Defeating Anti-forensics in Contemporary Complex Threats

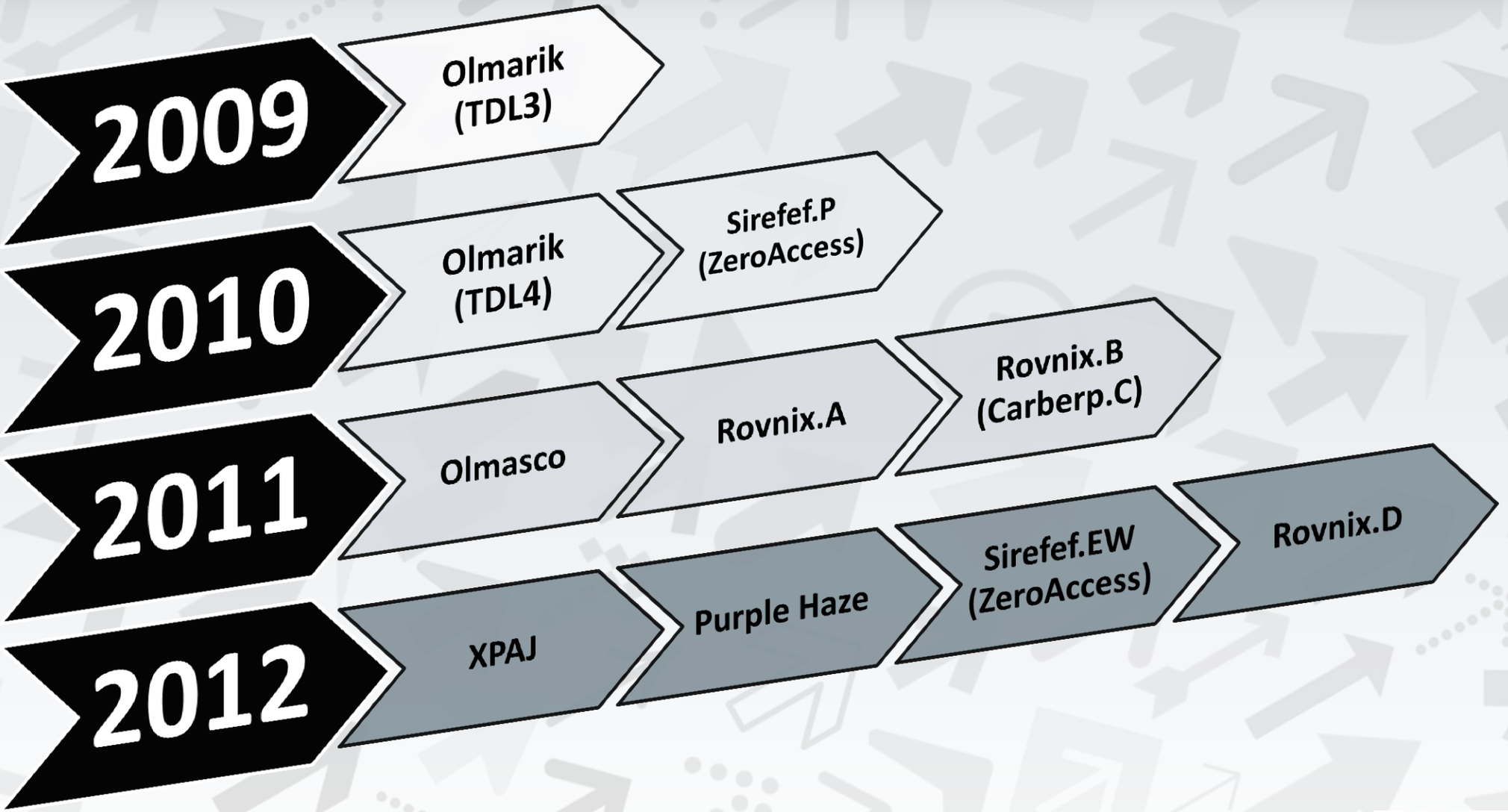
Eugene Rodionov
Aleksandr Matrosov

Outline of The Presentation

- Hidden file systems
  - ✓ Why?
  - ✓ Anti-Forensics approaches
- Hidden storage design principles
  - ✓ Architecture
  - ✓ Hidden file systems in the wild
    - ✓ TDL3/TDL4/Olmasco/Rovnix/Sirefef …
- HiddenFsReader forensic tool

Hidden File Systems

Hidden File Systems Evolution Over Time



Why?

Why do modern complex threats need a hidden FS?

- ✓ Secure storage of components and configuration file(s)
  - bypass standard forensic methods for extraction
- ✓ High level of stealth
  - there are no malicious files in the file system to detect
- ✓ High degree of survivability
  - difficult to detect and remove
- ✓ Ability to disable and bypass security software
  - the malware is launched before security software

Anti-Forensics

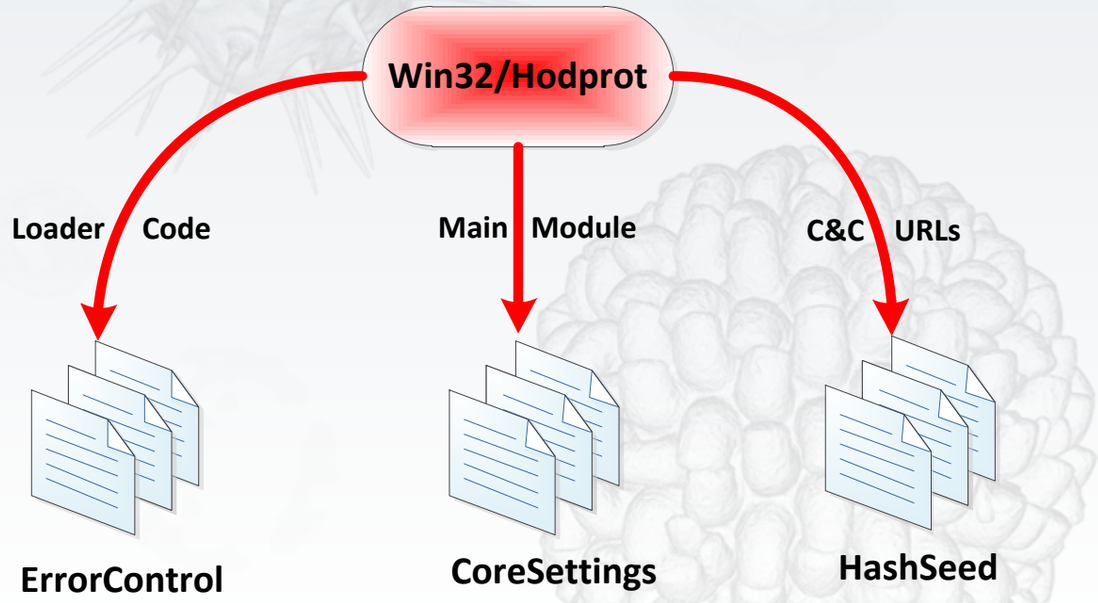
Bypass typical forensic techniques

- Problems:
  - ✓ Malicious files are not stored in the file system (difficult to extract)
  - ✓ Hidden storage cannot be decrypted without malware analysis
  - ✓ Typical forensic tools do not work out of the box
- Solutions:
  - ✓ Malware analysis and reconstruction algorithms
  - ✓ Development of custom tools for extracting hidden content

Anti-Forensics: Hodprot

One of the most frequently used droppers in 2011 for delivering banking trojans in Russia: Carberp, Sheldor, RDPdoor.

Relies on system registry to keep its modules and payload
HKLM\SOFTWARE\Settings

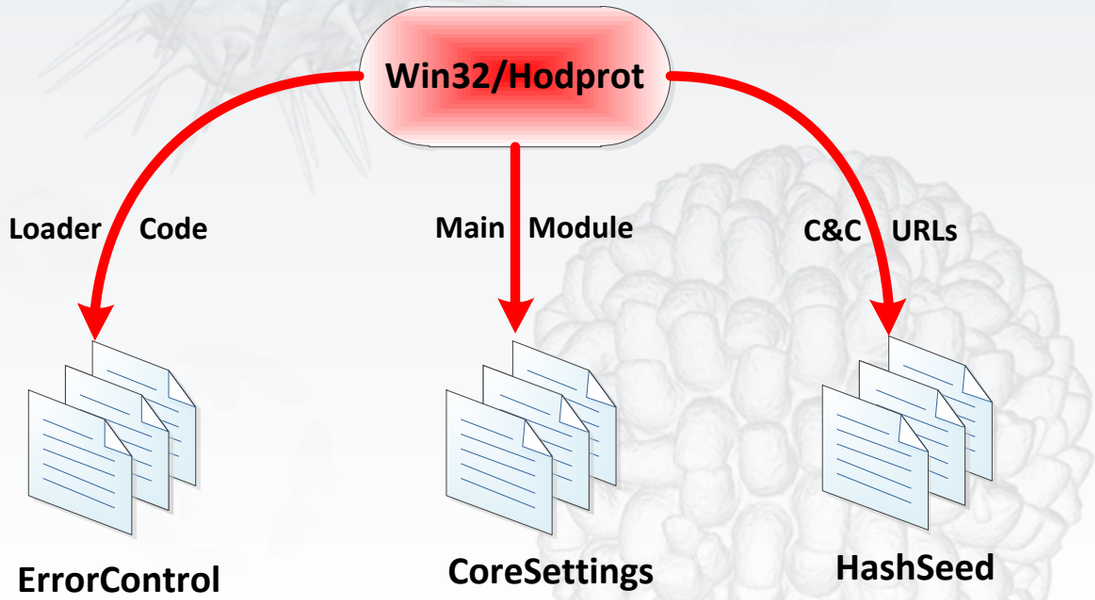


Anti-Forensics: Hodprot

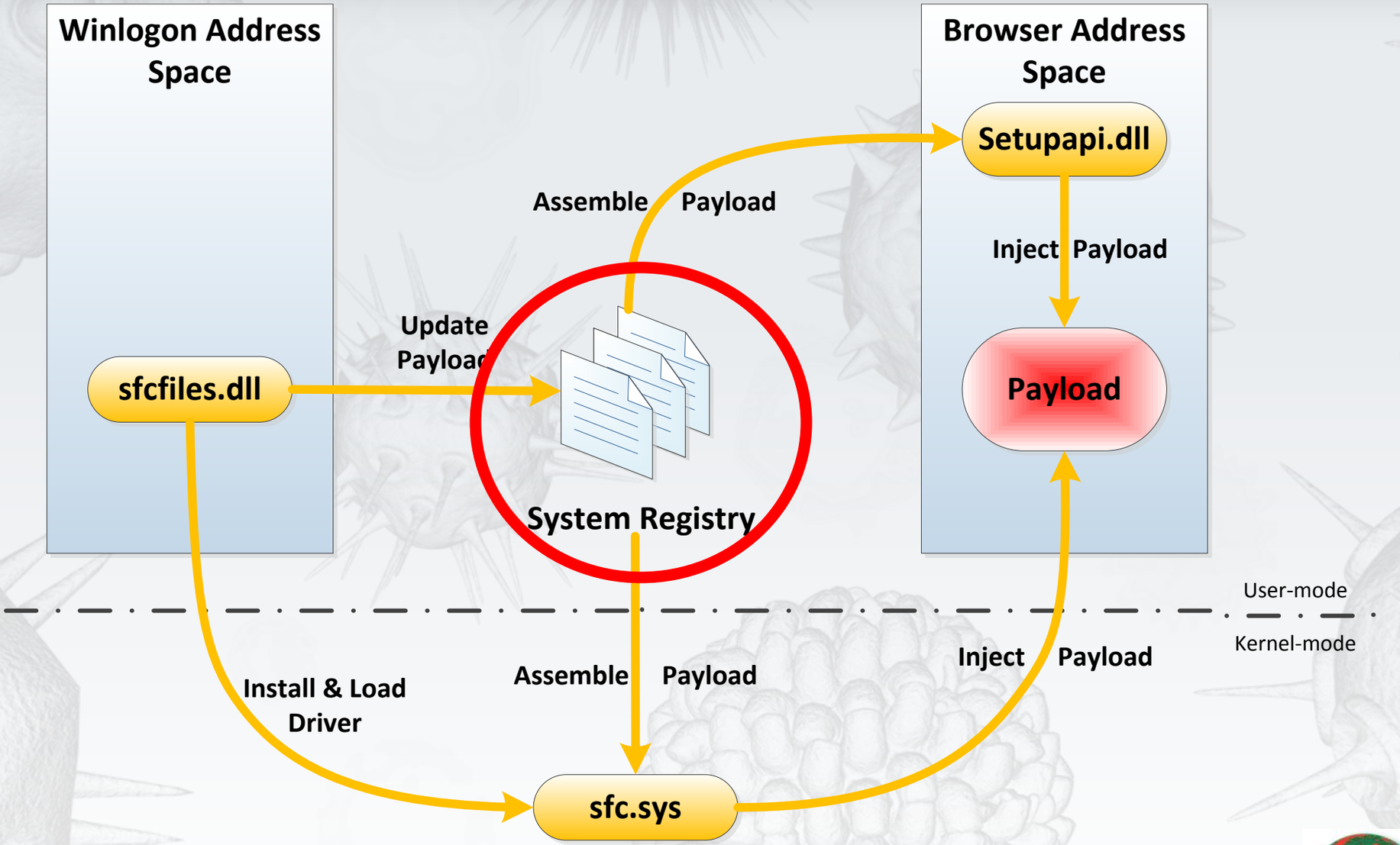
One c
banki

CoreSettings	REG_BINARY	3d 1b 4a a6 4d a5 8d 43 34 a5 43 99 e4 b2 33 46 63 a5 8c 33 38 86 83 b9
CryptoHash	REG_BINARY	3d 1b 4a a6 4d a5 8d 43 34 a5 43 99 e4 b2 33 46 63 a5 8c 33 38 86 83 b9
DriveSettings	REG_BINARY	3d 1b 4a a6 4d a5 8d 43 34 a5 43 99 e4 b2 33 46 63 a5 8c 33 38 86 83 b9
ErrorControl	REG_BINARY	ab a3 52 84 c9 87 63 61 12 e1 61 64 1b 4d 33 46 8f 5a 91 63 b1 66 1b 43
HashSeed	REG_BINARY	cb a2 31 c8 0e 57 ad 9a 4b 26 32 19 c0 a7 d3 79 0e 06 3f 8b 62 e2 51 b8

Relies
HKLM



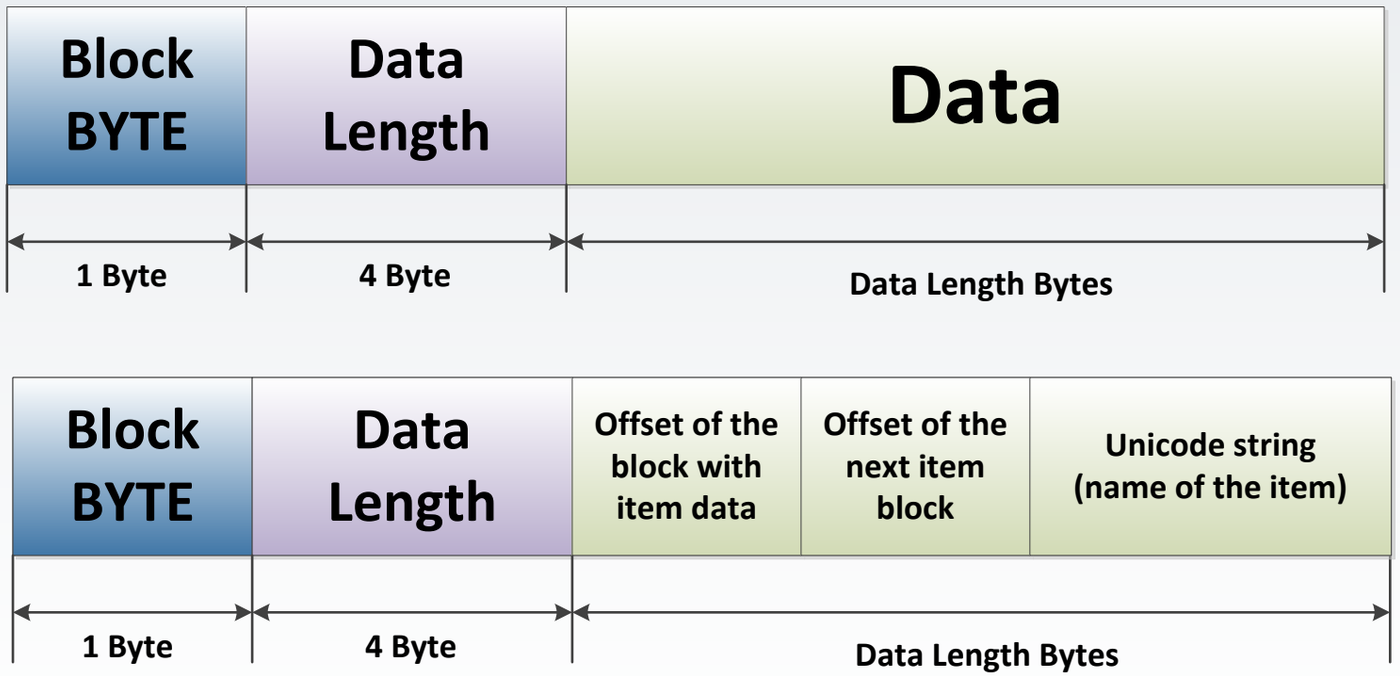
Anti-Forensics: Hodprot



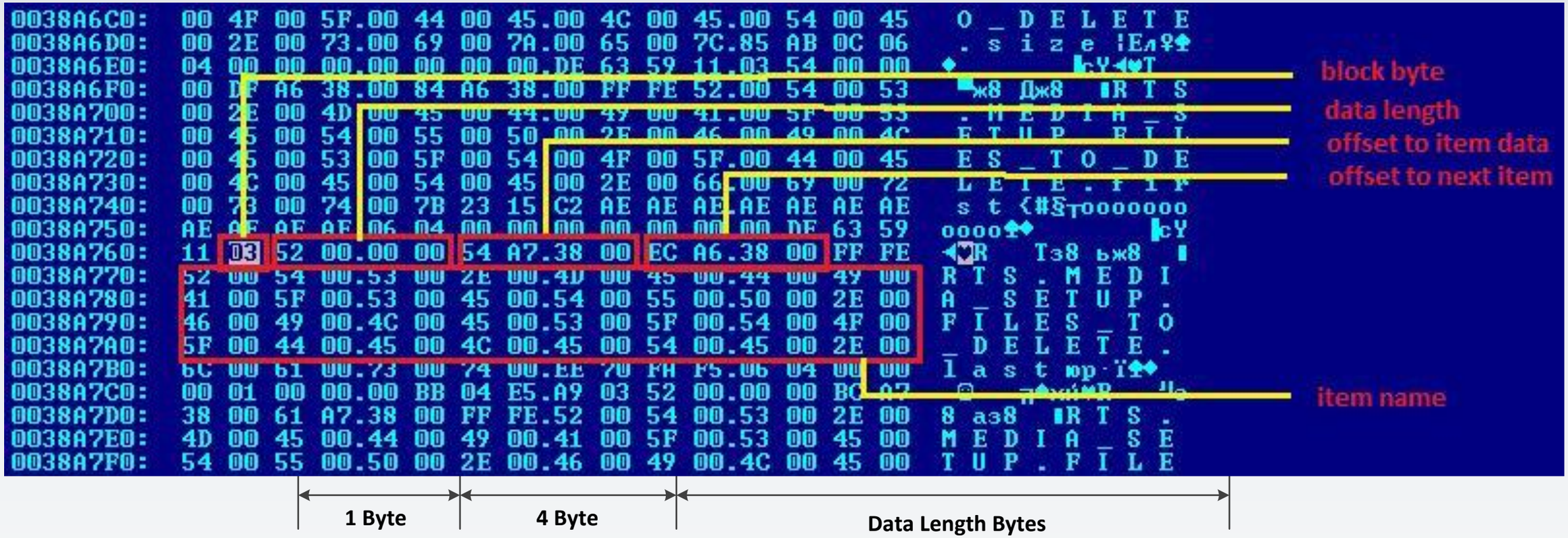
Flame configuration data storage

Configuration data are stored in a resource of the Flame main module

Configuration data are encrypted with custom algorithm and compressed with *DEFLATE* algorithm



Flame configuration data storage

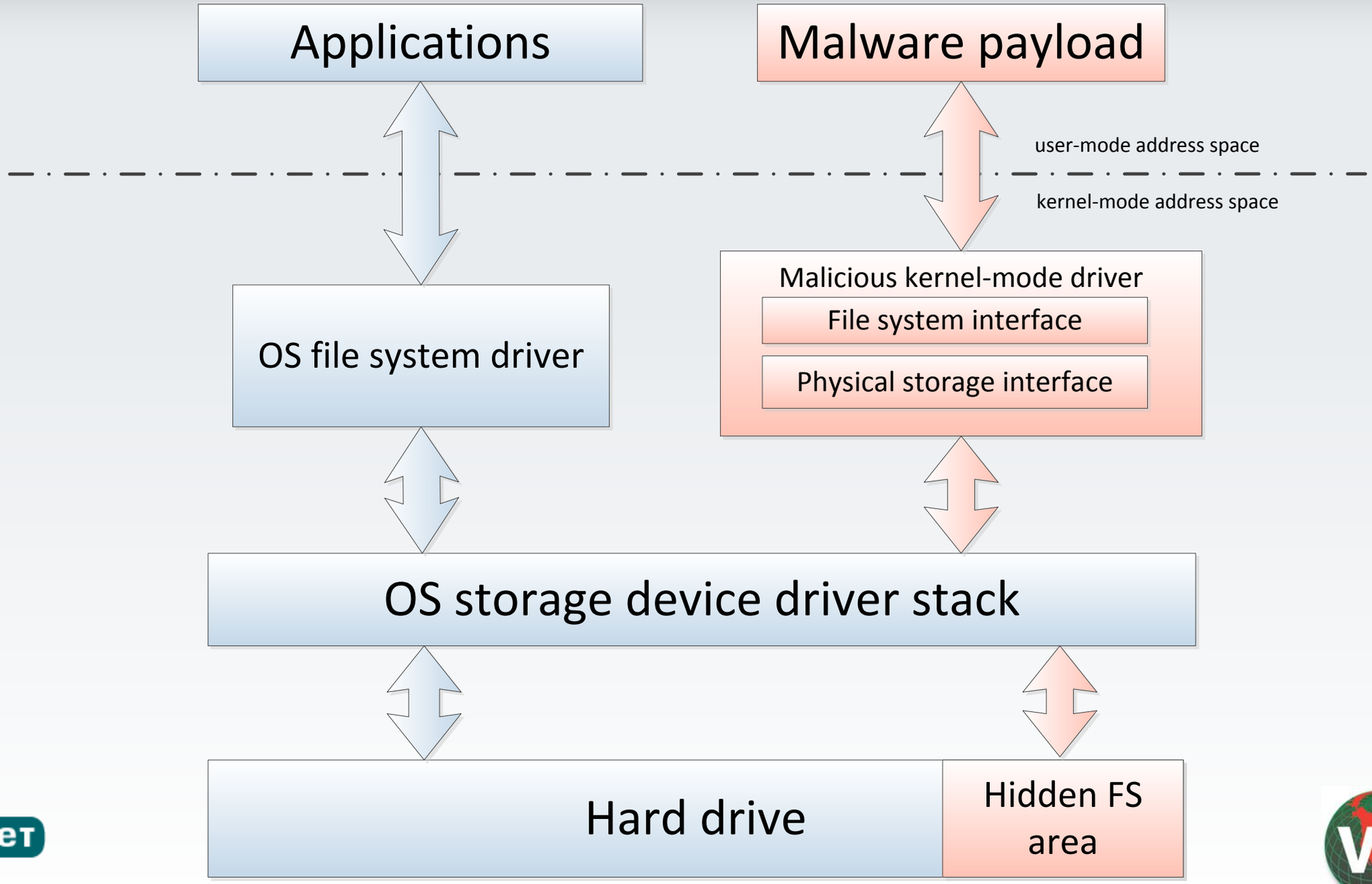


Block BYTE	Data Length	Offset of the block with item data	Offset of the next item block	Unicode string (name of the item)

1 Byte | 4 Byte | Data Length Bytes

Hidden storage design principles

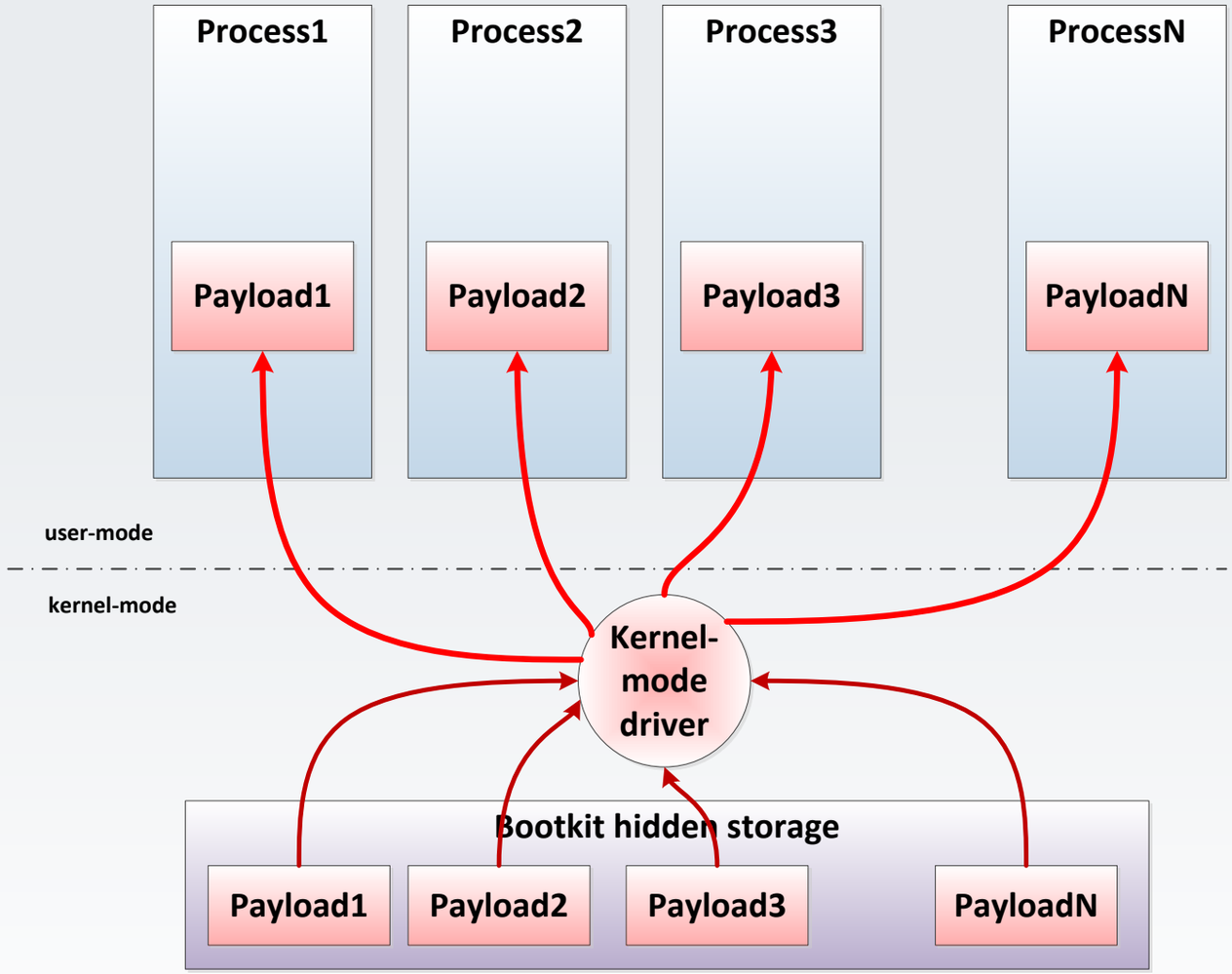
Anti-Forensics: Hidden Storage Architecture



Injecting Payload

Injection approach

- ✓ APC routines
- ✓ Patching entry point of the executable



Hidden file systems in the wild

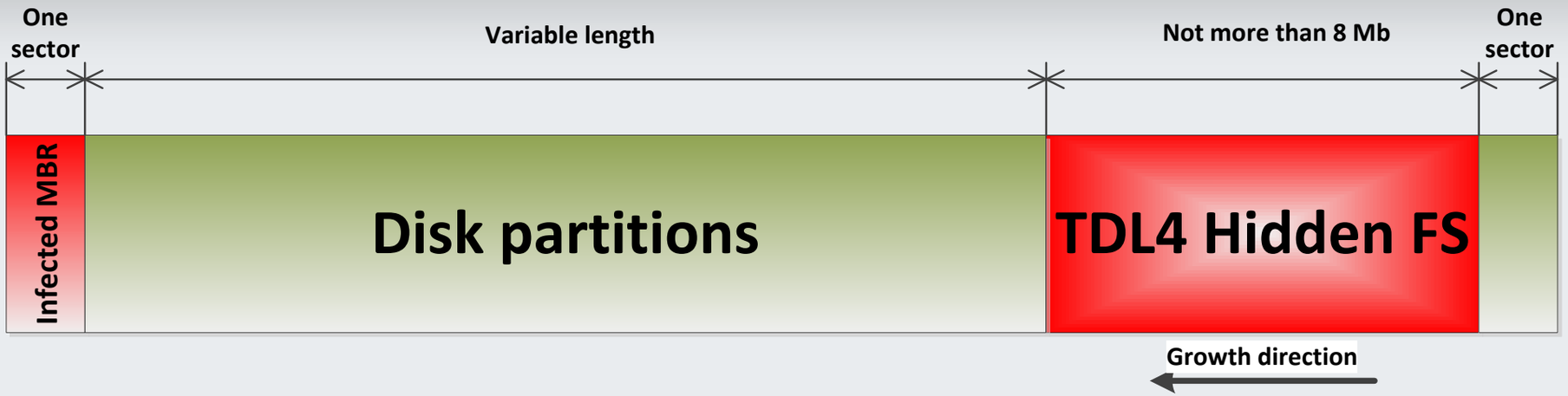
eset

vb 2012 DALLAS 26 - 28 September 2012

Olmarik (TDL4)

- First widely spread bootkit targeting Microsoft Windows x64 platform
- Infects MBR of the bootable hard drive to receive control before OS
- Abuses Windows PE (Preinstallation Environment) boot mode to disable kernel-mode code signing policy
- Keeps payload in the hidden file system
- Hijacks pointer to the driver object of the lowest device object in storage device stack

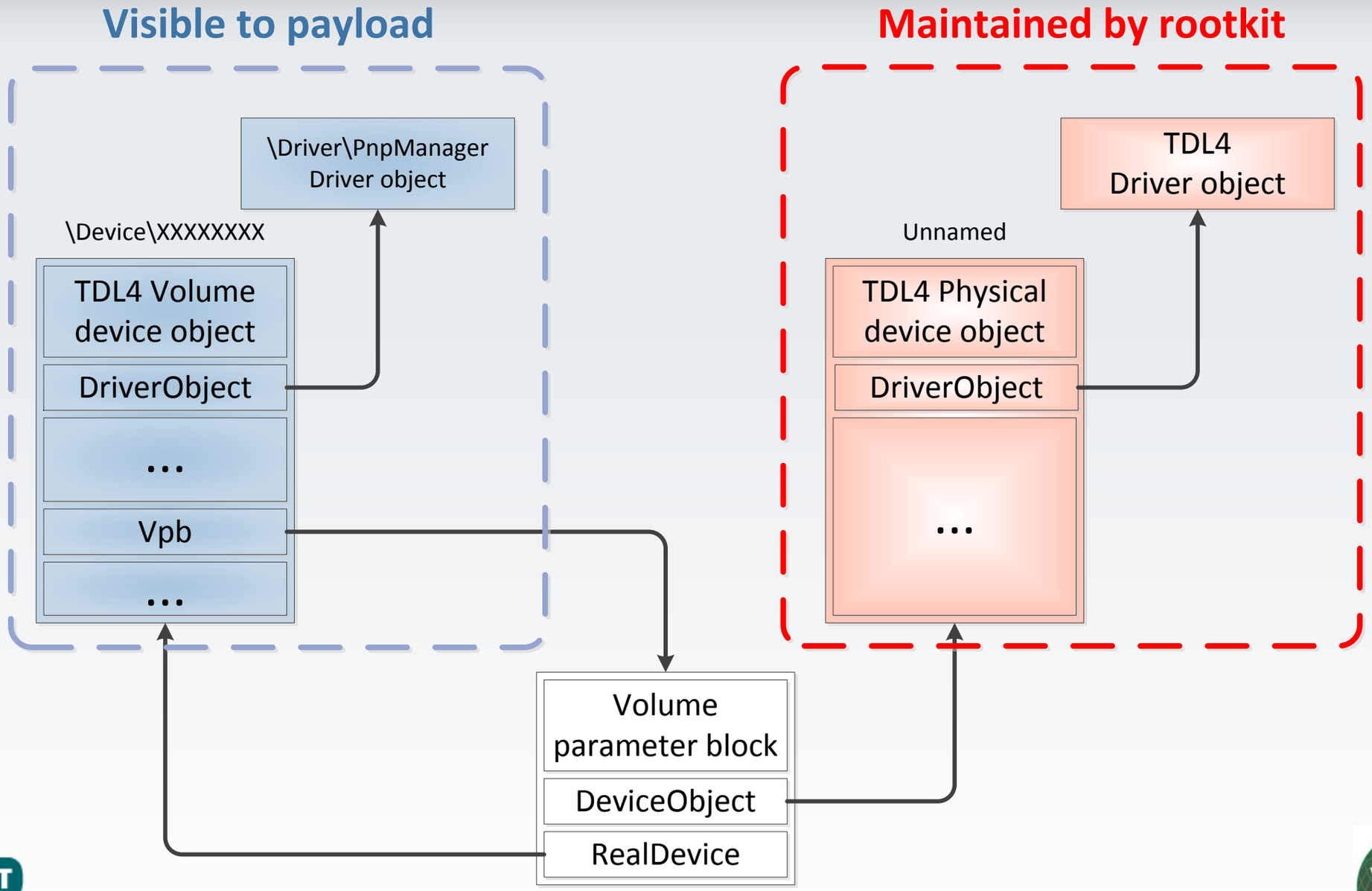
Olmarik (TDL4): Hidden File System Layout




```
typedef struct _TDL4_FS_ROOT_DIRECTORY
{
  // Signature of the block
  WORD Signature;
  // Set to zero
  DWORD Reserved;
  // Array of entries corresponding to files in FS
  TDL4_FS_FILE_ENTRY FileTable[15];
}TDL4_FS_ROOT_DIRECTORY, *PTDL4_FS_ROOT_DIRECTORY;
```

```
typedef struct _TDL4_FS_FILE_ENTRY
{
  // File name - null terminated string
  char FileName[16];
  // Offset from beginning of the file system to file
  DWORD FileBlockOffset;
  // Reserved
  DWORD dwFileSize;
  // Time and Date of file creation
  FILETIME CreateTime;
}TDL4_FS_FILE_ENTRY, *PTDL4_FS_FILE_ENTRY;
```

Olmarik (TDL4): Hidden Storage Implementation

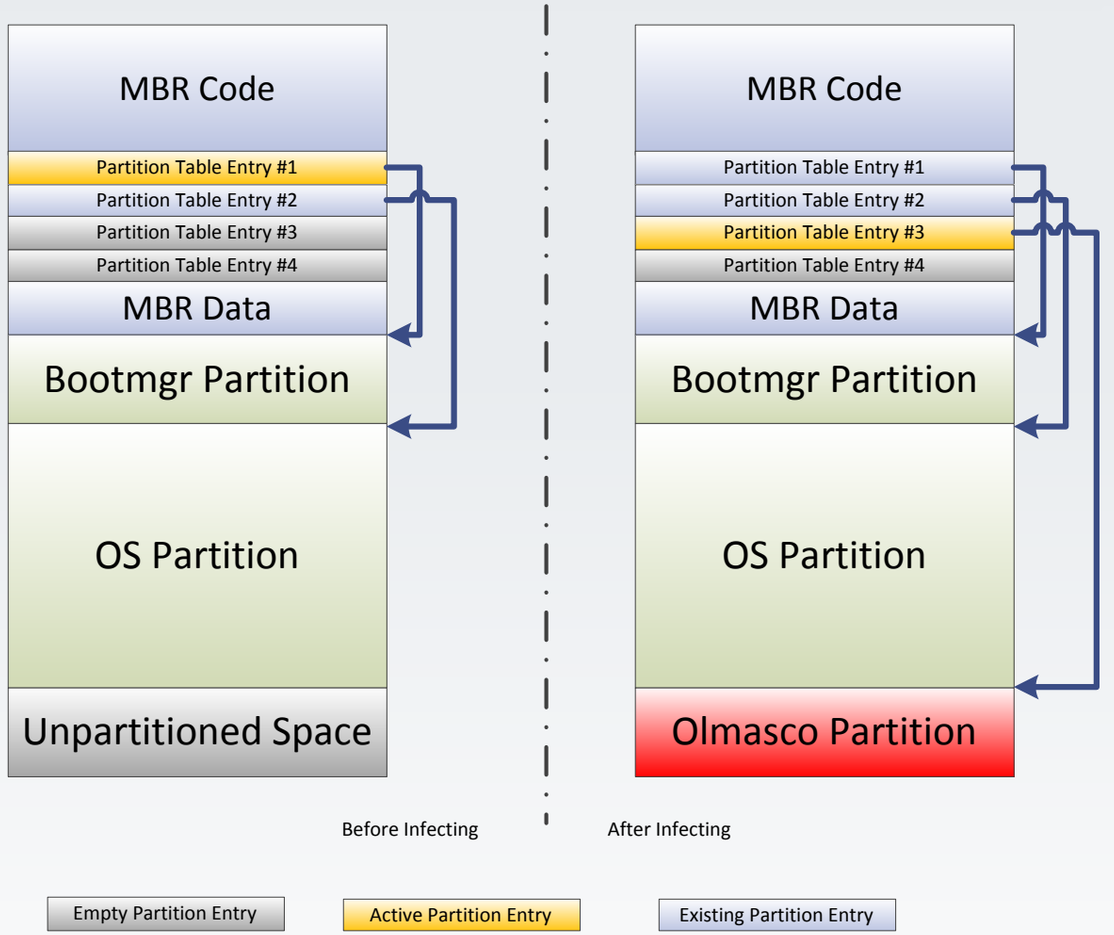


XXXXXXXX – random 32-bit hexadecimal integer

Olmasco (MaxSS)

Employs the same approach for disabling kernel-mode signing policy as TDL4 bootkit

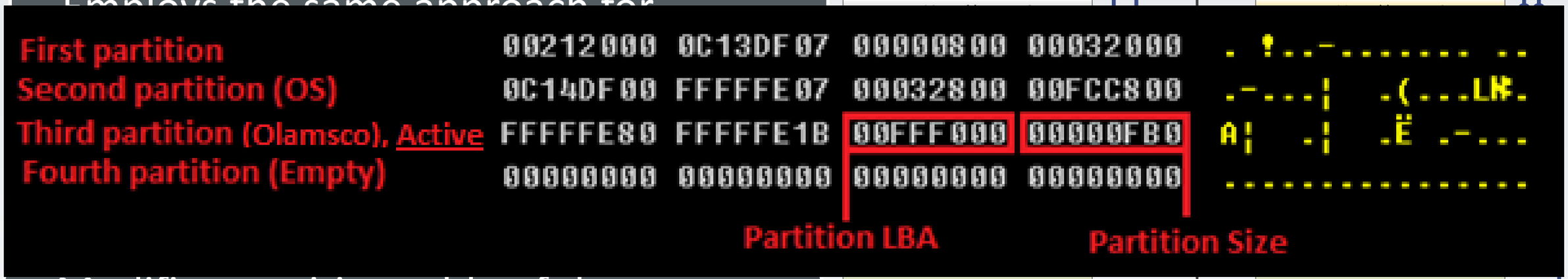
Modifies partition table of the bootable hard drive to create malicious partition and mark it active

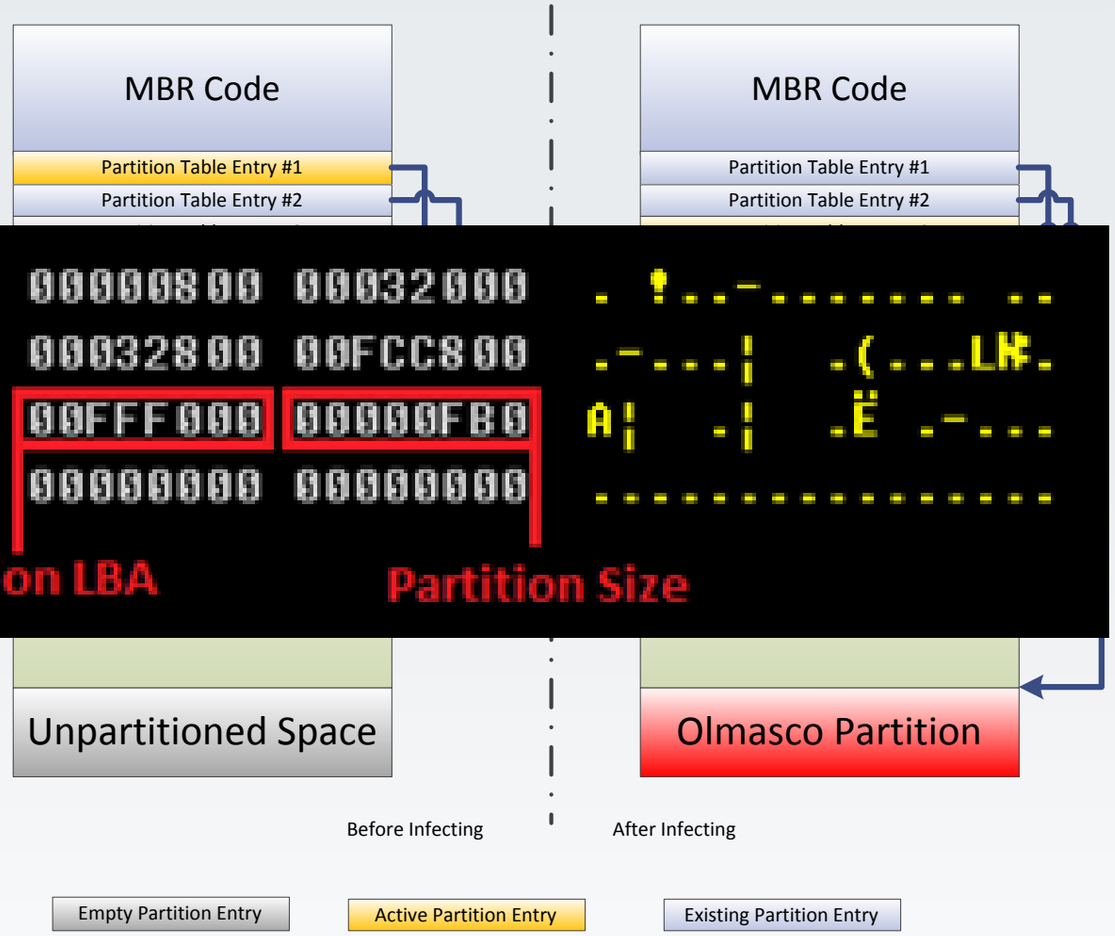


Olmasco (MaxSS)

Employs the same approach for

Modifies partition table of the bootable hard drive to create malicious partition and mark it active





Olmasco (MaxSS): Hidden File System Layout

```
typedef struct _OLMASCO_FS_ROOT_DIRECTORY
{
  // Signature of the block
  // DC - root directory
  DWORD Signature;
  // Set to zero
  DWORD Reserved1;
  // Set to zero
  DWORD Reserved2;
  // Set to zero
  DWORD Reserved3;
  // Size of the file system cluster
  DWORD ClusterSize;
  // Size of file table in clusters
  DWORD SizeOfSysTableInClusters;
  // Size of file table in bytes
  DWORD SizeOfSysTableInBytes;
  // Checksum of file table
  DWORD SysTableCRC32;
  // Array of entries corresponding to files in FS
  OLMASCO_FS_FILE_ENTRY FileTable[];
}OLMASCO_FS_ROOT_DIRECTORY, *POLMASCO_FS_ROOT_DIRECTORY;
```

```
typedef struct _OLMASCO_FS_FILE_ENTRY
{
  // File name - null terminated string
  char FileName[16];
  // Offset from beginning of the file system to file
  DWORD OffsetInClusters;
  // Size of the file in clusters
  DWORD SizeInClusters;
  // Size of the file in bytes
  DWORD SizeInBytes;
  // Checksum
  DWORD Crc32;
}OLMASCO_FS_FILE_ENTRY, *POLMASCO_FS_FILE_ENTRY;
```

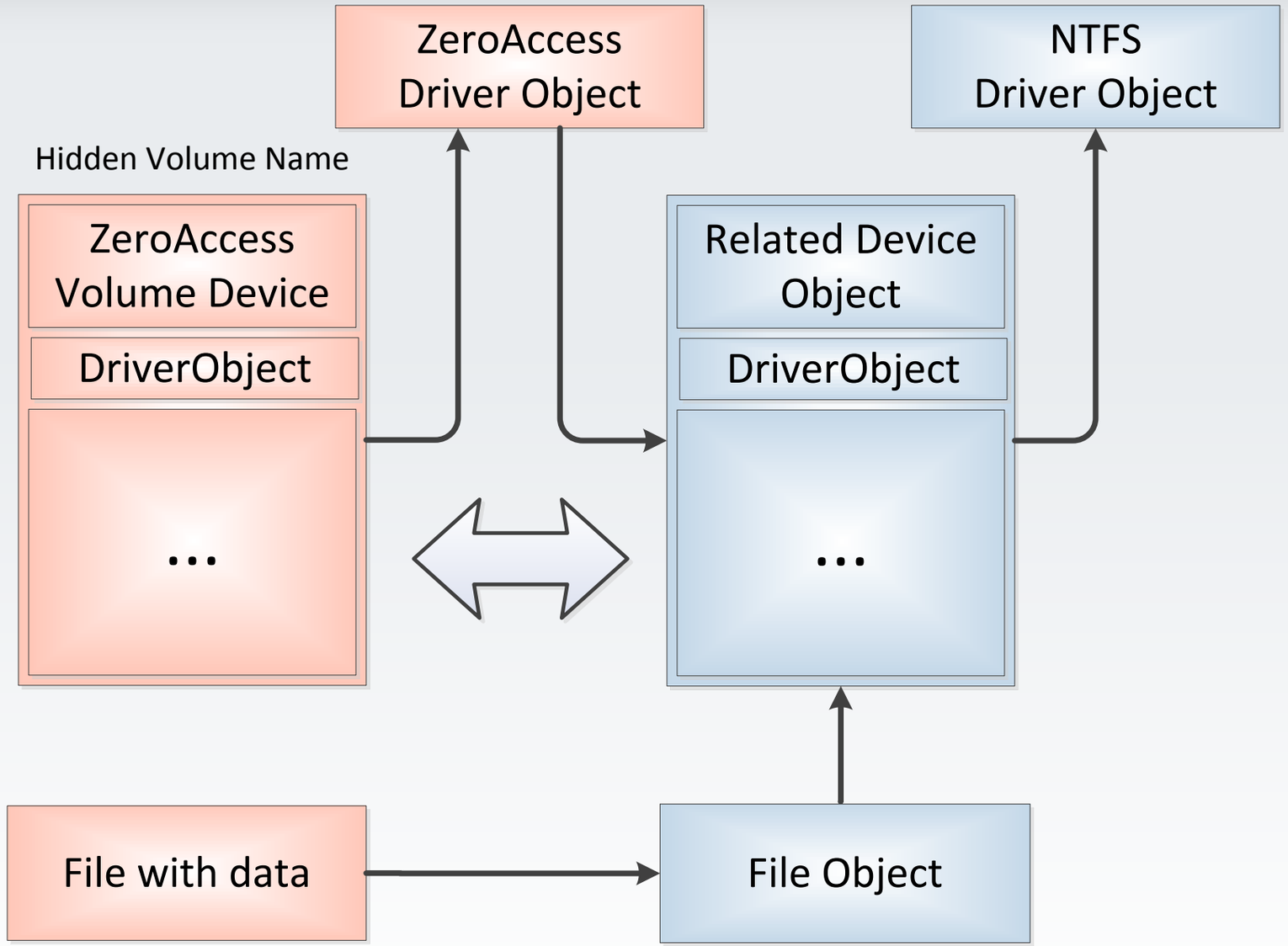
Sirefef (ZeroAccess)

Mounts a file containing payload & configuration data as NTFS volume with transparent encryption

VS.

Keeps payload & configuration data encrypted in *“C:\windows\$NtUninstallKB35373$”* directory

ZeroAccess: Hidden Storage Implementation



Rovnix (Cidox)

First known bootkit which infects VBR (Volume Boot Record) with polymorphic malicious bootstrap code

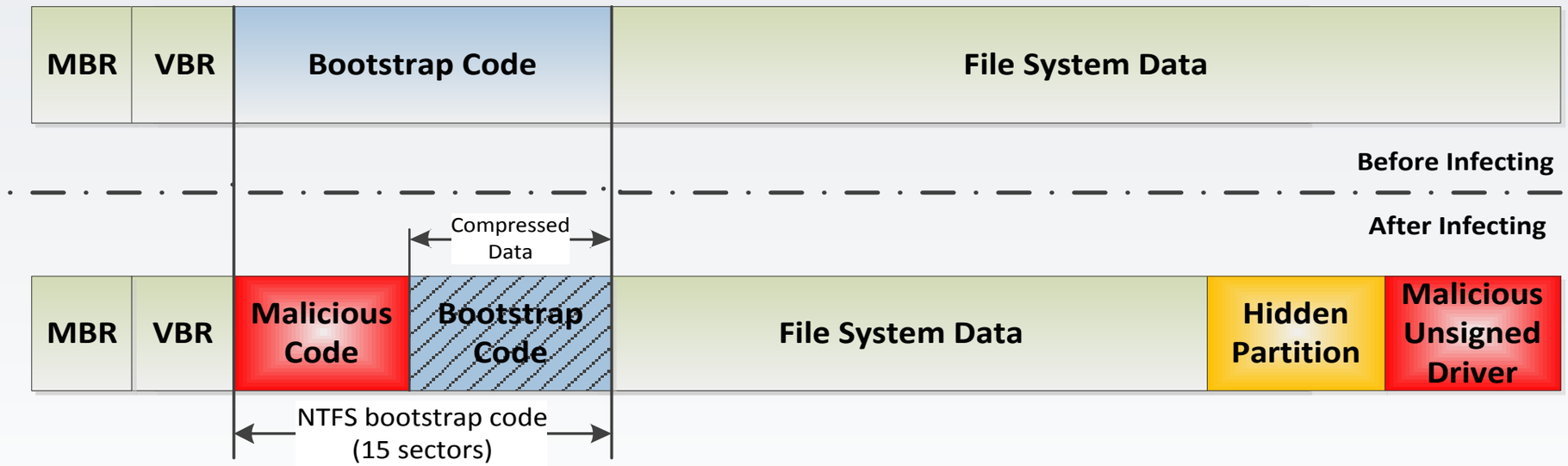
It utilizes debugging facilities of the hardware (debugging registers) to persists among processor execution mode switching and set up hooks

Rovnix bootkit employs modification of FAT16 for hidden partition

Rovnix bootkit was used in Carberp banking trojan

Rovnix (Cidox): Hidden Storage Layout

- Hidden partition & kernel-mode driver are written either:
  - ✓ before first partition on the disk – if there are more than 2,000 (1 MB) free sectors
  - ✓ otherwise, at the end of the hard drive



Goblin (XPAJ)

- Employs advanced bootkit techniques to load unsigned malicious kernel-mode driver on 64-bit version of Windows OS
- Combines bootkit techniques present in Olmarik (TDL4) & Rovnix
- Bypasses PatchGuard by means of modifying kernel before integrity enforcement service is started
- Goblin hidden implements hidden file system in a similar way to Olmarik (TDL4)

Hidden File Systems Comparison

functionality	Olmarik (TDL4)	Sirefef (ZeroAccess)	Rovnix (Cidox)	Goblin (XPAJ)	Olmasco (MaxSS)
MBR modification	☑	☑	☒	☑	☑
VBR modification	☒	☒	☑	☒	☒
Hidden file system type	Custom	NTFS	FAT16 modification	Custom (TDL4 based)	Custom
encryption algorithm	XOR/RC4	RC4	Custom (XOR+ROL)	☒	RC6 modification
compression algorithm	☒	☒	aPlib	aPlib	☒

HiddenFsReader forensic tool

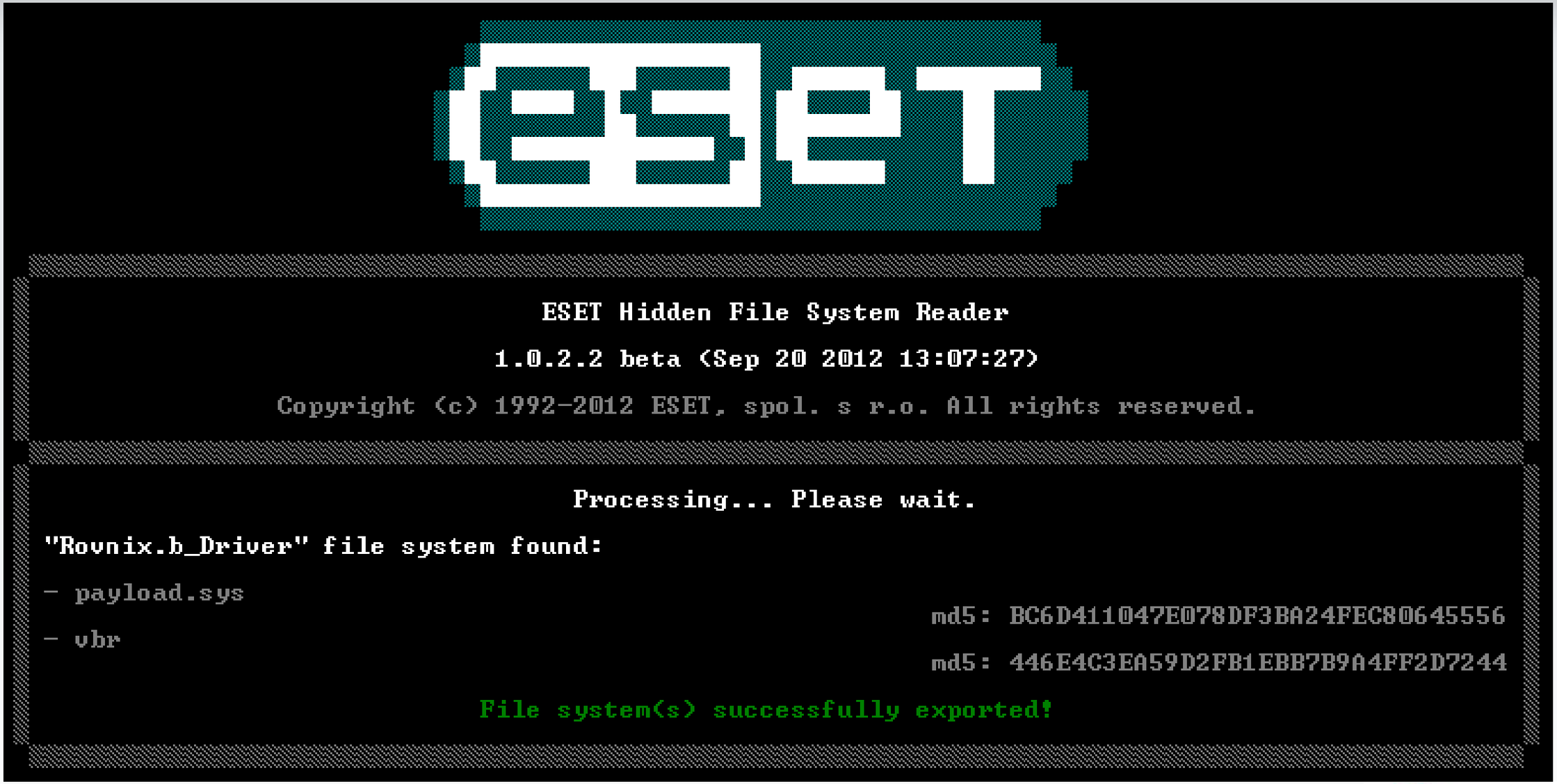
HiddenFsReader

HiddenFsReader: The evolution of TdlFsReader

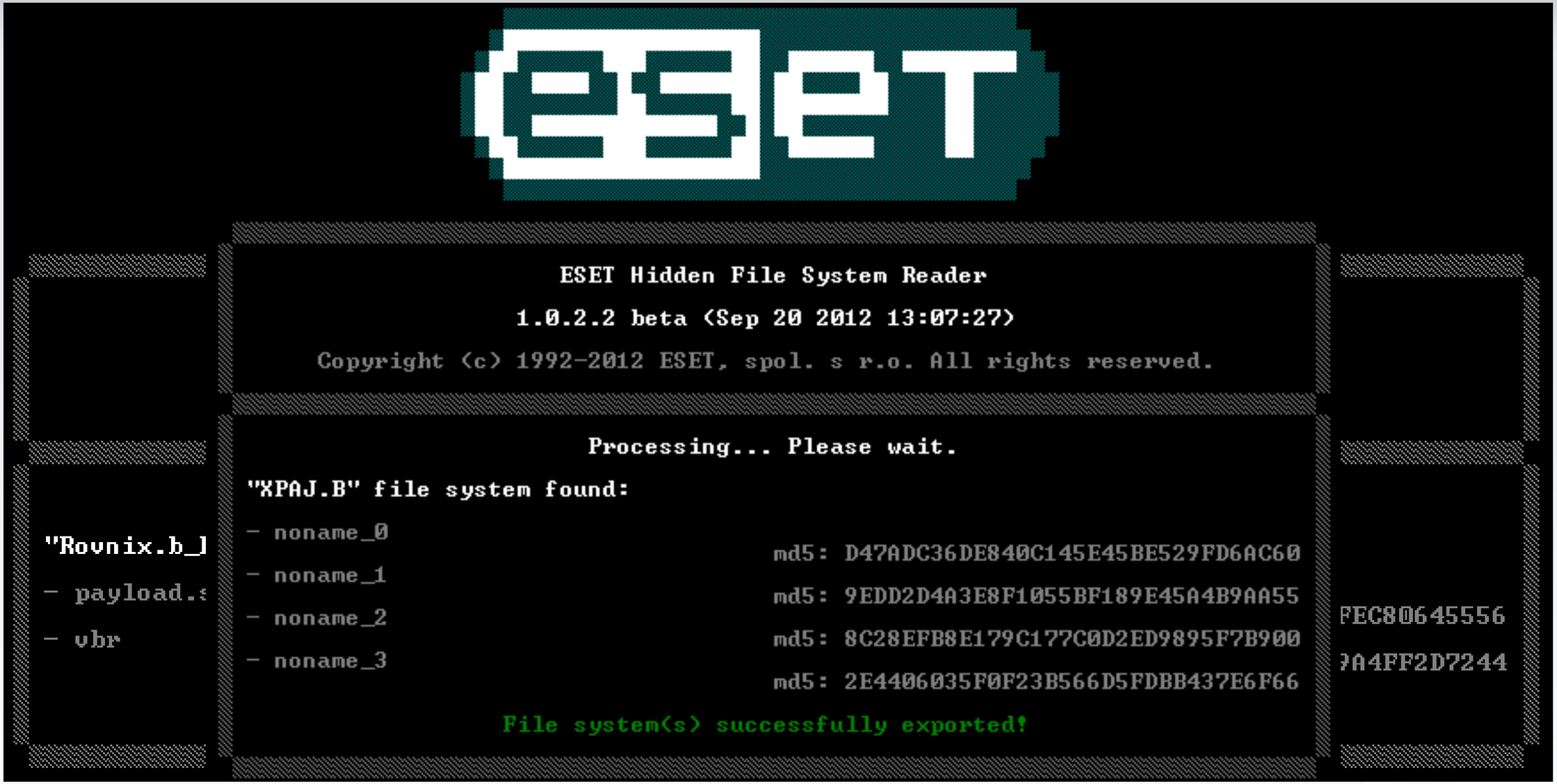
- Supports the following hidden file systems:
  - ✓ TDL3 and modifications
  - ✓ TDL4 and modifications
  - ✓ Olmasco
  - ✓ Rovnix.A
  - ✓ Rovnix.B
  - ✓ Sirefef (ZeroAccess)
  - ✓ Goblin (XPAJ)
  - ✓ Flame (dump decrypted resource section)

HiddenFsReader - free public tool

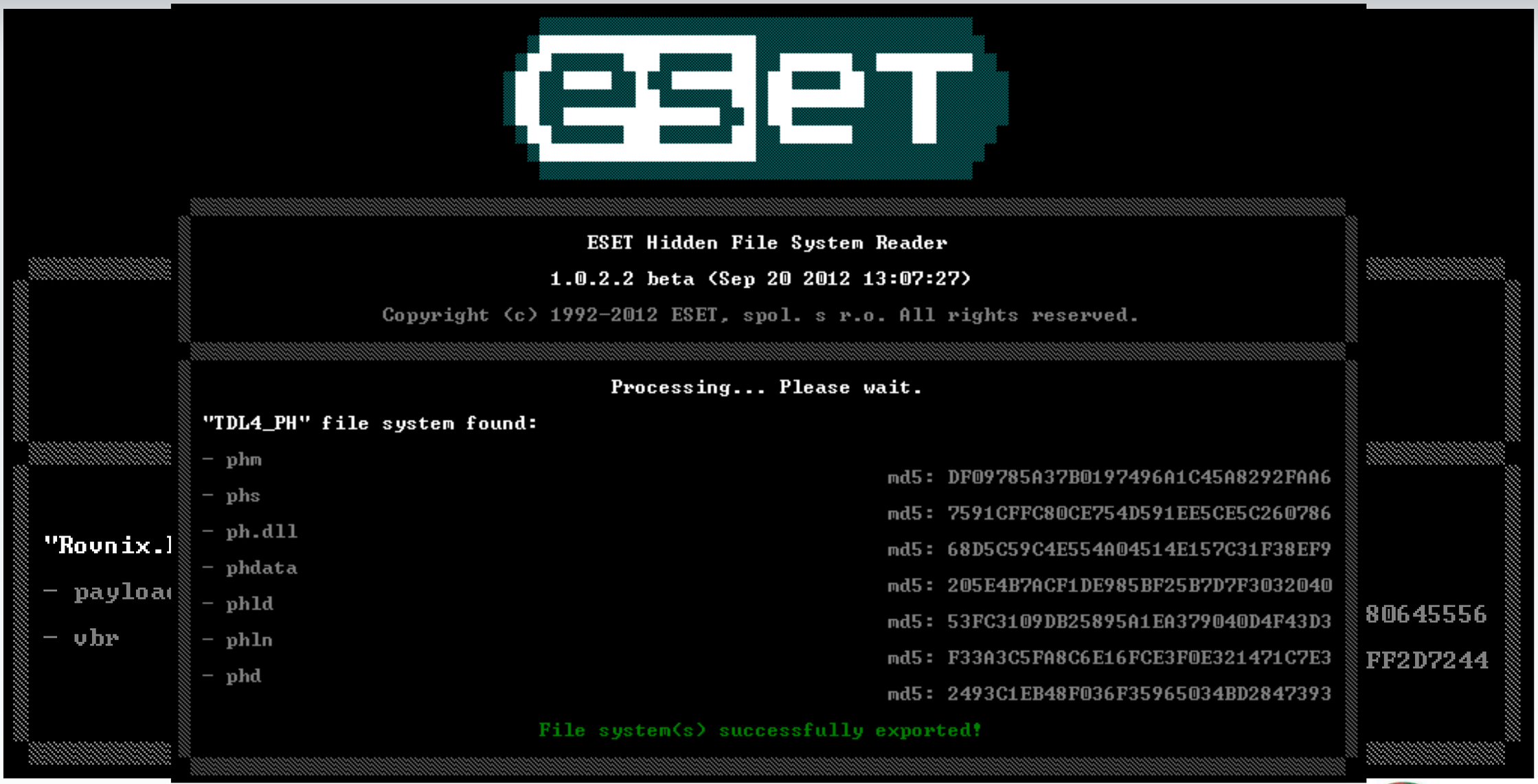
Hidden File System Reader



Hidden File System Reader



Hidden File System Reader



HiddenFsReader: using scenarios

Incident response

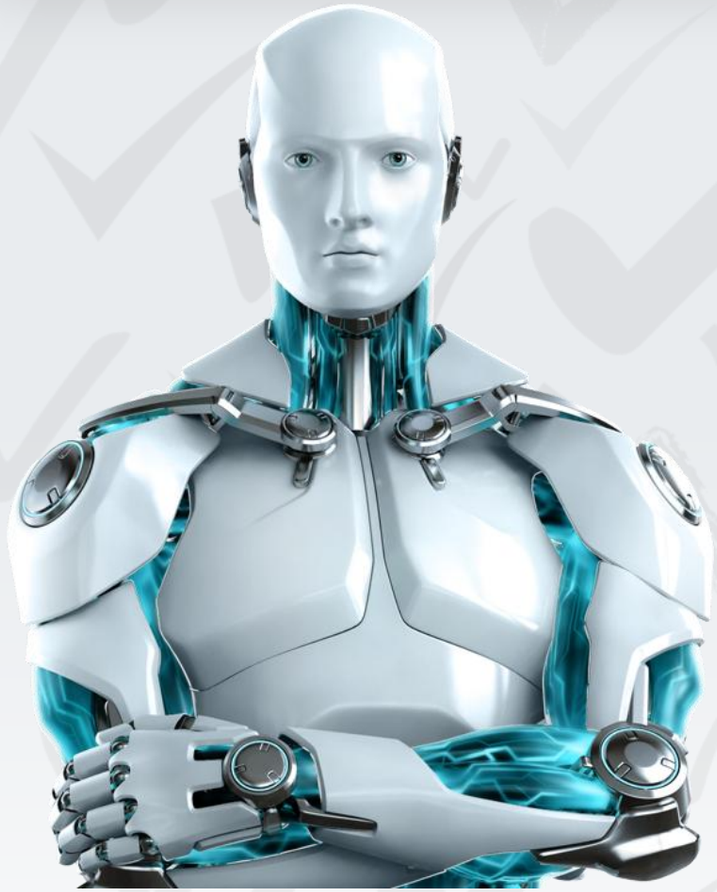
- Dump and decrypt hidden file system
- Dump MBR/VBR or any range of sectors

Threat analysis and monitoring

- Quick dump of payload from hidden file system
- Developing botnet monitoring tracker

LIVE DEMO

HiddenFsReader: Free public forensic tool



Try to use it right now ;)

eset.com/download/utilities/detail/family/173/

Download

ESET

vb 2012 DALLAS 26 - 28 September 2012

Conclusion

- Implementing techniques to counteracting forensic analysis is a common feature of complex threats
- The most well-known threats implementing hidden file systems:
  - ✓ Olmarik (TDL4)
  - ✓ Olamsco (MaxSS)
  - ✓ Rovnix (Cidox)
  - ✓ Goblin (XPAJ)
  - ✓ Sirefef (ZeroAccess)
- HiddenFsReader is a tool that makes it possible to retrieve contents of the most widely spread hidden file systems

References

- ✓ Rovnix Reloaded: new step of evolution

http://blog.eset.com/2012/02/22/rovnix-reloaded-new-step-of-evolution

- ✓ TDL4 reloaded: Purple Haze all in my brain

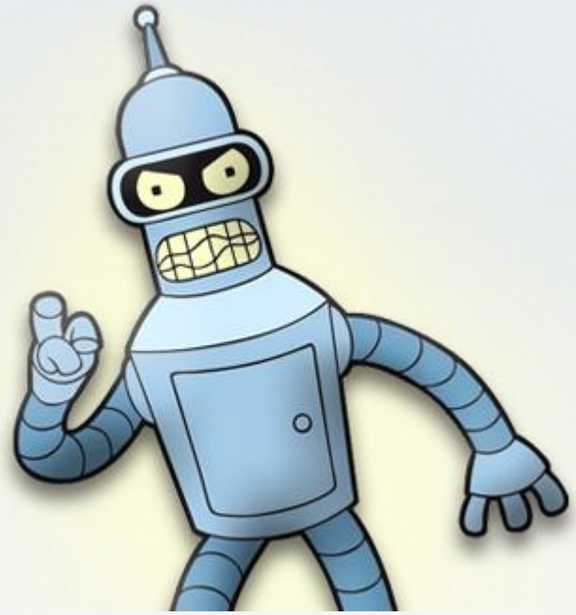
http://blog.eset.com/2012/02/02/tdl4-reloaded-purple-haze-all-in-my-brain

- ✓ Bootkit Threat Evolution in 2011

http://blog.eset.com/2012/01/03/bootkit-threat-evolution-in-2011-2

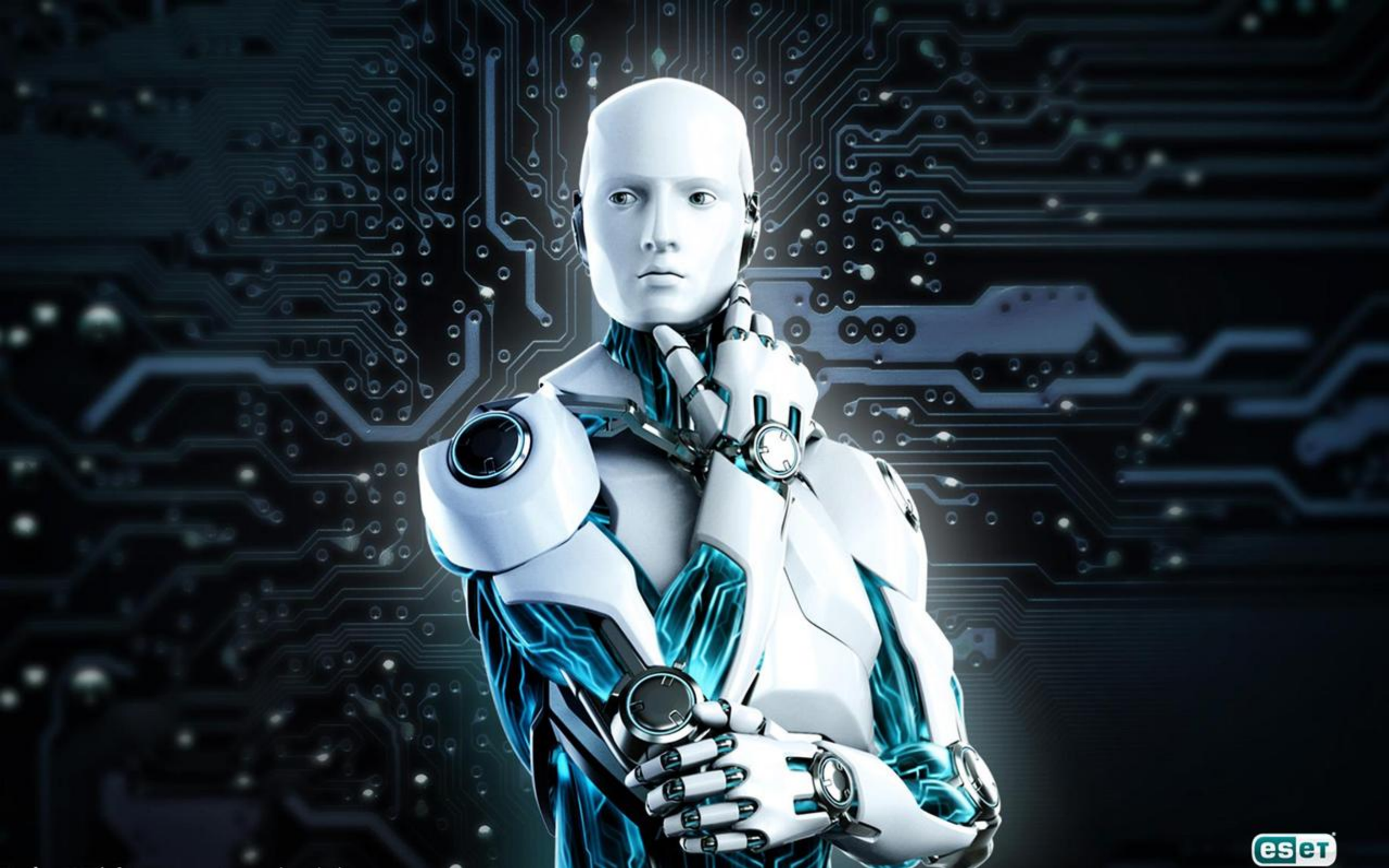
- ✓ Rovnix.D: the code injection story

http://blog.eset.com/2012/07/27/rovnix-d-the-code-injection-story



- ✓ Modern bootkit trends: bypassing kernel-mode signing policy

http://www.virusbtn.com/conference/vb2011/abstracts/LastMinute1.xml


eset

Thank you for your attention!

Aleksandr Matrosov
matrosov@eset.sk
@matrosov

Eugene Rodionov
rodionov@eset.sk
@vxradius

vb 2012 DALLAS 26 - 28 September 2012