Forensics, Malware, and Penetration Testing

Ransomware – Case study: Petya (and WannaCry)

Mihai Ordean
University of Birmingham

m.ordean@cs.bham.ac.uk

Ransomware

- **Ransomware:** a type of malicious software that threatens to publish the victim's data or perpetually block access to it unless a ransom is paid.

Ransomware

- Ransomware:
  - Specific behaviour: modern ransomware makes use of cryptography to prevent users from accessing their data.
  - Deploy vectors: ransomware often has additional malicious behaviour that allows it to behave like a:
    - Virus - infect a host that will trigger the malicious behaviour.
    - Worm - a ransom-worm that propagate in a network using known/0-day vulnerabilities (WannaCry).
    - Trojan – ransomware can be packaged together with other programs that can have “useful” behaviour.

Ransomware-as-a-Service

- Ransomware-as-a-Service model
  - The ransomware “product” is offered “on demand”.
  - The ransomware has build in “protection mechanisms” that do not allow the unauthorized use.
  - Targets criminals who lack the skills, resources or inclination to develop their own malware.

Case-study: The Petya Trojan

- The Petya Trojan (May 2016)

> "… Petya ransomware that not only encrypts data stored on a computer, but also overwrites the hard disk drive's master boot record (MBR), leaving infected computers unable to boot into the operating system …"
>
> Kaspersky Lab

- Petya targeted German speaking victims

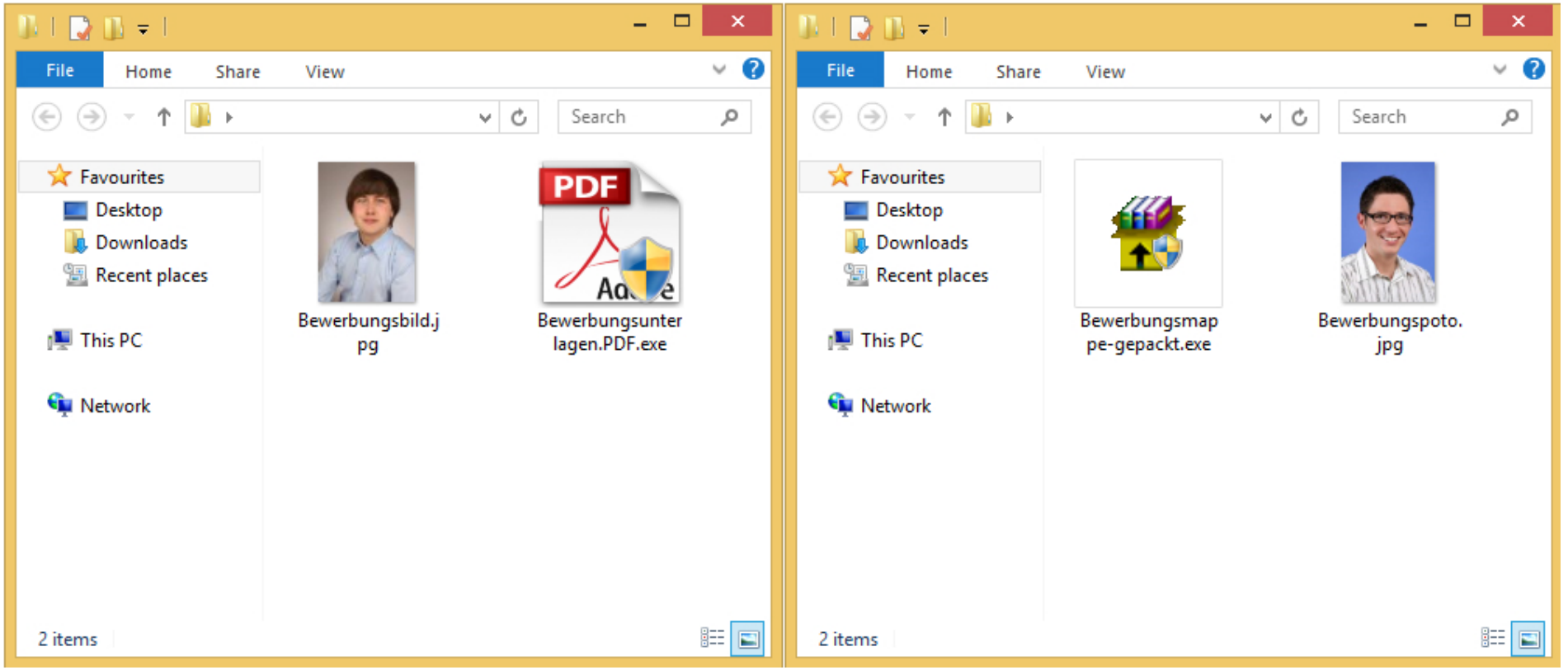
Ref: https://securelist.com/petya-the-two-in-one-trojan/74609/

Case-study: The Petya Trojan

- The Petya Trojan
  - Uses a **Trojan** behaviour to infect the system
  - Is a **boot sector infector** because it infects the boot sector
  - Is a **ransomware** because it demands payment to restore access to the system

Petya infection scenario

Spam email: potential victims are asked to download a ZIP archive



https://s3-eu-west-1.amazonaws.com/khub-media/wp-content/uploads/sites/43/2016/04/07192913/petya_eng_1.jpg

Petya infection scenario

- Spam email: potential victims are asked to download a ZIP archive

- The Trojan does not exploit any **automatic escalation of privileges**
  - Running the executable will show the standard **Windows UAC request for privilege**.

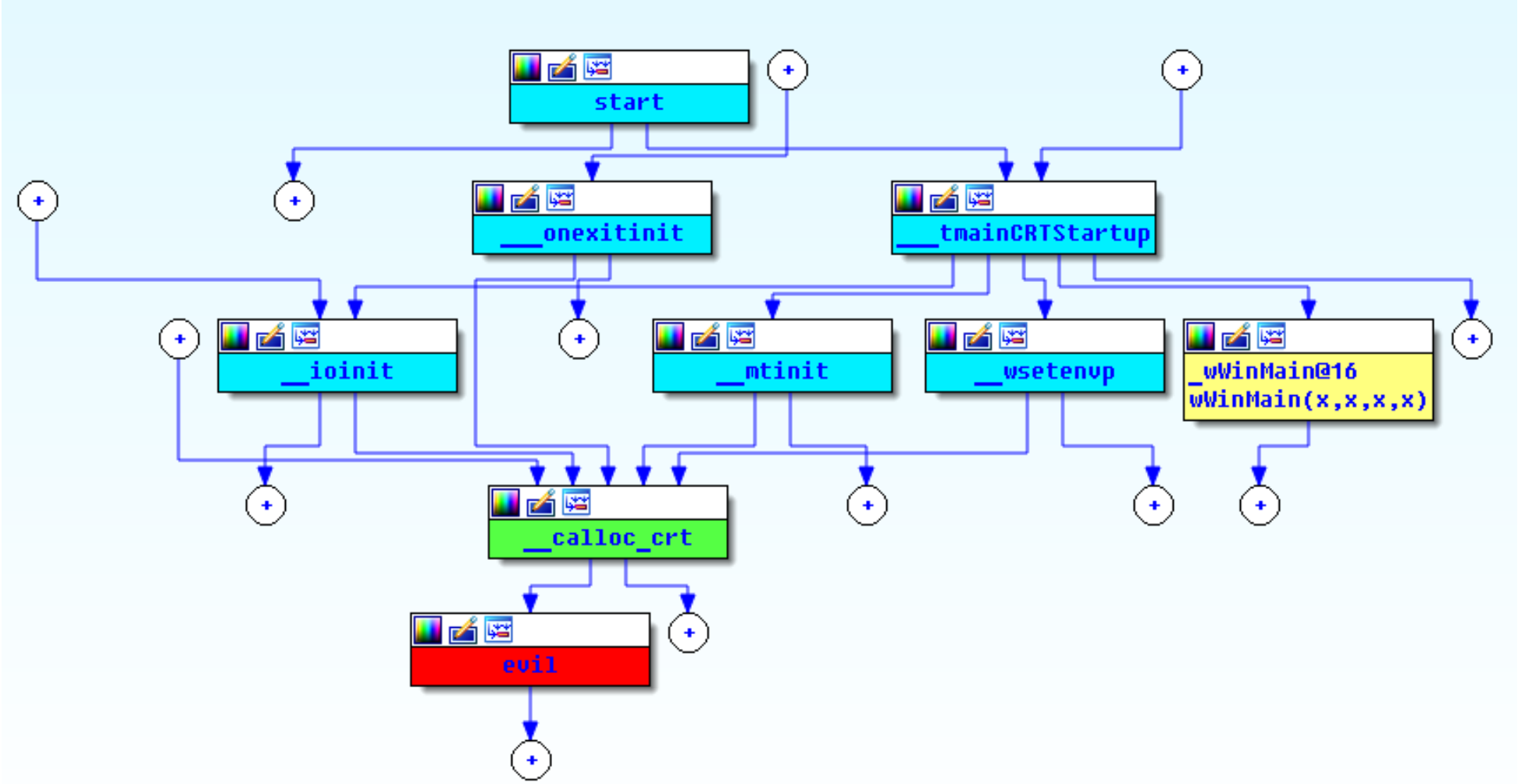
Functionality

Structure of the malware

- **Petya is packed** in order to avoid detection, circumvent static signatures, trick the heuristic analyzer, …
  - Goal: attempt to disguise the malware to look like a regular file

- **And** part of the standard compiler-generated runtime DLL that was replaced with malicious code keeping the function WinMain intact

Functionality



https://s3-eu-west-1.amazonaws.com/khub-media/wp-content/uploads/sites/43/2016/04/07192904/petya_eng_3.png

Functionality

- Structure of the malware
  - **Petya is packed**
  - **And** part of the standard compiler-generated runtime DLL that was replaced with malicious code keeping the function WinMain intact:
    - The file looks legitimate – WinMain doesn't contain malicious code.
    - If the malicious code works will send the system into BSOD so any breakpoint set at WinMain will never be reached.

Functionality

- Structure of the malware
  - **Petya is packed**
  - **And** part of the standard compiler-generated runtime DLL that was replaced with malicious code keeping the function WinMain intact.

  - Malware is contained inside a dynamically linked library: Setup.dll

Functionality

- Structure of the malware
  - Malware is contained inside a dynamically linked library: Setup.dll
    - Standard cryptographic routines are used to encrypt the code (mbedtls/polarssl)
    - Malware written in Microsoft Visual Studio (high level c/c++)
    - The Setup.dll is kept in RAM – never written to the disk.

Functionality

- High level functionality of Setup.dll:

    1. Re-write legitimate boot-sector with own malicious boot loader
    2. Generate a key, infection ID and other auxiliary information, and save them to the hard drive
    3. Cause a system abort and reboot, and pass control to the malicious loader.

Re-writing the bootloader (MBR)

1. XOR data at sector 0 (i.e. original boot-sector) with 0x37 (ASCII '7') and writes to sector 56
2. Encrypts sectors 1-33 with the same operation XOR 0x37
3. Generates configuration data for the malicious loader, writes them to sector 54
4. Creates the *verification sector* 55
5. Copies valid signatures and the partition table from the original MBR into its own first-level loader
6. Writes first-level malicious code to sector 0 of the disk
7. Writes *malicious loader* to sectors 34-50
8. Calls the function NtRaiseHardError, to cause BSOD.

Re-writing the bootloader (MBR)

Number of sector	Content
0	First-level malicious loader
1 – 33	Encrypted sectors 1-33 (XOR 0x37)
34 – 50	Second-level malicious code
...	
54	Configuration sector of the malicious program
55	Verification sector (populated with byte 0x37)
56	Encrypted original MBR code (XOR 0x37)

https://securelist.com/petya-the-two-in-one-trojan/74609/

Re-writing the bootloader (GPT)

- Similar process for GPT disks:

Number of sector	Content
0	First-level malicious loader
1 – 33	Encrypted sectors 1-33 (XOR 0x37)
34 – 50	Second-level malicious code
…	
54	Configuration sector of the malicious program
55	Verification sector (populated with byte 0x37)
56	Encrypted original MBR code (XOR 0x37)
…	
Backup LBA – Backup LBA + 33	Encrypted copy of GPT Header (XOR 0x37)

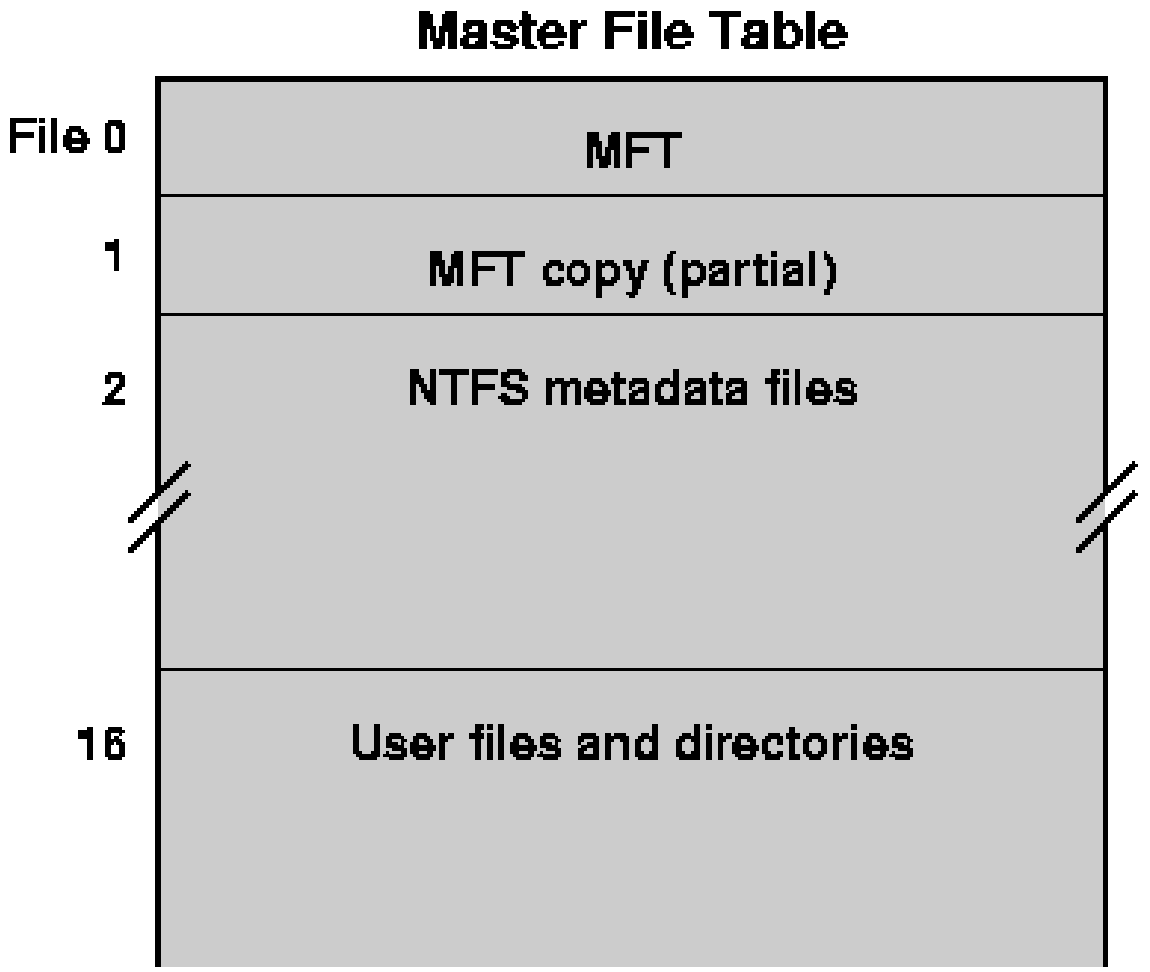
https://securelist.com/petya-the-two-in-one-trojan/74609/

Generating keys

- Setup.dll generates
    1. A random 16 characters long **password**
    2. An infection ID: **ecSessionData**
        1. generate **[ecSesPub, ecSesPrv]**
        2. secret <- ECDH (**ecSesPrv**, ecMasterPub)
        3. aesKey <- SHA512(secret)
        4. passwordXOR <- XOR(**ecSesPub, password)**
        5. passwordAES <- AES(**aesKey**, **passwordXOR**)

    - ecSessionData <- [ecSesPub, passwordAES]

Master File Table



Generating keys

- Setup.dll generates
  1. A random 16 characters long **password**
  2. An infection ID: **ecSessionData**
  3. A master file table (MFT) encryption key from the password

```
i = 0;
do
{
  config->salsa_key[2 * i] = password[i] + 0x7A;
  config->salsa_key[2 * i + 1] = 2 * password[i];
  ++i;
}
while ( i < 16 );
```

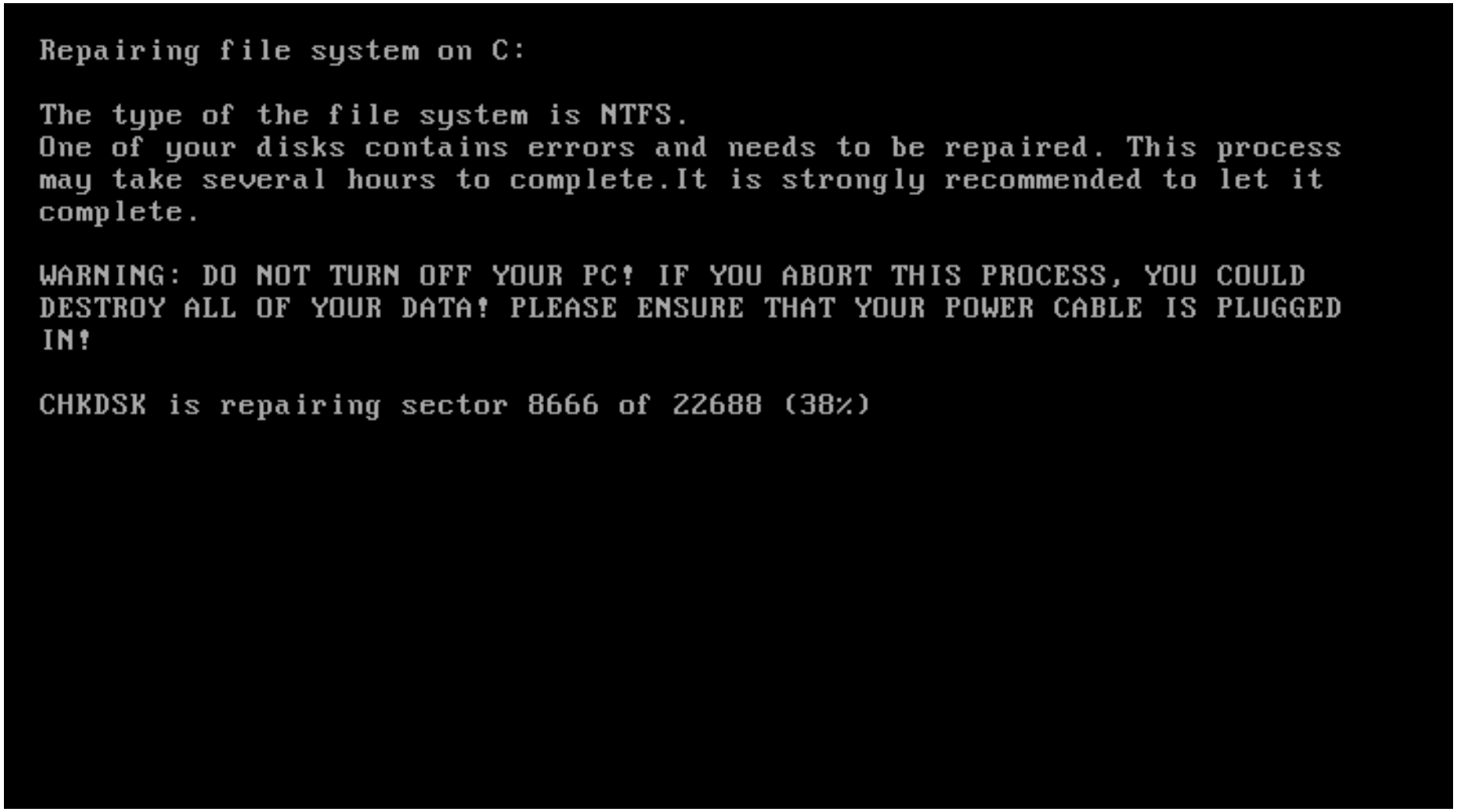
Generating keys

- Structure written to disk (@ sector 54) and available for use after reboot:

```
struct config
{
    char state;                 //Infection state
    char salsa_key[32];         //Key for MFT encryption
    char salsa_iv[8];           //IV for MFT encryption
    char mal_urls[128];         //URLs of ransom payment webpages
    char ec_data[343];          //Key data for the payment page
};
```

Encryption of MFT

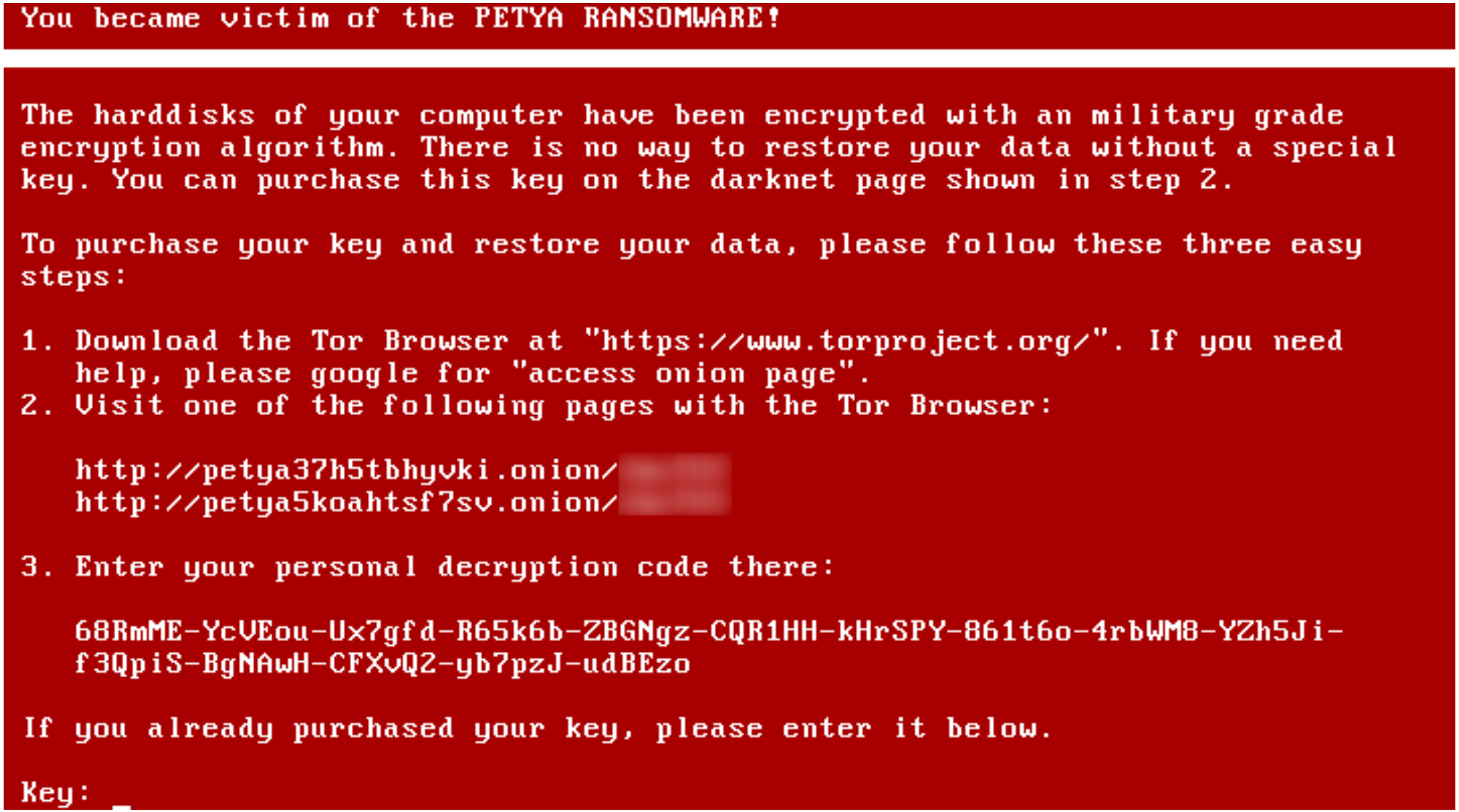
After reboot:



The Malicious loader

- Malicious loader (i.e. after reboot)
  1. Reads config data from disk (sector 54)
  2. Encrypt the verification sector 55 with the stream cipher Salsa20 (previously was only enc with xor 0x37)
  3. Searches for each partition's MFT on each connected hard drive
  4. Encrypts the MFT data with cipher Salsa20
  5. When encryption is over -> system reboot.

The Malicious loader



https://securelist.com/petya-the-two-in-one-trojan/74609/

The Malicious loader

- Key recovery (by criminal):
  1. [**ecSesPub**, **passwordAES**]<- ecSessionData
  2. secret = ECDH(**ecSesPub**, ecMasterPriv)
  3. aesKey= SHA512(secret)
  4. passwordXOR <- AES(aesKey, **passwordAES**)
  5. password <- XOR(**ecSesPub,** passwordXOR**)**

- '**password**' is sent to the user if ransom is paid.

https://securelist.com/petya-the-two-in-one-trojan/74609/

Petya Trojan

- Petya Trojan:
  - An unusual hybrid of an MBR blocker and data encryptor
  - Different than other ransomware, however its elements not new

WannaCry

- WannaCry is a ransomware cryptoworm (or ransomworm)
- Uses existing exploits in Windows:
  - EternalBlue: a SMBv1/v2* remote code execution vulnerability in Microsoft Windows.
  - Exploit became available **April** 14th 2017
  - Patched by Microsoft on **March** 14th 2017

*As discovered by Kaspersky and Symantec

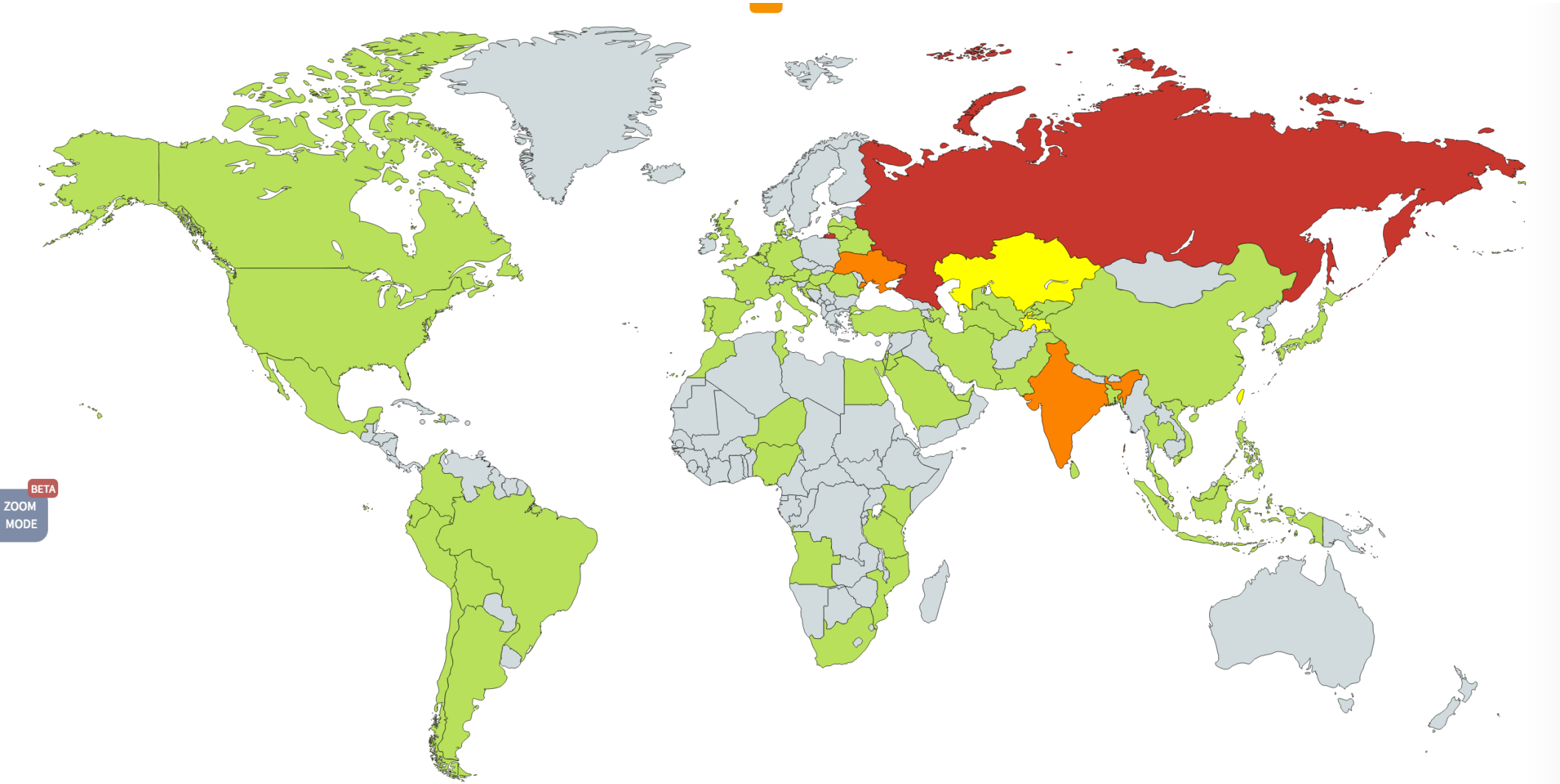
Botnet like behaviour

- Particularities:
  - Targeted only specific files (documents, achieves, emails, databases, VMs, ….)
  - Disabled default Windows backup protections i.e. shadow copy
  - Used TOR to connect to .onion addresses belonging to a C&C servers
- KillSwitch:
  - WannaCry was checking whether a certain gibberish URL led to a live web page:

    `hxxp://www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com`

Impact

On May 12, 2017 there were reported more than 45,000 attacks



https://s3-eu-west-1.amazonaws.com/khub-media/wp-content/uploads/sites/43/2017/05/07175138/wannacry_04.png

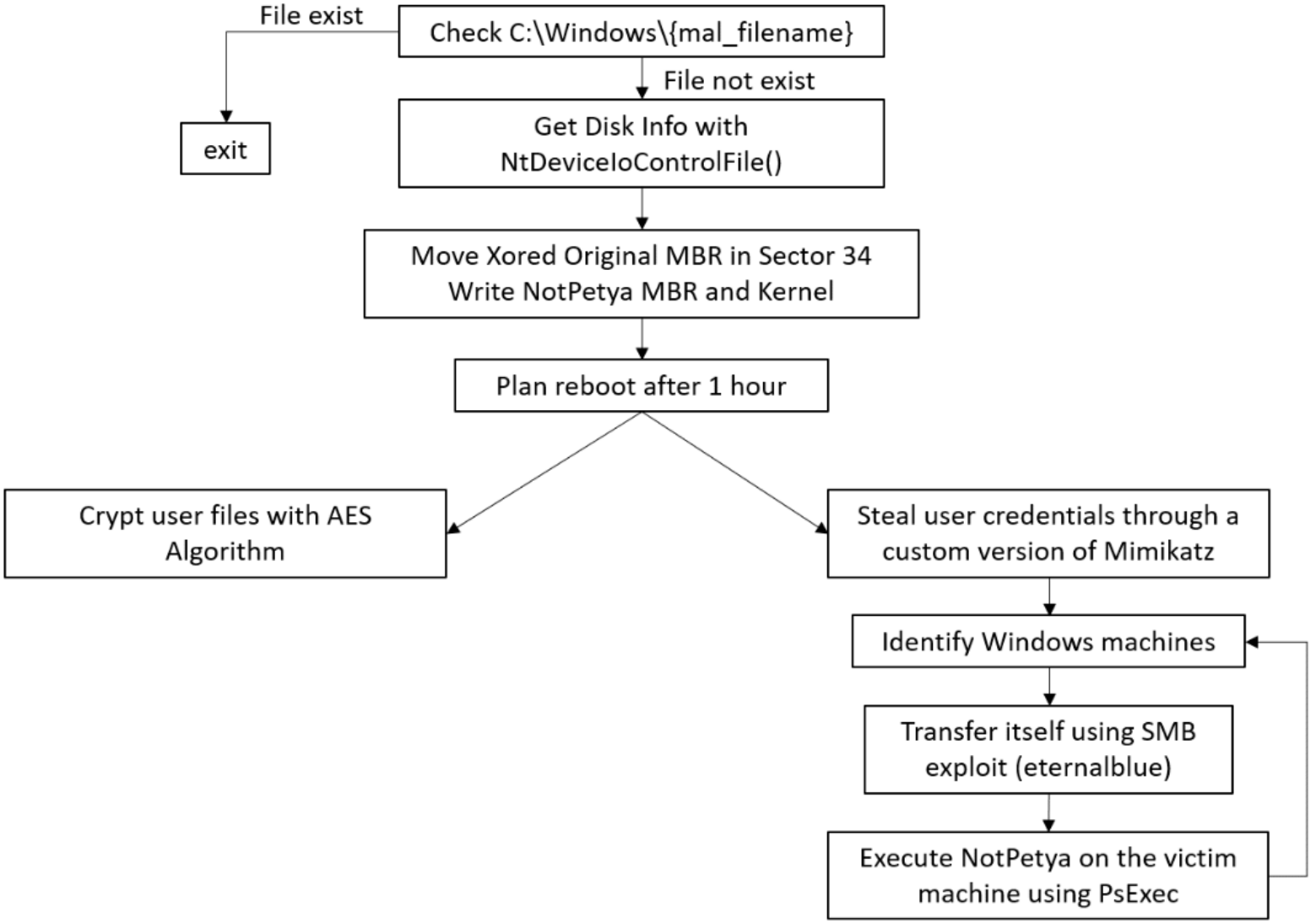
NotPetya

- Second generation Petya ransomware (June 2017)
- Used WannaCry spread mechanism and original Petya very effective method of infection
- Meant to be a destructive malware rather then a revenue stream:
  - Made less than four bitcoins (about $10,000)
  - WannaCry: more than 51 bitcoin (over $130,000)
  - CryptoLocker of 2013/14: more than $3 million

NotPetya

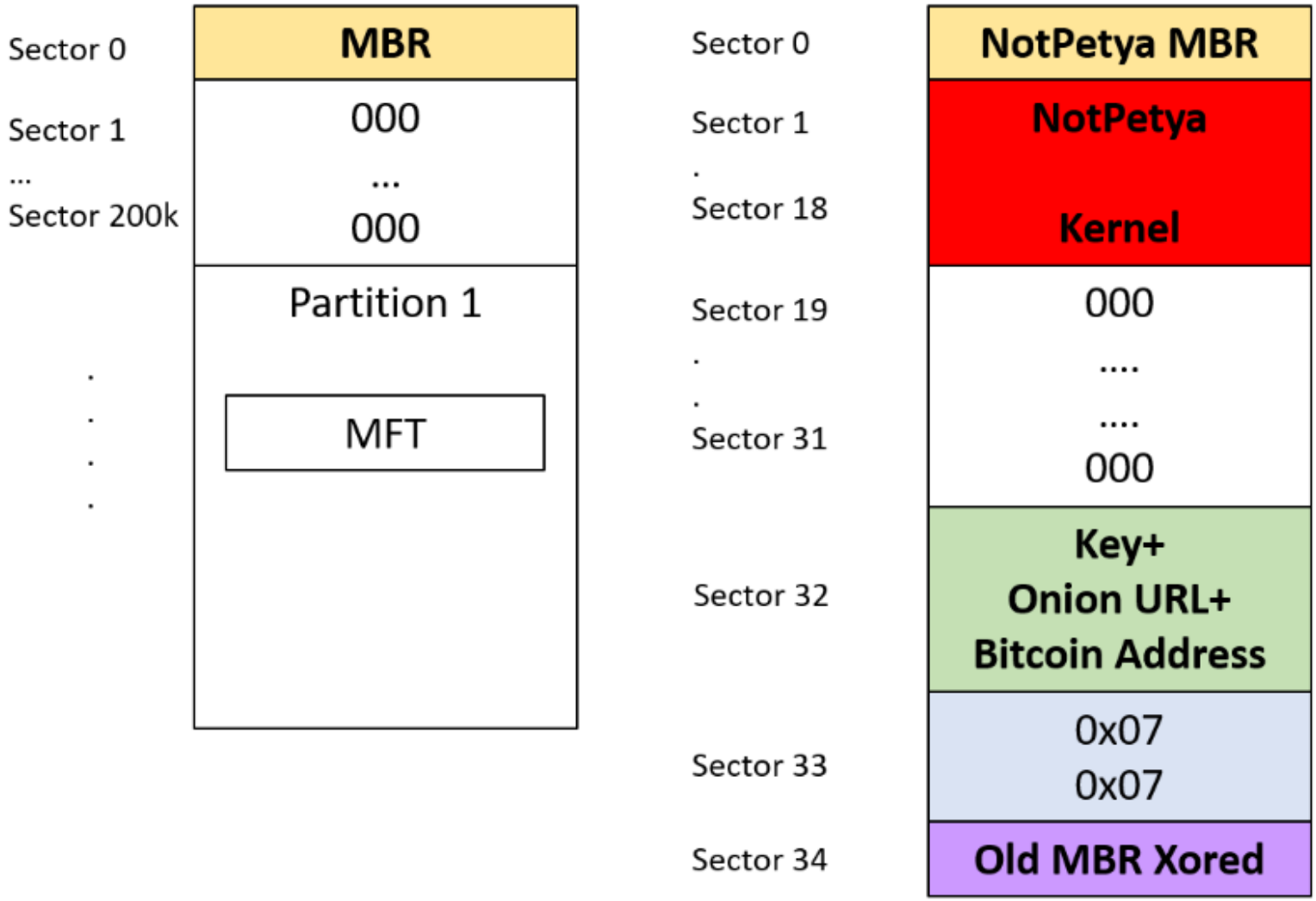
- Uses two different ransomware modules:
  - Encrypt up to 1MB of data in files with popular extensions (e.g. zip, 7z, pdf, php, doc(x), ppt(x), cpp, php…) with an AES-128 key.
  - Use original Petya MFT/Bootloader encryption lockout.

NotPetya



https://cybaze.it/download/zlab/NotPetya-report.pdf

NotPetya



https://cybaze.it/download/zlab/NotPetya-report.pdf

Conclusions

- Ransomware can use any spreading mechanism: virus-like, worm-like, Trojan-like, …
- **The “malicious behaviour“ consists of regular encryption which can very easily be considered as legitimate behaviour**
- Ransomware functionality can be integrated with botnets to deliver more devastating blows (i.e. logic bombs trigger by specific C&C commands)