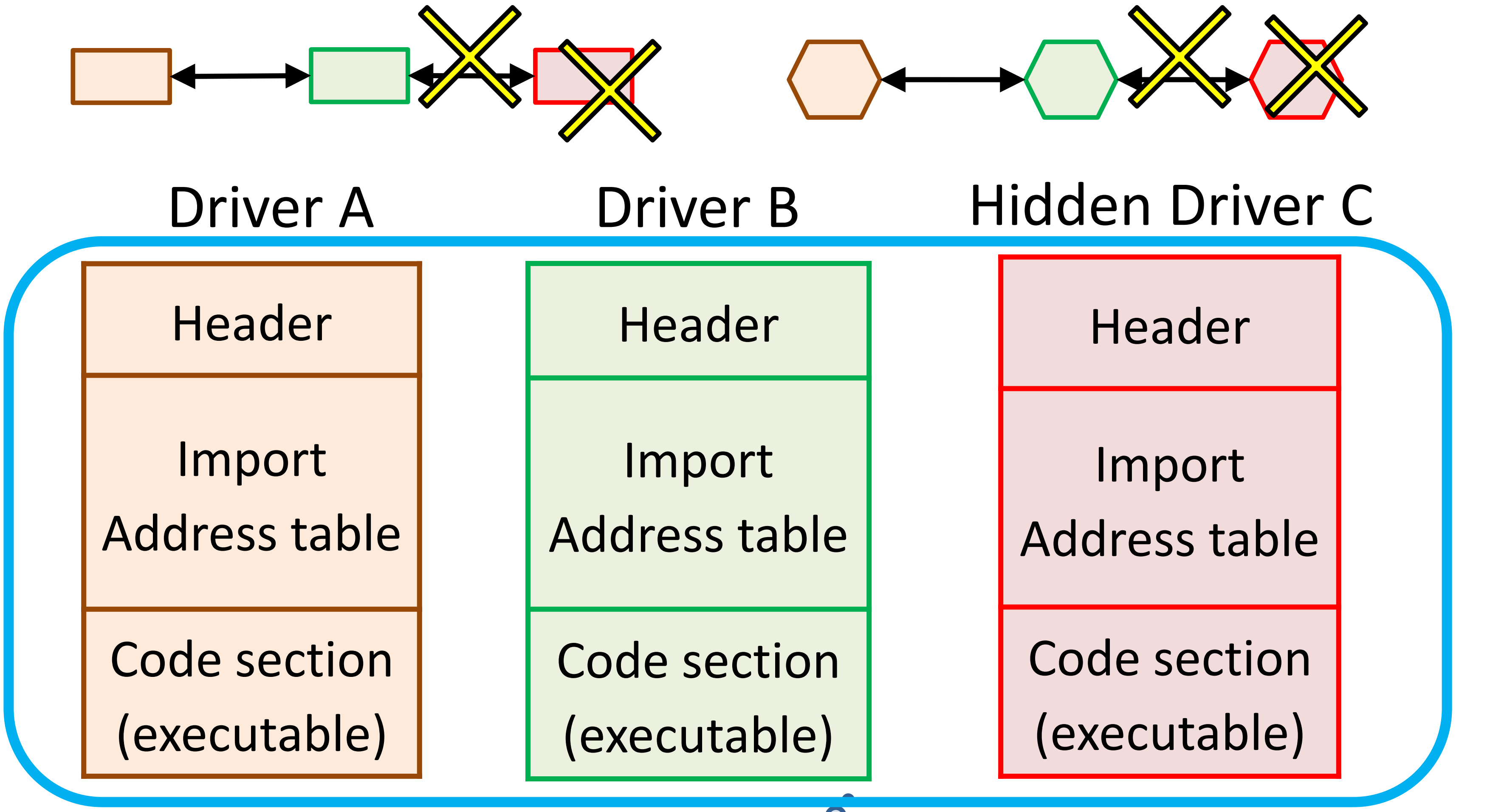
Get a list using structs signatures



Overwrite fields to hide structure



What do we still have in the memory?



Driver A

Header

Import Address table

Code section (executable)

Driver B

Header

Import Address table

Code section (executable)

Hidden Driver C

Header

Import Address table

Code section (executable)

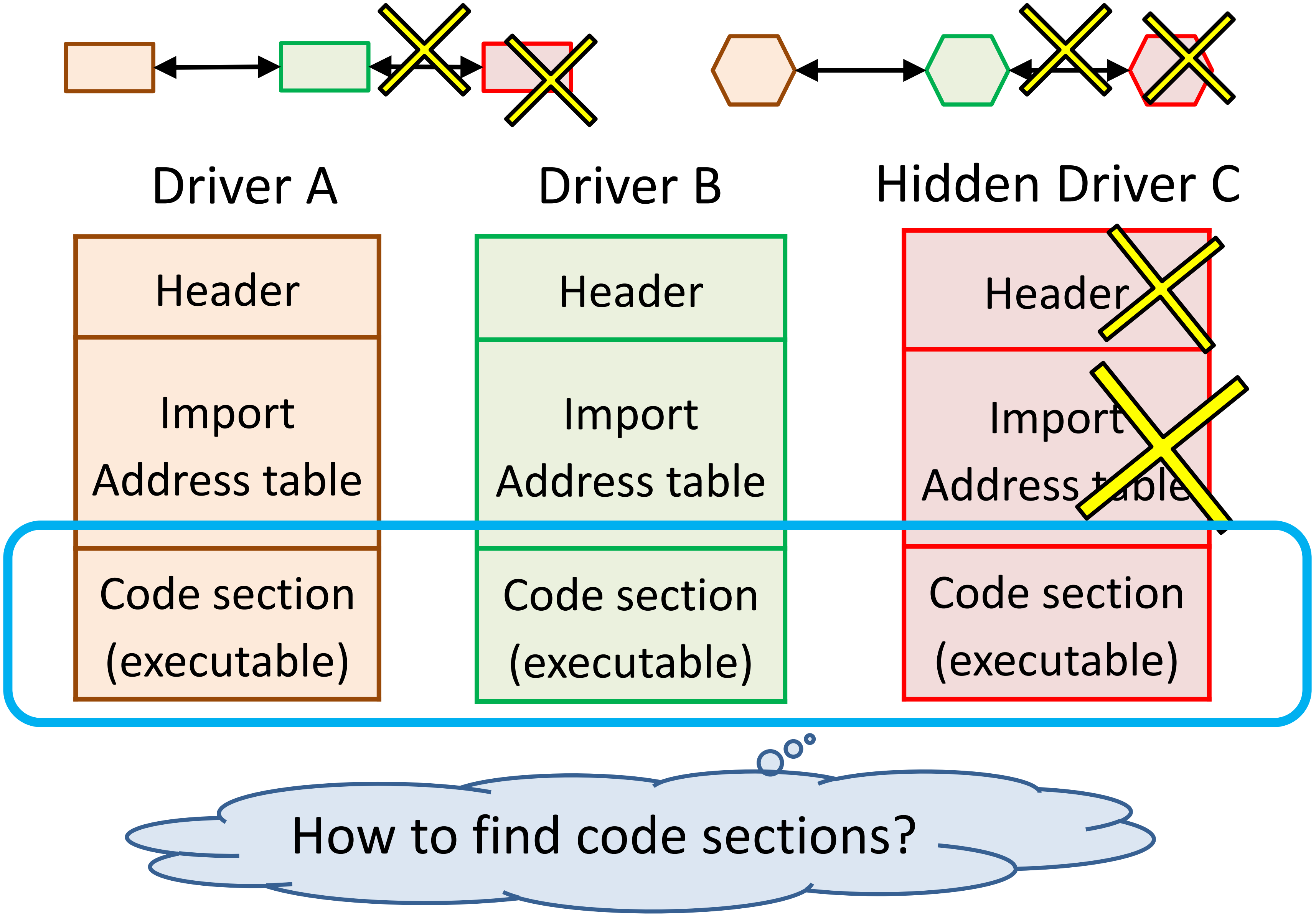
How to find PE-files in the memory?

Apply byte-to-byte scanning using features of PE-file to detect drivers

Driver as a PE-file includes:	PE-file features		Countermeasures
	Type of signature	Examples	
Header	ASCII Strings	‘MZ’ , ‘PE’ , ‘This program cannot be run in DOS mode’	Data overwriting
Import Address table	ASCII Strings	‘ZwOpenFile’	

Code section (executable)	Bytes combination (prologue & epilogue)	`8BFF   MOV EDI,EDI`
		`55     PUSH EBP`
		`8BEC   MOV EBP,ESP`
		
		`8BE5   MOV ESP,EBP`
		`5D     POP EBP`
		`C20400 RET 4`
			Code obfuscating & packing

What do we still have in the memory?



Driver A

Driver B

Hidden Driver C

Header

Import Address table

Code section (executable)

Header

Import Address table

Code section (executable)

Header

Import Address table

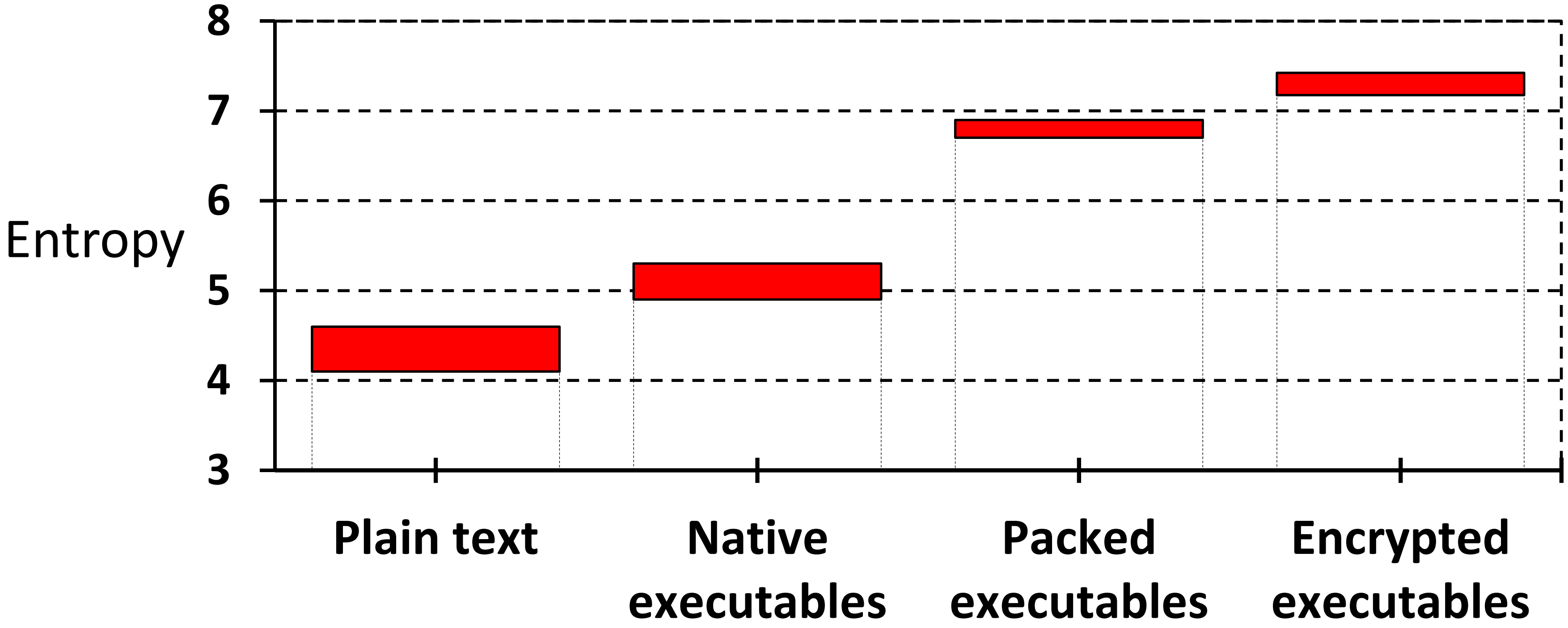
Code section (executable)

How to find code sections?

Using binary Entropy to separate data types

Definition: $$S = -\sum_{i=1}^{255} p_i * \log_2 p_i$$

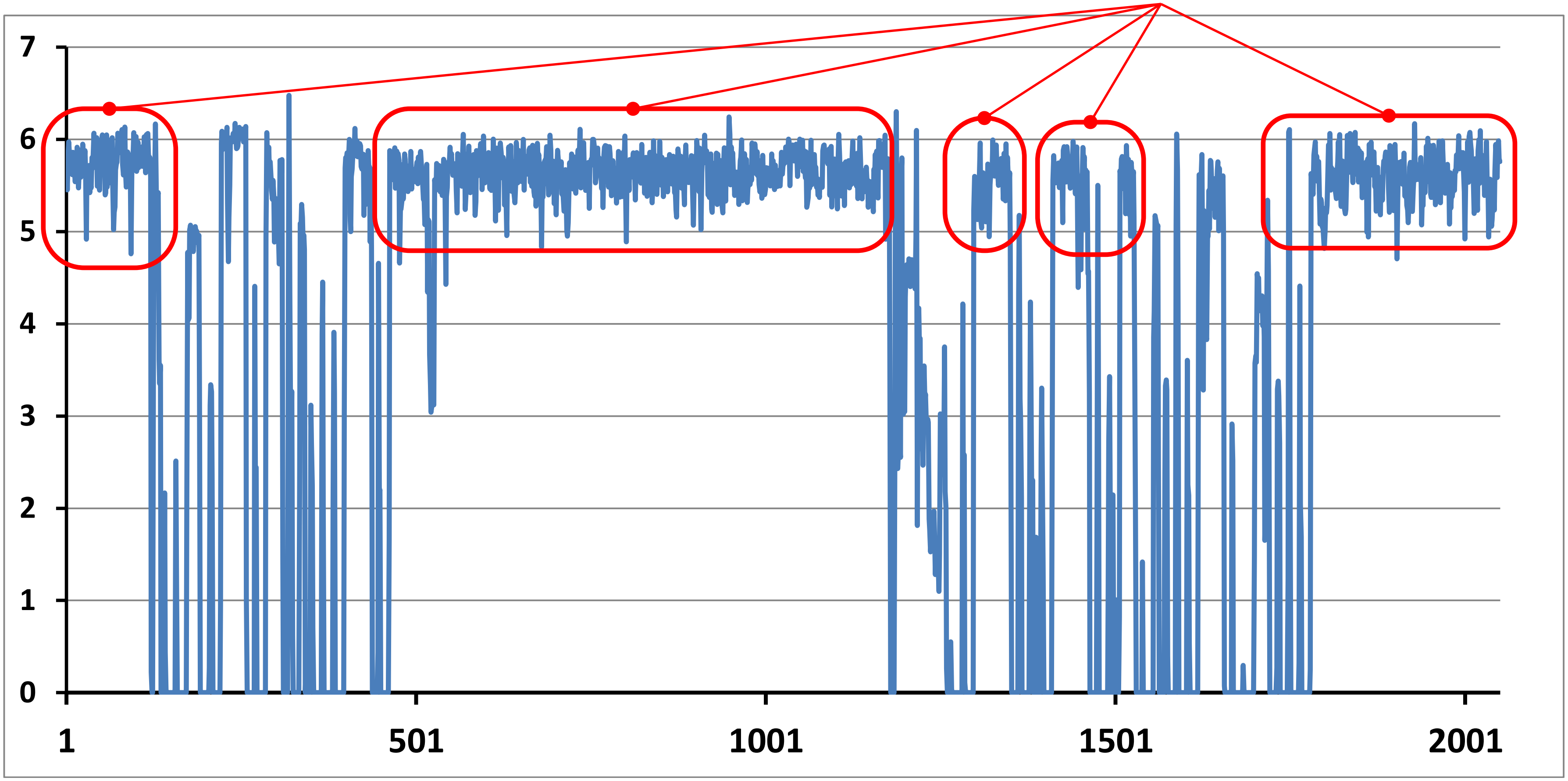
$$p_i - the\ frequency\ of\ each\ byte\ value\ in\ the\ file.$$



*Using Entropy Analysis to Find Encrypted and Packed Malware by R. Lyda & J. Hamrock

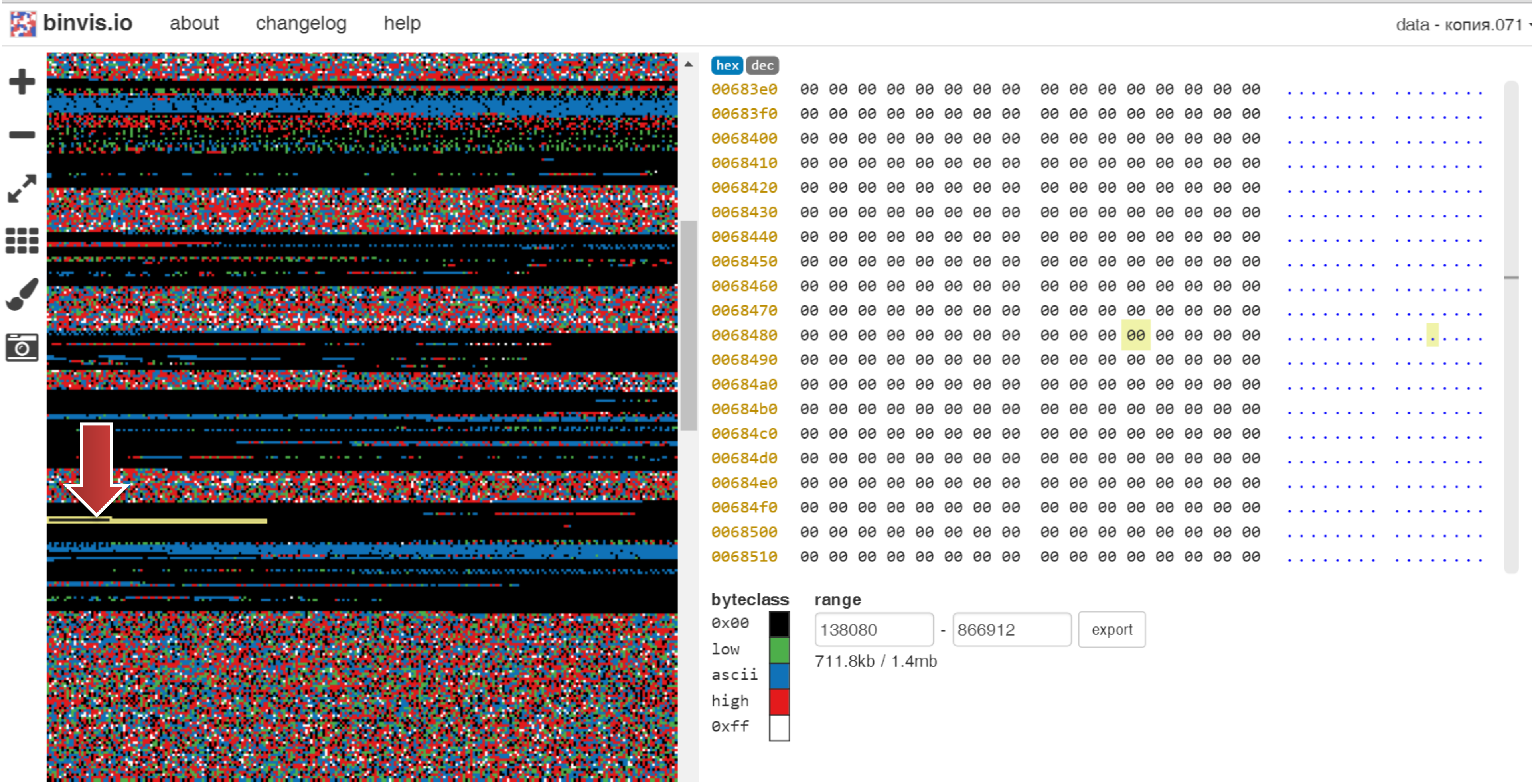
Using sliding-window approach to locate executable code

E.g. **Window size is 256 byte** **Overlap is 256 byte**



Offsets

Colored diagram of memory dump via binvis



```
00683e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
00683f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0068400  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0068410  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0068420  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0068430  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0068440  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0068450  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0068460  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0068470  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0068480  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0068490  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
00684a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
00684b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
00684c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
00684d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
00684e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
00684f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0068500  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0068510  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
```

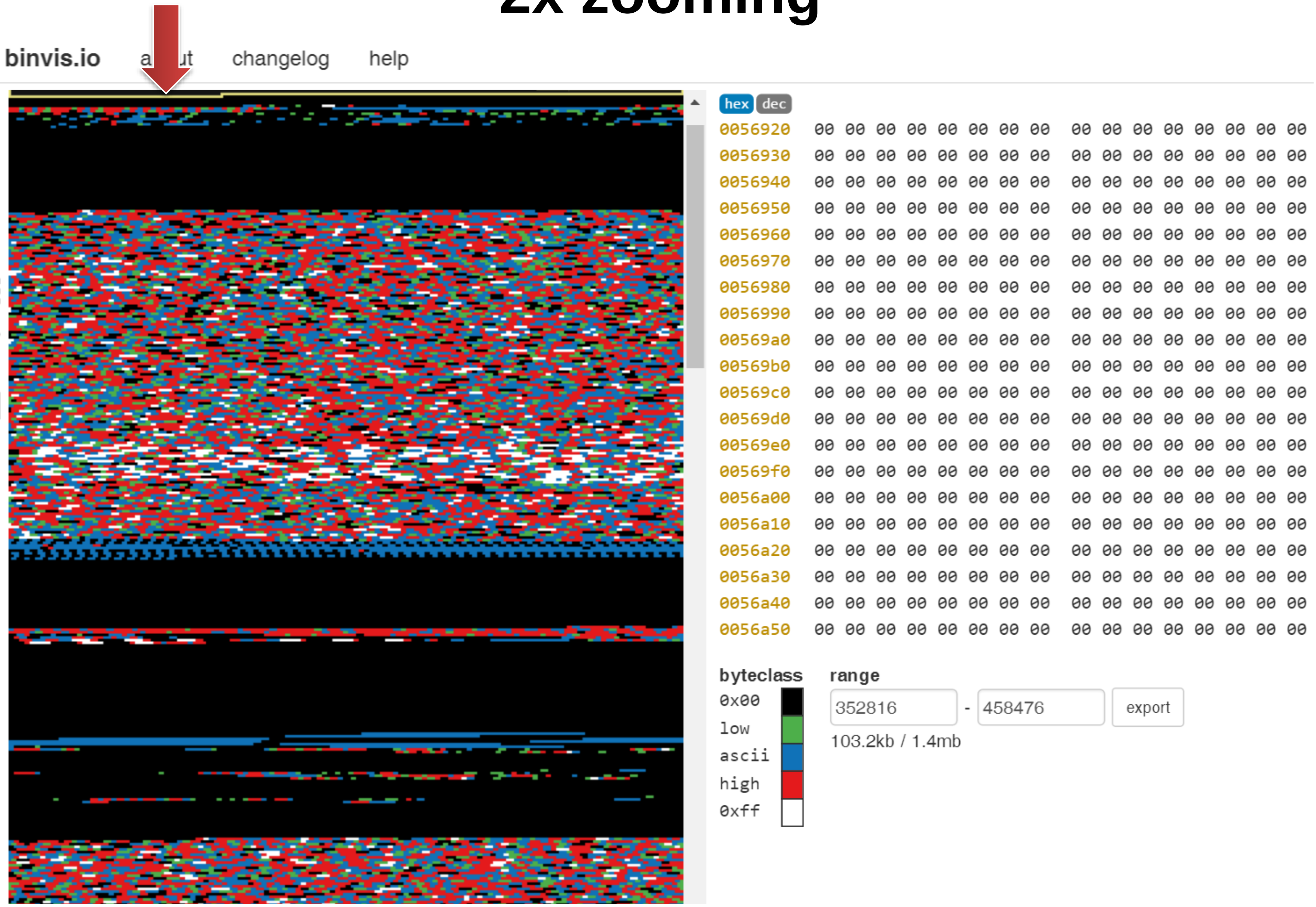
byteclass	
0x00	⬛
low	🟩
ascii	🟦
high	🟥
0xff	⬜

range

138080 - 866912

711.8kb / 1.4mb

2x zooming



binvis.io about changelog help

hex dec

```
0056920  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0056930  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0056940  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0056950  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0056960  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0056970  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0056980  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0056990  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00569a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00569b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00569c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00569d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00569e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00569f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0056a00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0056a10  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0056a20  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0056a30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0056a40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0056a50  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

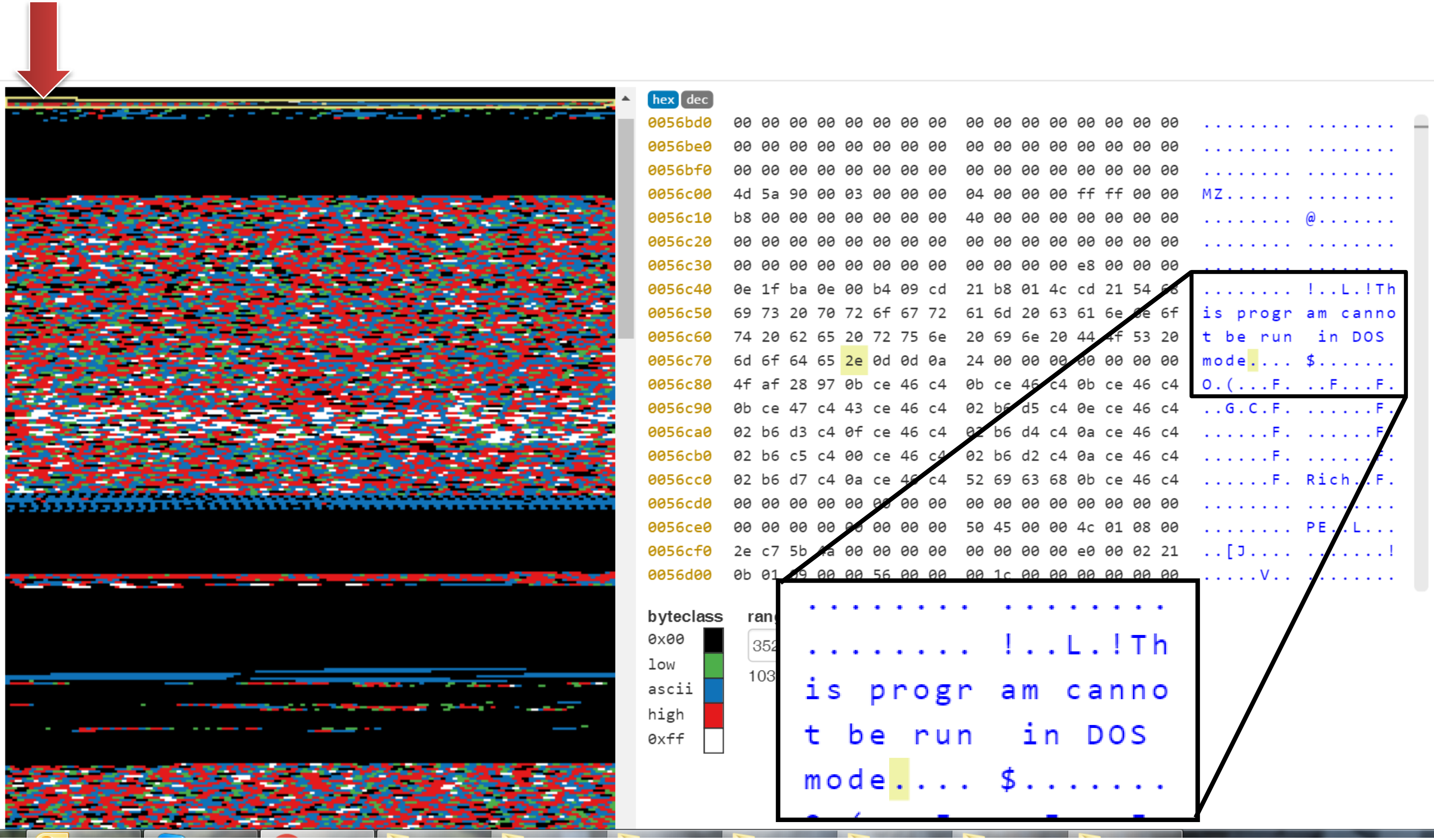
byteclass	
0x00	⬛
low	🟩
ascii	🟦
high	🟥
0xff	⬜

range

352816 - 458476 export

103.2kb / 1.4mb

PE header of the driver file

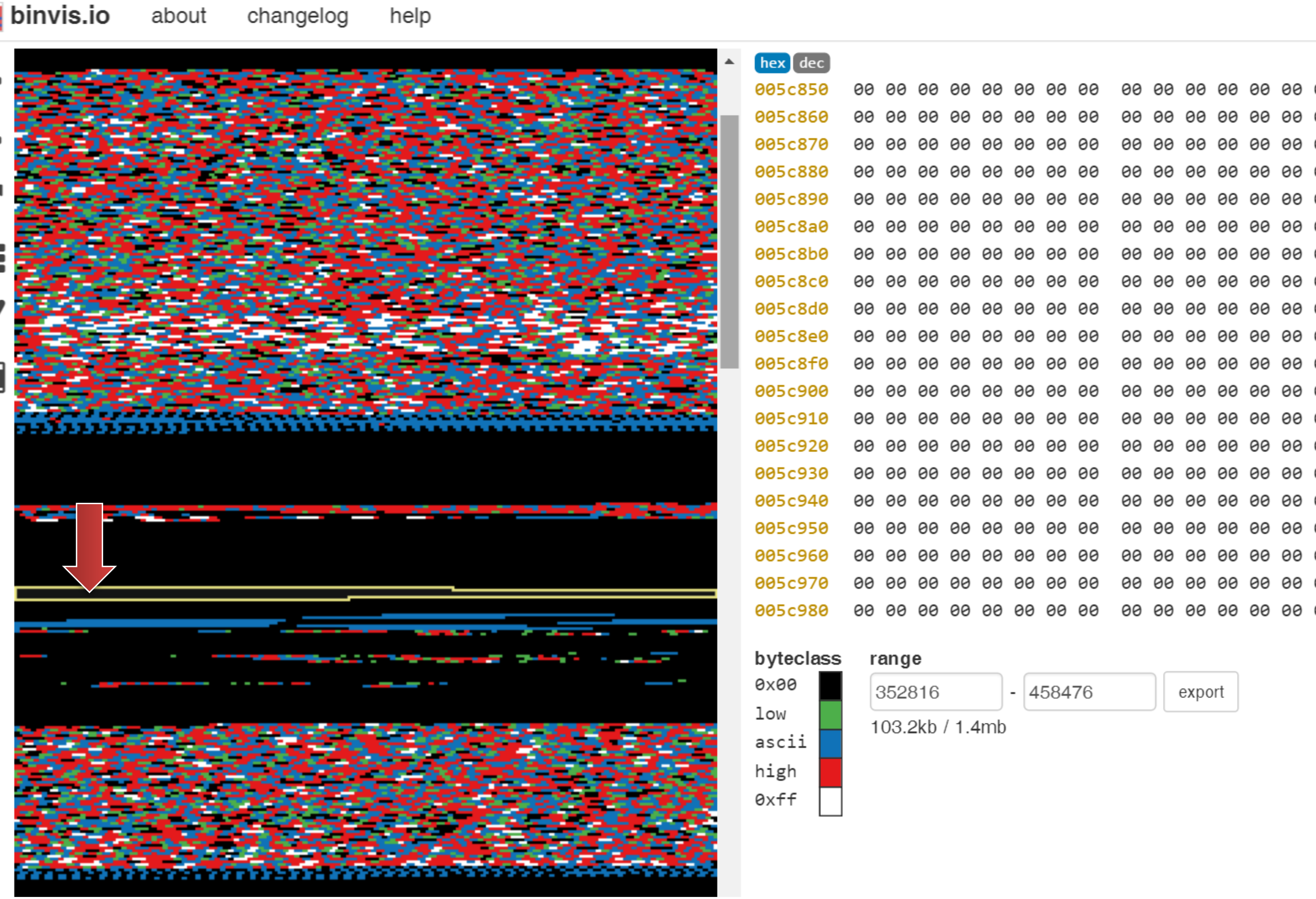


```
0056bd0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0056be0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0056bf0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0056c00  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  MZ...... ........
0056c10  b8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  ........ @.......
0056c20  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0056c30  00 00 00 00 00 00 00 00  00 00 00 00 e8 00 00 00  ........ ........
0056c40  0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 54 68  ........ !..L.!Th
0056c50  69 73 20 70 72 6f 67 72  61 6d 20 63 61 6e 6e 6f  is progr am canno
0056c60  74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20  t be run  in DOS
0056c70  6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00  mode.... $.......
0056c80  4f af 28 97 0b ce 46 c4  0b ce 46 c4 0b ce 46 c4  O.(...F. ..F...F.
0056c90  0b ce 47 c4 43 ce 46 c4  02 b6 d5 c4 0e ce 46 c4  ..G.C.F. ......F.
0056ca0  02 b6 d3 c4 0f ce 46 c4  02 b6 d4 c4 0a ce 46 c4  ......F. ......F.
0056cb0  02 b6 c5 c4 00 ce 46 c4  02 b6 d2 c4 0a ce 46 c4  ......F. ......F.
0056cc0  02 b6 d7 c4 0a ce 46 c4  52 69 63 68 0b ce 46 c4  ......F. Rich..F.
0056cd0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0056ce0  00 00 00 00 00 00 00 00  50 45 00 00 4c 01 08 00  ........ PE..L...
0056cf0  2e c7 5b 4a 00 00 00 00  00 00 00 00 e0 00 02 21  ..[J.... .......!
0056d00  0b 01 09 00 00 56 00 00  00 1c 00 00 00 00 00 00  .....V.. ........
```

```
........ ........
........ !..L.!Th
is progr am canno
t be run  in DOS
mode.... $.......
```

byteclass	
0x00	■
low	🟩
ascii	🟦
high	🟥
0xff	□

Move down



```
005c850   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c860   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c870   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c880   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c890   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c8a0   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c8b0   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c8c0   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c8d0   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c8e0   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c8f0   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c900   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c910   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c920   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c930   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c940   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c950   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c960   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c970   00 00 00 00 00 00 00 00   00 00 00 00 00 00
005c980   00 00 00 00 00 00 00 00   00 00 00 00 00 00
```

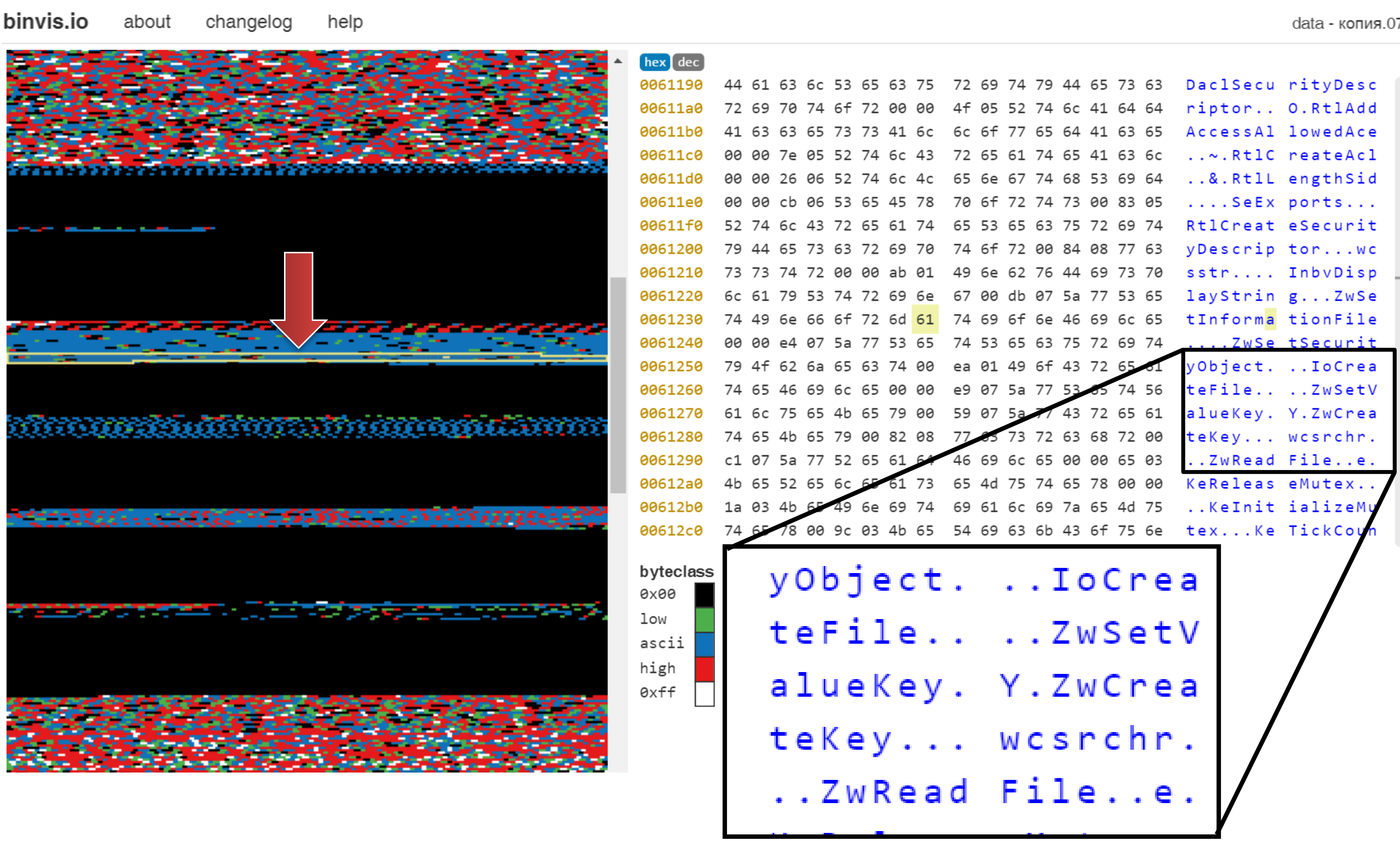
byteclass	
0x00	⬛
low	🟩
ascii	🟦
high	🟥
0xff	⬜

range

352816 - 458476 export

103.2kb / 1.4mb

Import table includes functions names

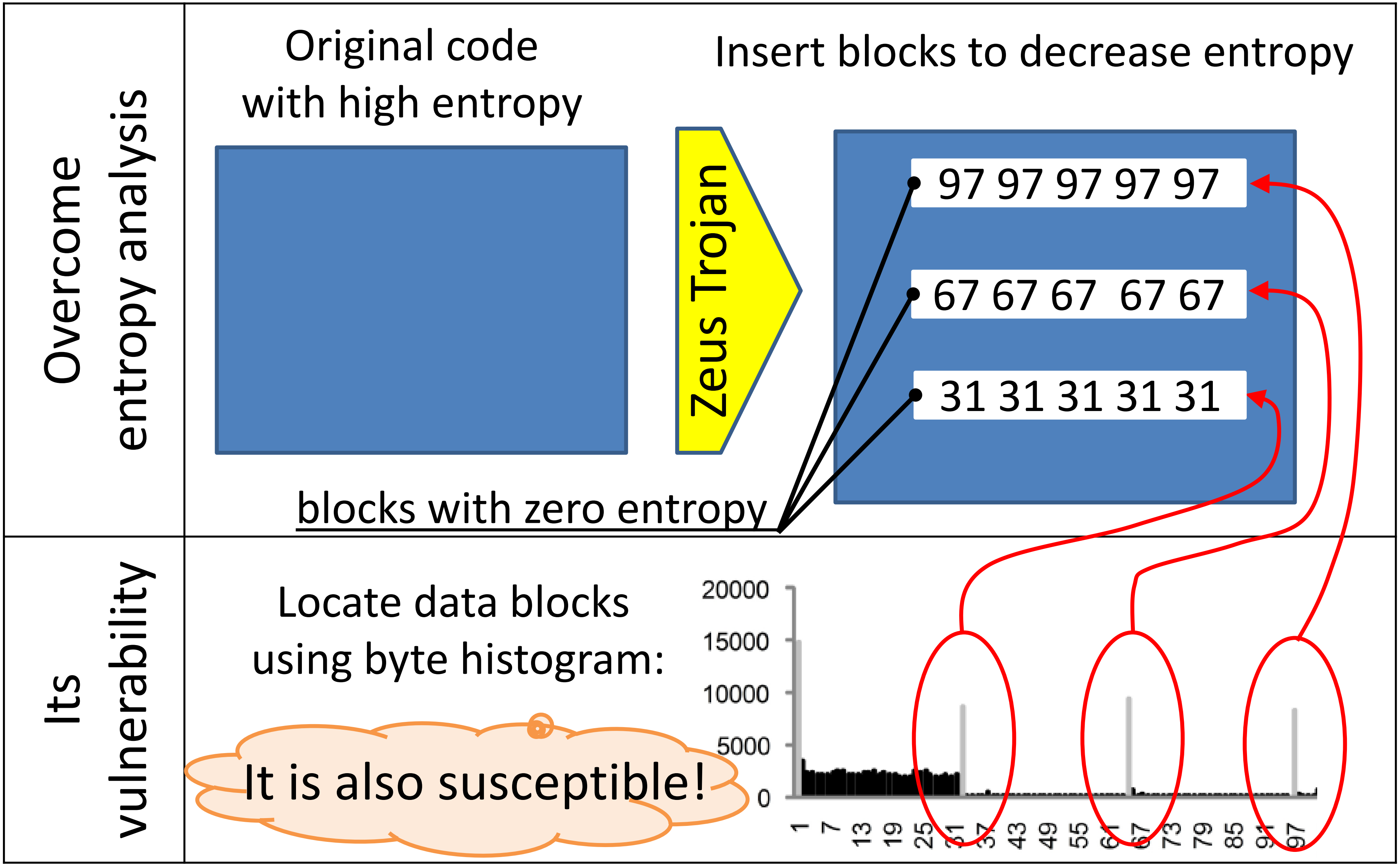


```
0061190  44 61 63 6c 53 65 63 75  72 69 74 79 44 65 73 63  DaclSecu rityDesc
00611a0  72 69 70 74 6f 72 00 00  4f 05 52 74 6c 41 64 64  riptor.. O.RtlAdd
00611b0  41 63 63 65 73 73 41 6c  6c 6f 77 65 64 41 63 65  AccessAl lowedAce
00611c0  00 00 7e 05 52 74 6c 43  72 65 61 74 65 41 63 6c  ..~.RtlC reateAcl
00611d0  00 00 26 06 52 74 6c 4c  65 6e 67 74 68 53 69 64  ..&.RtlL engthSid
00611e0  00 00 cb 06 53 65 45 78  70 6f 72 74 73 00 83 05  ....SeEx ports...
00611f0  52 74 6c 43 72 65 61 74  65 53 65 63 75 72 69 74  RtlCreat eSecurit
0061200  79 44 65 73 63 72 69 70  74 6f 72 00 84 08 77 63  yDescrip tor...wc
0061210  73 73 74 72 00 00 ab 01  49 6e 62 76 44 69 73 70  sstr.... InbvDisp
0061220  6c 61 79 53 74 72 69 6e  67 00 db 07 5a 77 53 65  layStrin g...ZwSe
0061230  74 49 6e 66 6f 72 6d 61  74 69 6f 6e 46 69 6c 65  tInforma tionFile
0061240  00 00 e4 07 5a 77 53 65  74 53 65 63 75 72 69 74  ....ZwSe tSecurit
0061250  79 4f 62 6a 65 63 74 00  ea 01 49 6f 43 72 65 61  yObject. ..IoCrea
0061260  74 65 46 69 6c 65 00 00  e9 07 5a 77 53 65 74 56  teFile.. ..ZwSetV
0061270  61 6c 75 65 4b 65 79 00  59 07 5a 77 43 72 65 61  alueKey. Y.ZwCrea
0061280  74 65 4b 65 79 00 82 08  77 63 73 72 63 68 72 00  teKey... wcsrchr.
0061290  c1 07 5a 77 52 65 61 64  46 69 6c 65 00 00 65 03  ..ZwRead File..e.
00612a0  4b 65 52 65 6c 65 61 73  65 4d 75 74 65 78 00 00  KeReleas eMutex..
00612b0  1a 03 4b 65 49 6e 69 74  69 61 6c 69 7a 65 4d 75  ..KeInit ializeMu
00612c0  74 65 78 00 9c 03 4b 65  54 69 63 6b 43 6f 75 6e  tex...Ke TickCoun
```

byteclass
0x00 ⬛
low 🟩
ascii 🟦
high 🟥
0xff ⬜

yObject. ..IoCrea
teFile.. ..ZwSetV
alueKey. Y.ZwCrea
teKey... wcsrchr.
..ZwRead File..e.

An idea to overcome entropy analysis & its vulnerability



Overcome entropy analysis

Original code with high entropy

Insert blocks to decrease entropy

blocks with zero entropy

Its vulnerability

Locate data blocks using byte histogram:

It is also susceptible!

All detection methods are vulnerable

The ways to detect drivers		**Anti-forensic technique**
Using drivers structures	Using links between structures	Unlinking
	Signature-based scanning	Overwriting
Using content of drivers files	Signature-based scanning	Overwriting & PE packing
	Statistical-based detection	Inserting data blocks

Let’s consider the most difficult case for detection - HighStem

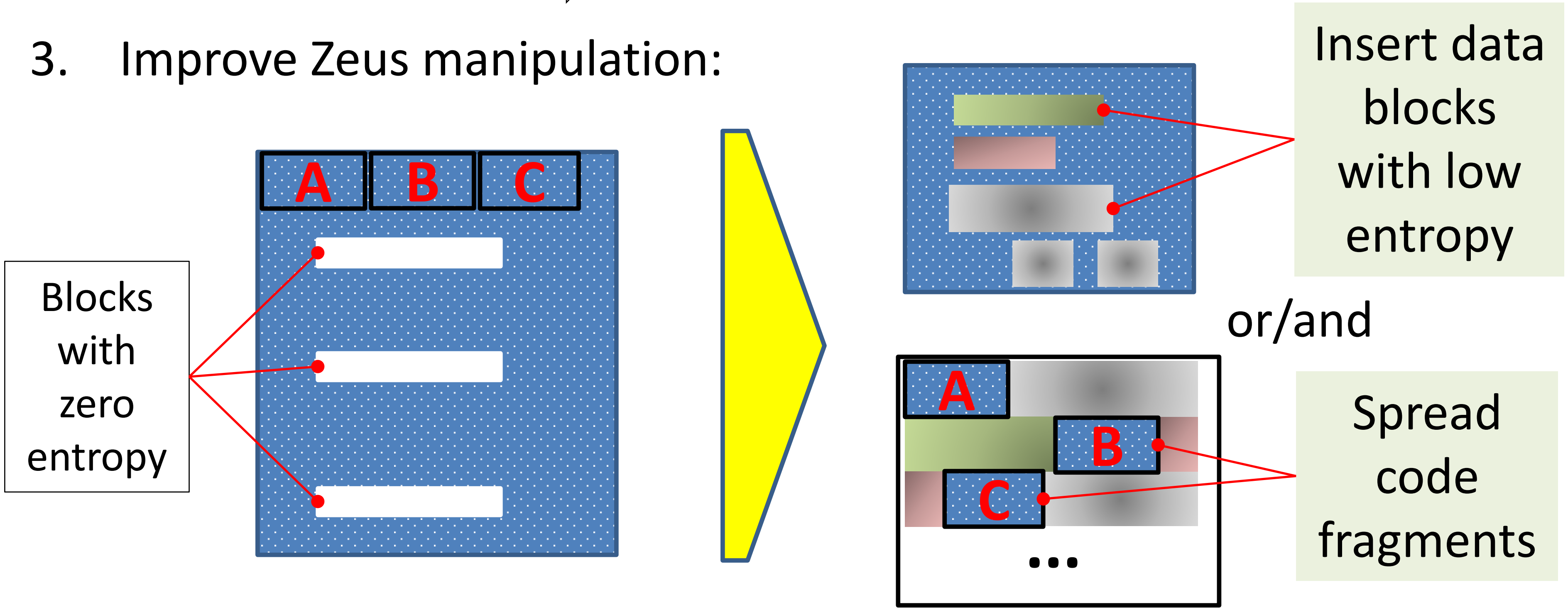
Highest Stealth Malware (HighStem) imitates the most difficult case for detection

1. Apply Atsiv or Turla Driver Loader to load a HighStem driver
2. Collect data without OS function:

Read keystrokes from the memory

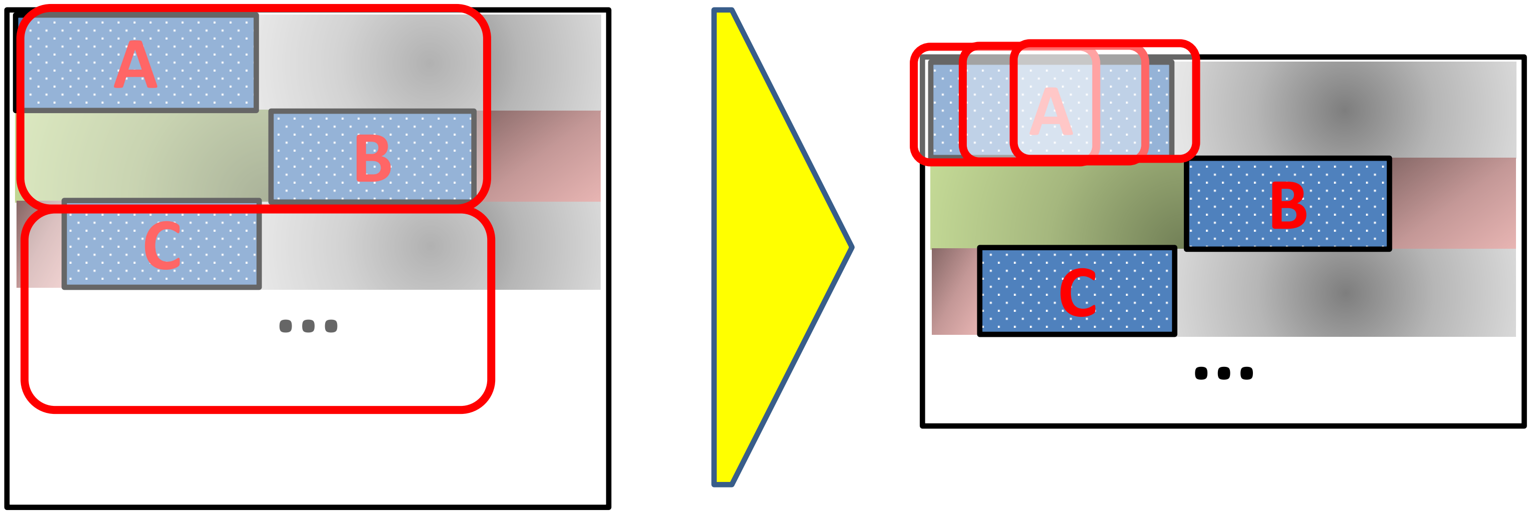
kbdhid.sys
→ DEVICE_EXTENSION
→KEYBOARD_ATTRIBUTES

3. Improve Zeus manipulation:

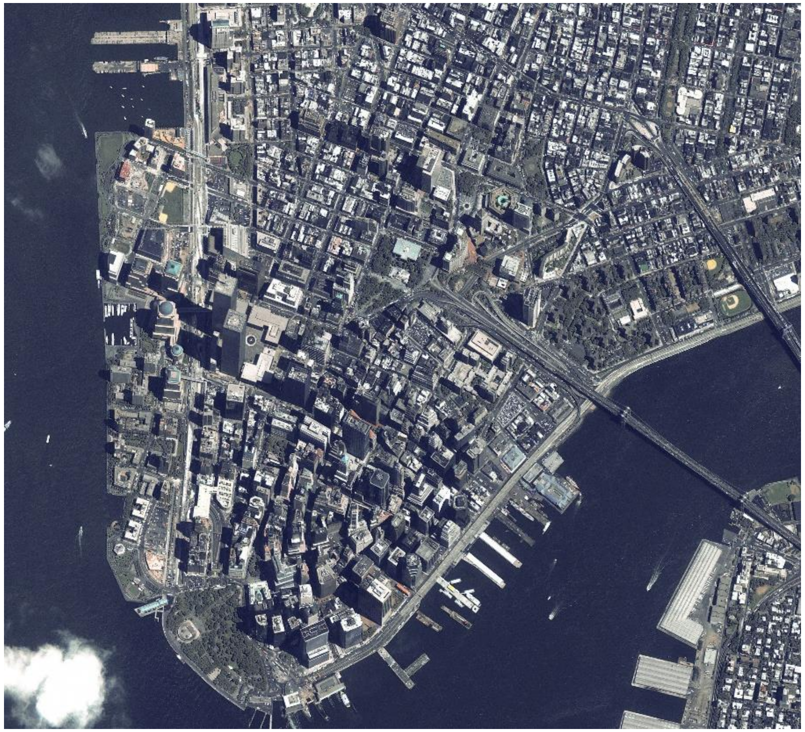


How to reveal all parts of diluted executable?

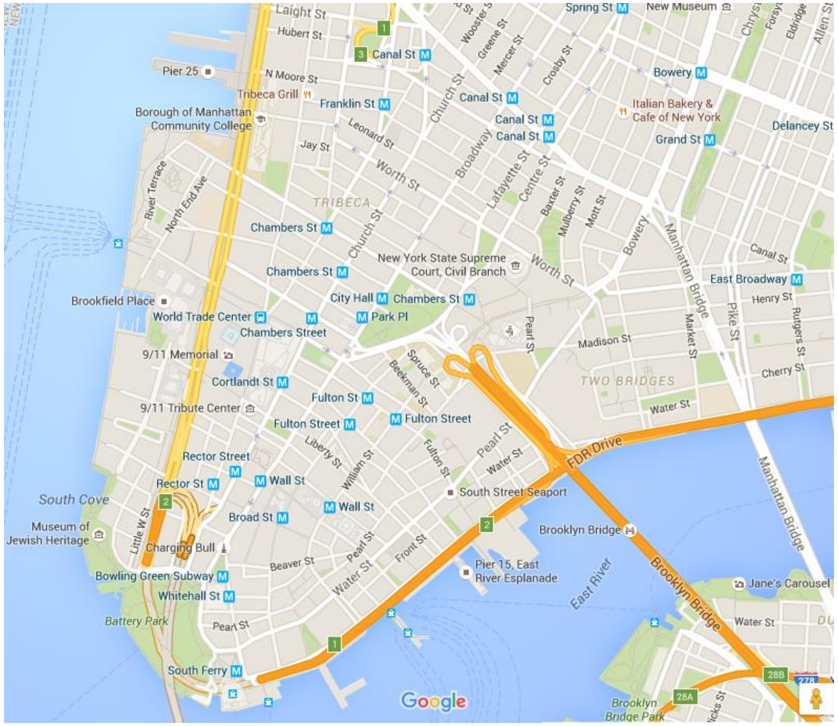
- Calculate entropy using smaller window size

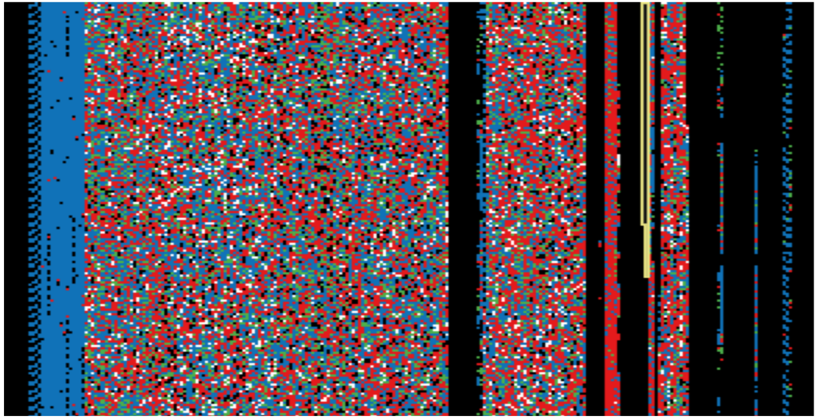


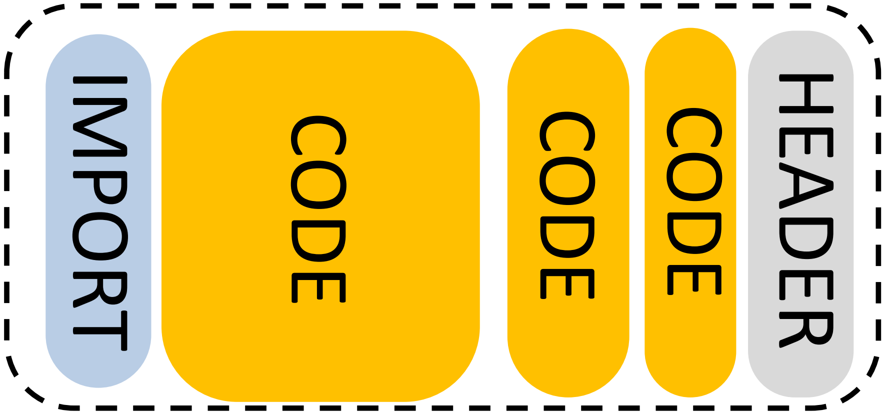
Apply digital photogrammetry to locate a code



*Lower Manhattan, November of 2000 by AirPhotoUSA







IMPORT

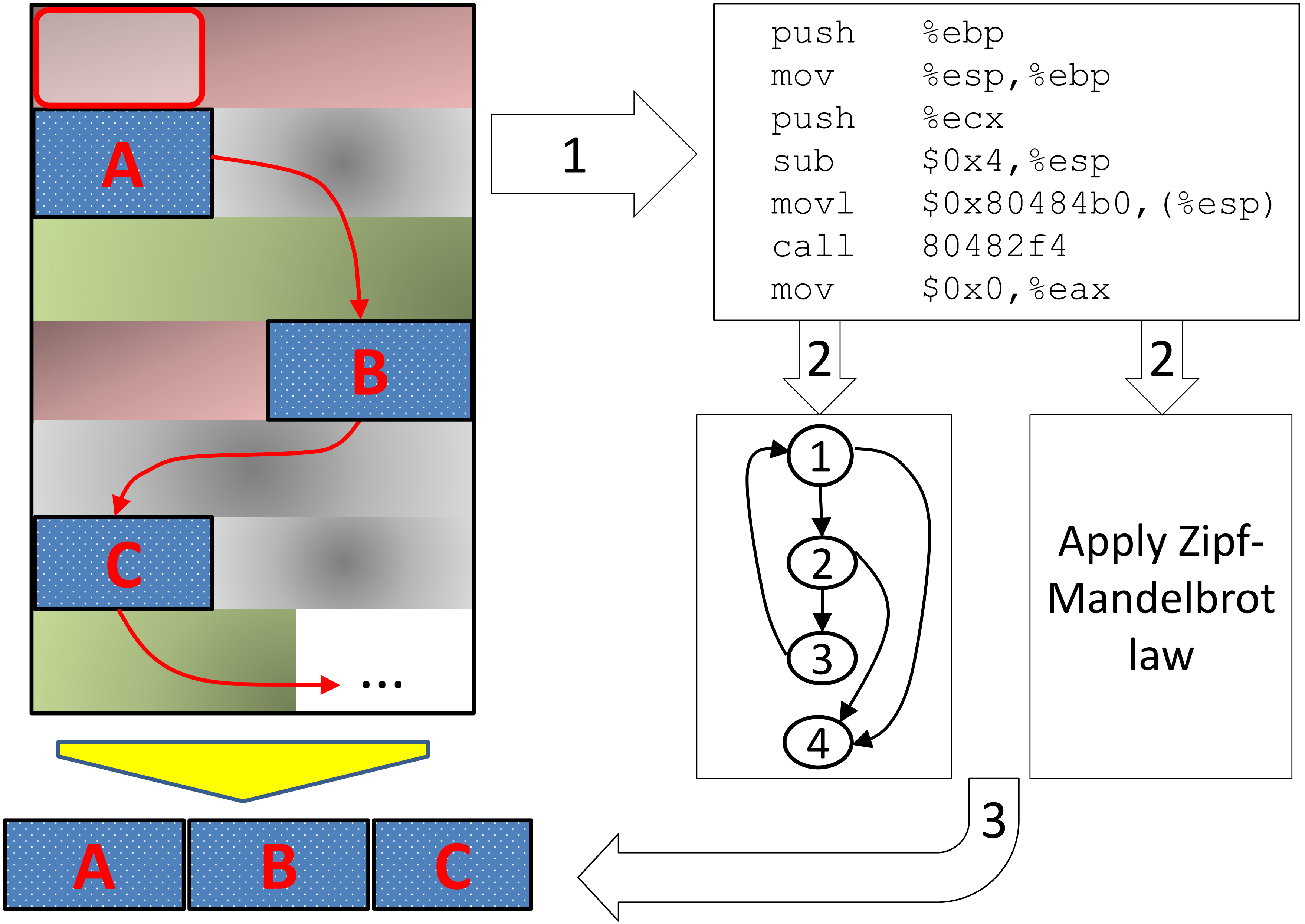
CODE

CODE

CODE

HEADER

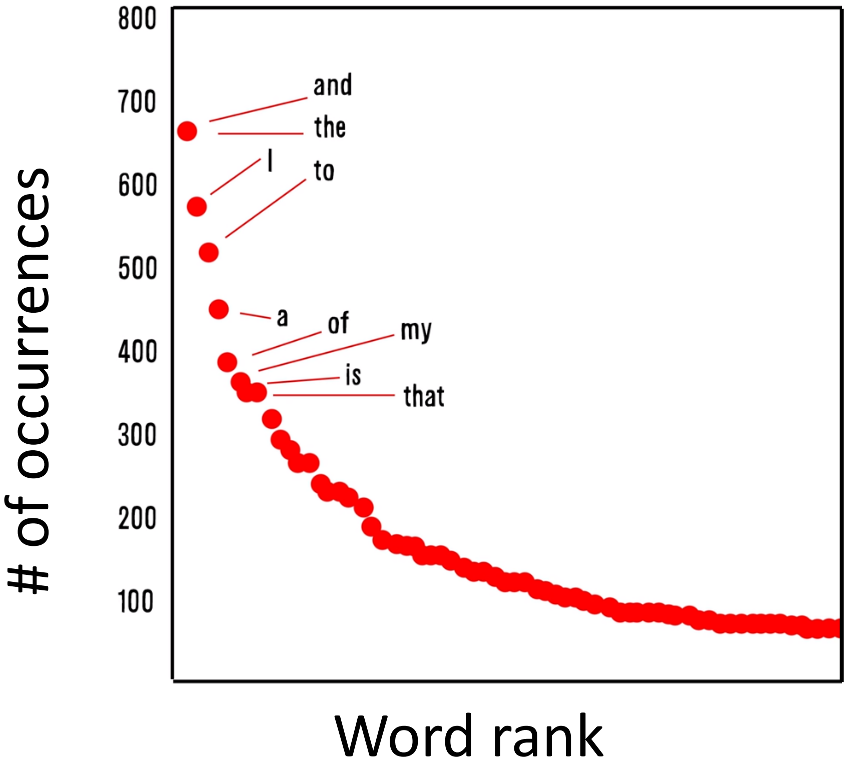
Analyze the disassembly code

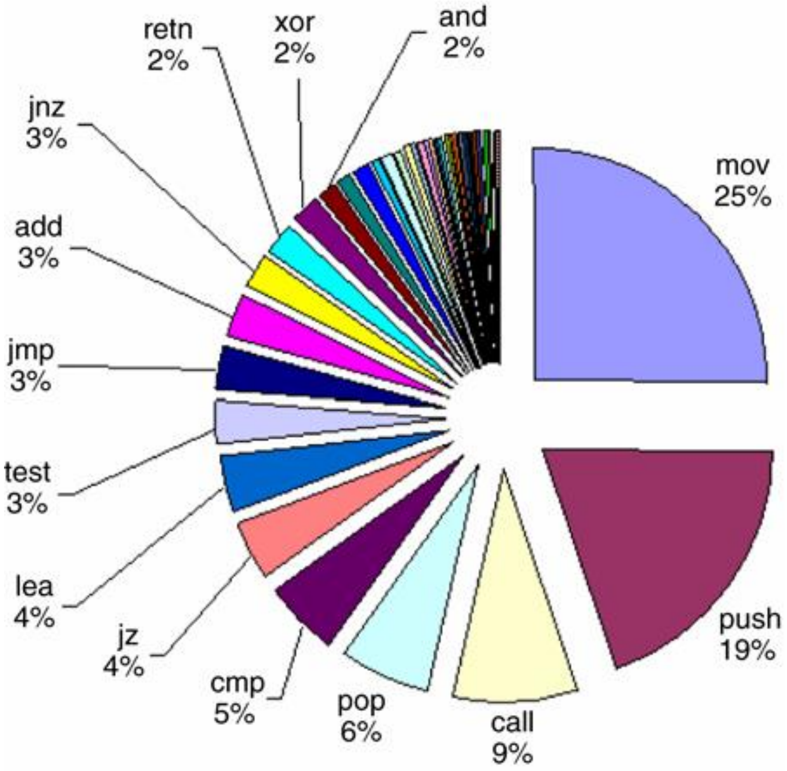


```
push    %ebp
mov     %esp,%ebp
push    %ecx
sub     $0x4,%esp
movl    $0x80484b0,(%esp)
call    80482f4
mov     $0x0,%eax
```

Apply Zipf-Mandelbrot law

Zipf-Mandelbrot law
From linguistics to disassembly listings





Zipf Law

$$p(i) * i = C = const,$$

$$p(i) - frequency\ of\ i - word$$

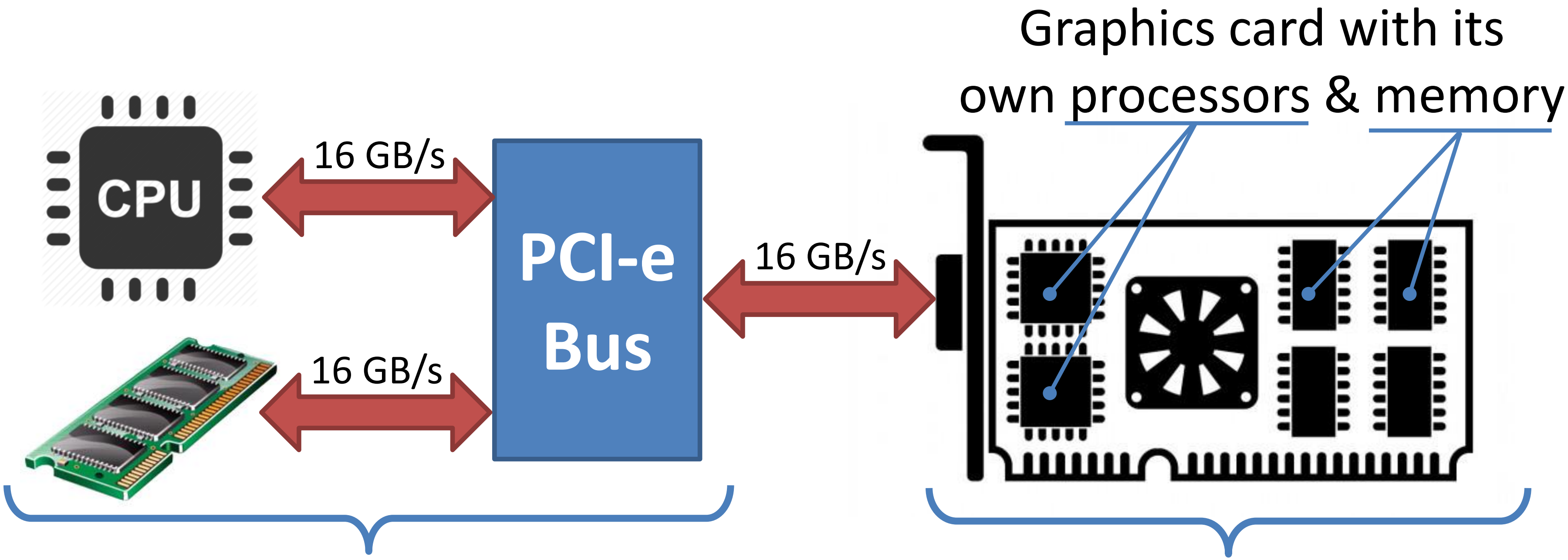
Zipf-Mandelbrot law

$$p(i) * \ (B + i)^{\gamma} = C,$$

$$B, C\ and\ \gamma\ are\ consts$$

*Zipf, Human behavior and the principle of least efforts Mandelbrot, An information theory of the statistical

Graphics card - a powerful computing unit in a PC



Graphics card with its own processors & memory

CPU & RAM:

- 32 cores on CPU
- 16 GB/s bus speed
- 8-16 GB of RAM

~0,6 Teraflops

Graphics card:

- **1536** cores on GPU
- **130 GB/s** bus speed
- **4 GB** of RAM

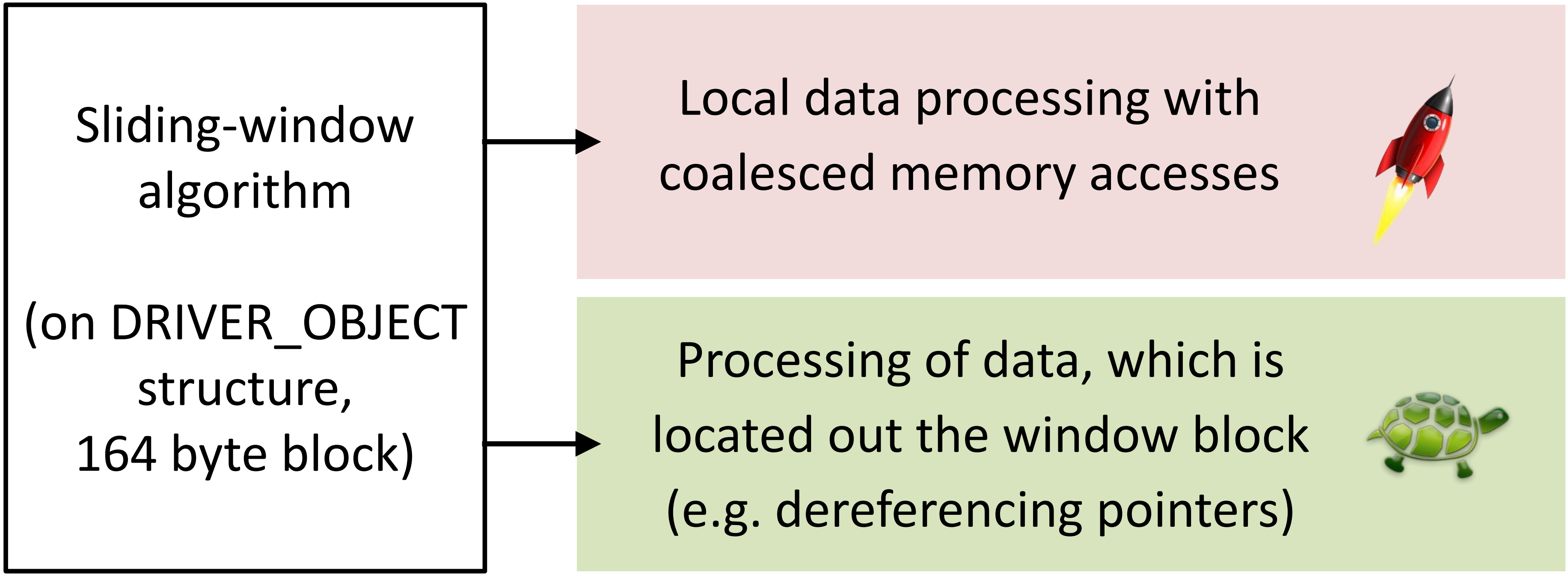
>1 Teraflops

*Intel i7-6785R, launch date Q2'16

*GeForce GTX 980M

Porting issues of common sliding-window algorithm to GPU

- We tested drivers detector from the paper ‘Applying memory forensics to rootkit detection’ ADFSL’2014, Richmond, VA
- GPU works efficiently on 128-byte size coalesced memory
- GPU operates much slower on distinct memory fragments



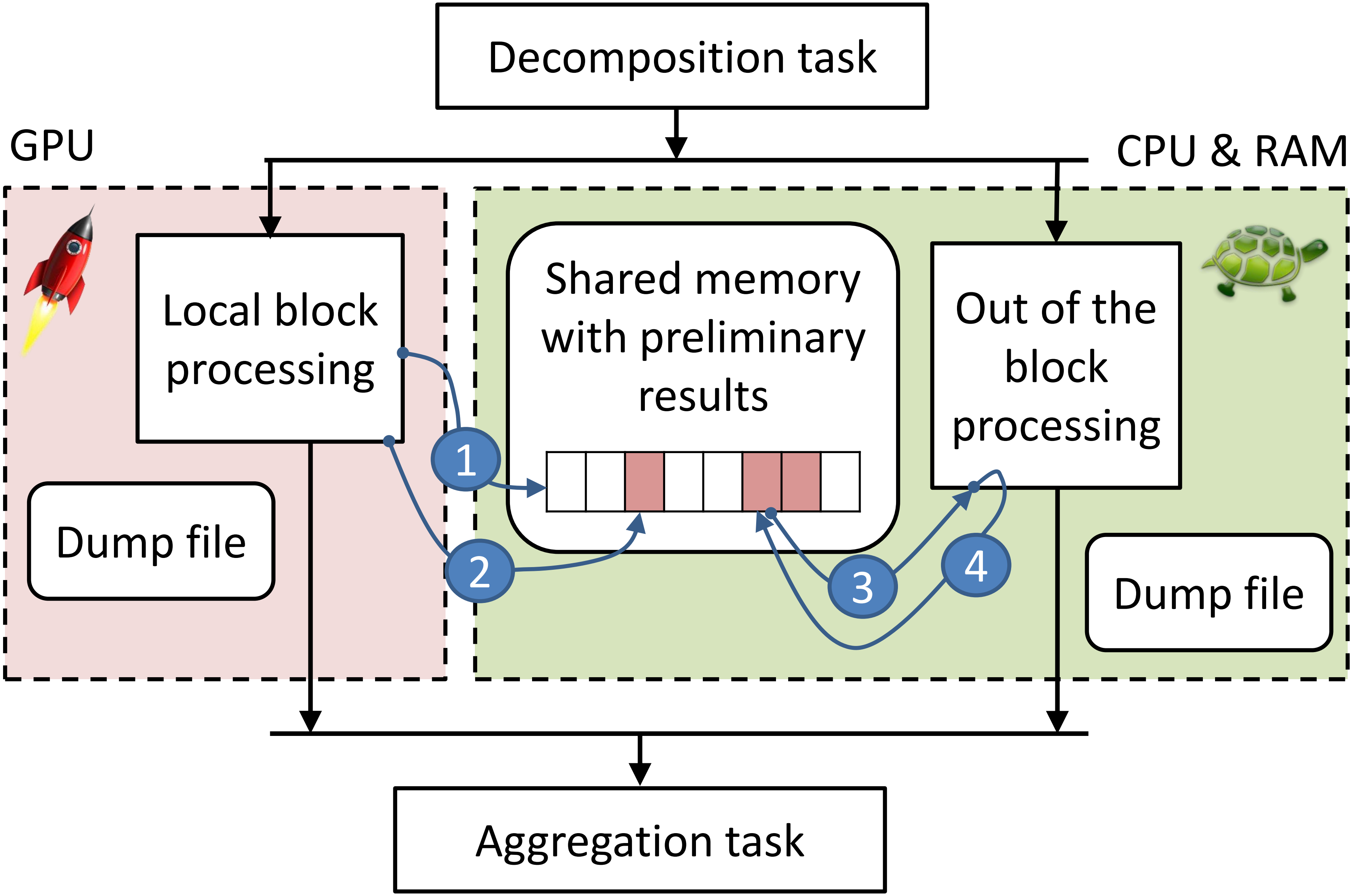
Sliding-window algorithm

(on DRIVER_OBJECT structure, 164 byte block)

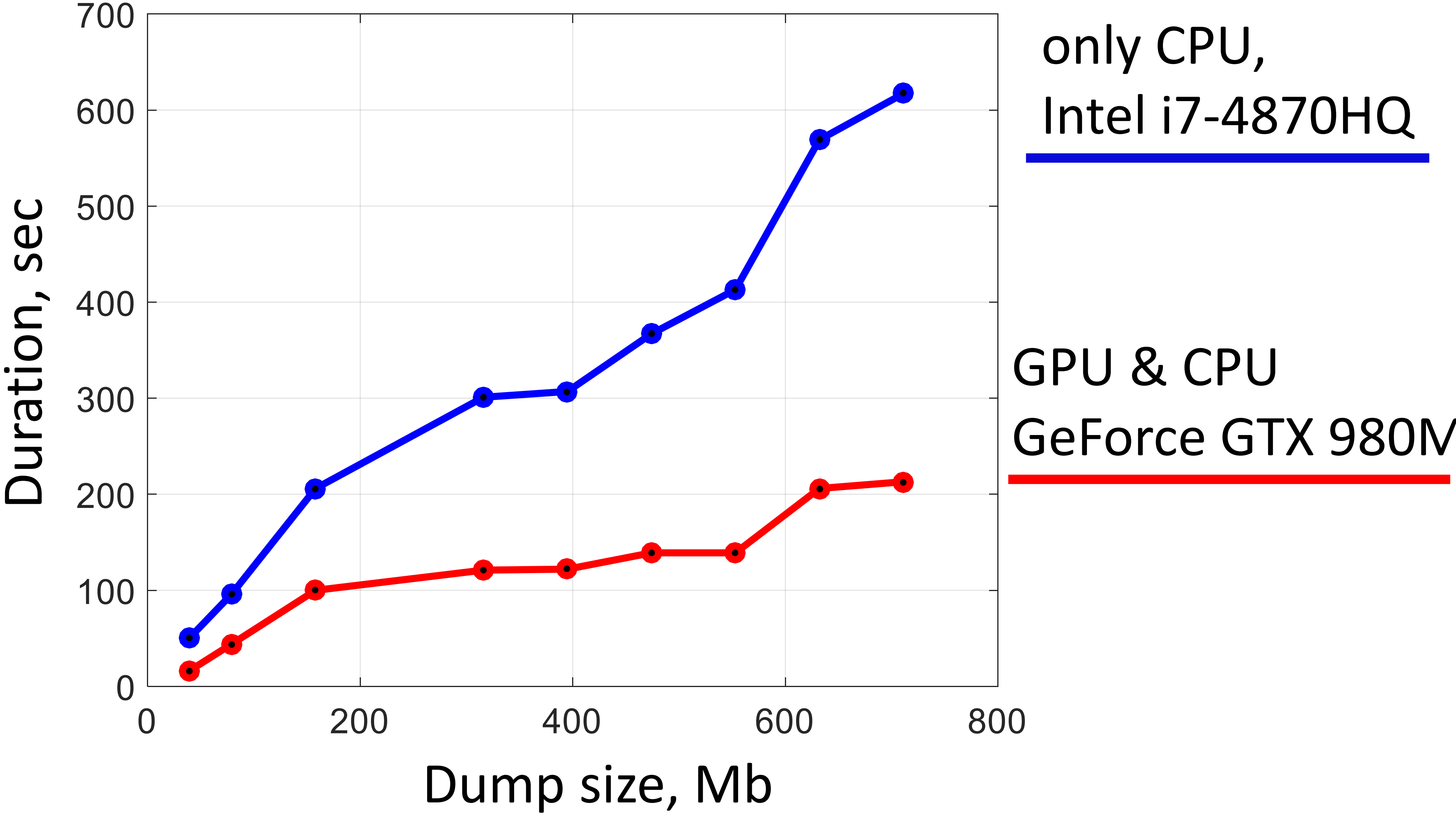
Local data processing with coalesced memory accesses

Processing of data, which is located out the window block (e.g. dereferencing pointers)

Hybrid GPU & CPU architecture for common sliding-window algorithm processing



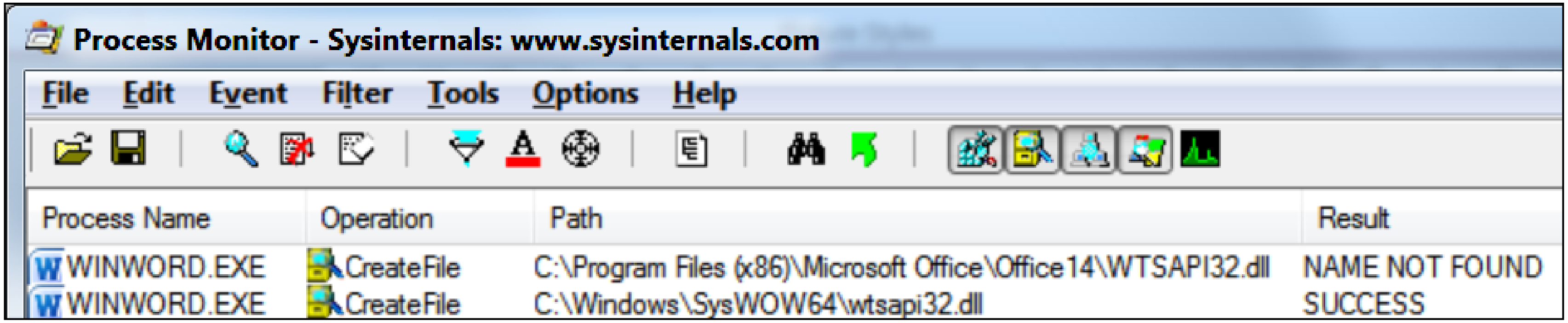
Speeding up memory forensics by CUDA-enabled GPU hardware



Conclusions

- Prototype of the most hidden code – a HighStem
- Ideas to locate executable code
- Using CUDA to speed up memory dump analysis

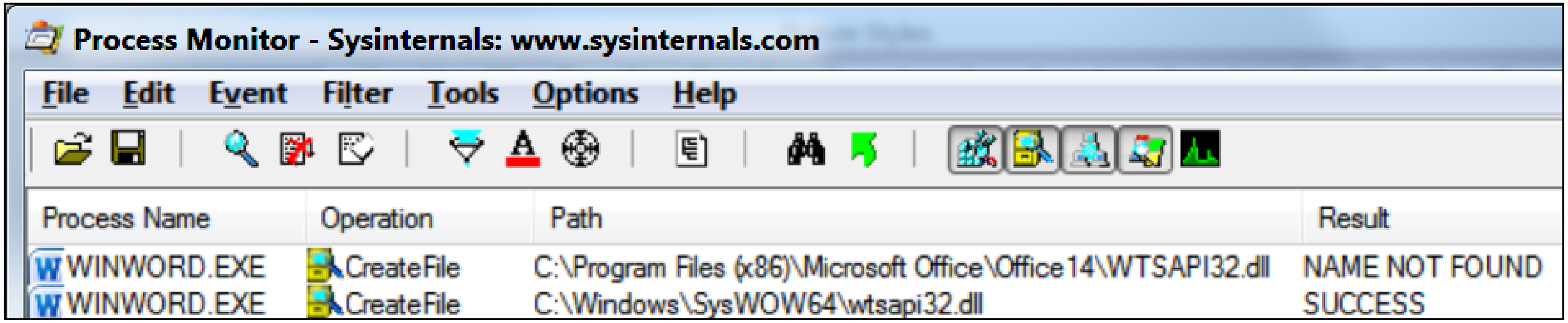
#1: MemoryMon monitors memory changes to track programs activities in real time

- 
Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

Process Name	Operation	Path	Result
WINWORD.EXE	CreateFile	C:\Program Files (x86)\Microsoft Office\Office14\WTSAPI32.dll	NAME NOT FOUND
WINWORD.EXE	CreateFile	C:\Windows\SysWOW64\wtsapi32.dll	SUCCESS

- MemoryMon

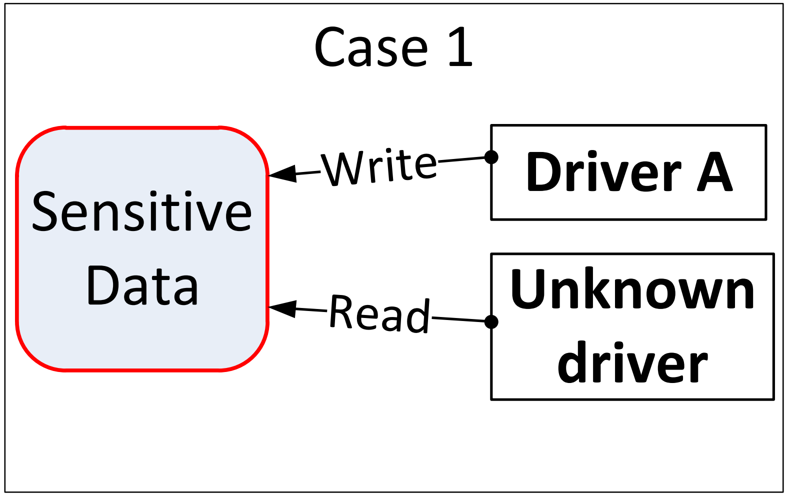
#1: MemoryMon monitors memory changes to track programs activities in real time

- 

Process Monitor - Sysinternals: www.sysinternals.com

Process Name	Operation	Path	Result
WINWORD.EXE	CreateFile	C:\Program Files (x86)\Microsoft Office\Office14\WTSAPI32.dll	NAME NOT FOUND
WINWORD.EXE	CreateFile	C:\Windows\SysWOW64\wtsapi32.dll	SUCCESS

- MemoryMon scenarios:



Case 1

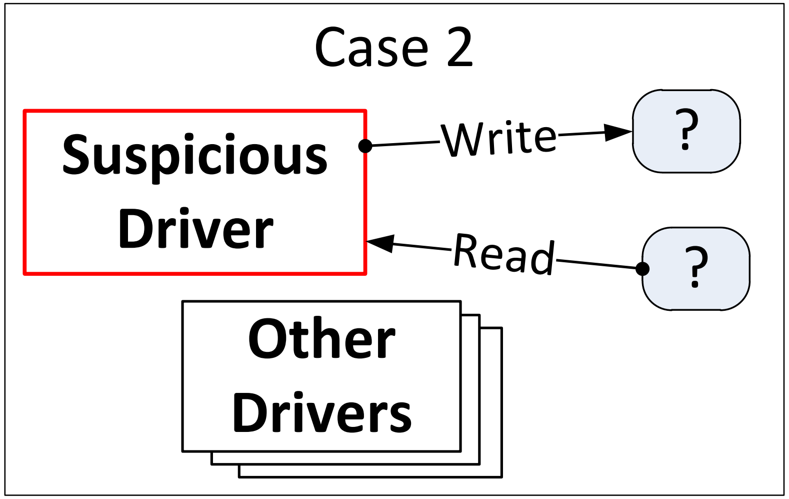
Sensitive Data

Write

Driver A

Read

Unknown driver



Case 2

Suspicious Driver

Write

?

Read

?

Other Drivers

- Details: Monitoring & controlling kernel-mode events by HyperPlatform by Satoshi Tanda @standa_t and Igor Korkin, REcon 2016.

#2: Apply virtual reality headset for digital forensics investigations

by Samsung

by Oculus

‘It's like watching a 130-inch television screen from 10 feet away’*

http://www.pocket-lint.com/news/134352-vuzix-iwear-720-a-vr-headset-that-s-like-putting-a-130-inch-tv-

Thank you!

- igor.korkin@gmail.com

Темы УИРов и дипломов

- Обнаружение уязвимостей программного обеспечения в условиях отсутствия их исходного текста
- Обнаружение скрытого программного обеспечения в мобильных операционных системах
- Создание облачного антируткита и антивируса
- Исследование перспективных технологий с позиции внедрения вредоносного ПО
- Исследование средств удалённого контроля работы компьютерных систем
- Применение технологии аппаратной виртуализации в задачах защиты информации

Подробнее - https://sites.google.com/site/igorkorkin/for-students

Чем ещё заниматься?

- Летние школы Майкрософт, Intel, ШАД:

- Выставки и конференции

- Стажировки – http://www.fulbright.ru/ru

- Изучением английского языка - http://amc.ru/