# Malware memory analysis for non-specialists

*Investigating publicly available memory image for the Stuxnet worm*

R. Carbone
Certified Forensic Hacking Investigator (EC-Council)
GIAC Certified Incident Handler (SANS)
DRDC Valcartier

**Defence Research and Development Canada**

Scientific Report
DRDC-RDDC-2013-R1
November 2013

## Abstract

This report examines how an investigator can analyse an infected Windows® memory dump. The author investigates how to carry out such an analysis using Volatility and other investigative tools, including data carving utilities and anti-virus scanners. Volatility is a popular and evolving open source-based memory analysis framework upon which the author has proposed a memory-specific methodology for aiding fellow novice memory analysts. The author examines how Volatility can be used to find evidence and indicators of infection. This report is the fourth in this series concerning Windows malware-based memory analysis. This current work examines a memory image infected with the Stuxnet worm.

## Significance to defence and security

Canadian Armed Forces (CAF) networks are targets of choice for malware and directed attacks. This particular report in a series provides junior incident handlers with the necessary knowledge to handle and mitigate complex attacks from a memory snapshot. Using these simple, non-expert level techniques, a larger number of junior incident handlers can help with serious incidents, as opposed to relying entirely on the CAF's limited software reverse engineers.

## Résumé

Dans ce rapport, on décrit comment un enquêteur procède pour analyser l'image mémoire d'un système Windows® infecté. L'auteur étudie les techniques d'analyse au moyen de Volatility et d'autres outils tels que les utilitaires de récupération de données et les scanneurs antivirus. Volatility est un cadre populaire et évolutif d'analyse de la mémoire de source ouverte sur lequel l'auteur s'appuie pour proposer une méthodologie propre à la mémoire dans le but d'aider ses collègues analystes novices. L'auteur examine comment Volatility peut être utilisé pour trouver des preuves ou des indices d'infection. Ce rapport est le quatrième d'une série consacrée à l'analyse de la mémoire dans un environnement Windows® infecté par un maliciel. Le présent ouvrage porte sur l'image mémoire infectée par le ver Stuxnet.

## Importance pour la défense et la sécurité

Les réseaux des Forces armées canadiennes (FAC) constituent une cible de choix pour les maliciels et les attaques dirigées. Dans cette série, le présent rapport fournit aux gestionnaires d'incidents novices les connaissances nécessaires pour gérer les attaques complexes et les atténuer à partir d'une image instantanée de la mémoire. L'utilisation de ces techniques simples et non expertes permet à un plus grand nombre de gestionnaires d'incidents novices d'apporter leur aide lors d'attaques sérieuses plutôt que de se fier entièrement aux experts limités en rétro-ingénierie logicielle des FAC.

## Table of contents

# List of tables

## Acknowledgements

The author would like to thank Mr. Francois Rheaume, Defence Scientist, for conducting a preliminary review and peer review of this text and providing helpful comments to improve it. Moreover, the author would also like to extend his thanks to Mr. Martin Salois, Defence Scientist, for translating portions of this text.

## Disclaimer policy

## Requirements, assumptions and exclusions

The author assumes that the reader is altogether familiar with digital forensics and the various techniques and methodologies associated therein. This report is not an introduction to digital forensics or to said techniques and methodologies. However, the author has endeavoured to ensure that the reader can carry out his own forensic analysis of a computer memory image suspected of malware infection based on the information and techniques described herein.

The experimentation conducted throughout this report was carried out atop a Fedora Core 19 64-bit Linux operating system. Six different anti-virus scanners were used throughout this investigation. They include, in alphabetical order, Avast, AVG, BitDefender, Comodo, FRISK F-Prot and McAfee command line scanners. As for data carving tools and utilities, the author used Photorec version 6.14, part of the Testdisk (version 6.14) suite of data recovery tools.

The reader must have permission to use these tools on his computer system or network. Use of these tools and the analysis of virulent software always carry some inherent risk that must be securely managed and adequately mitigated.

An in-depth study of memory analysis techniques is outside the scope of this work, as it requires a comprehensive study of operating system internals and software reverse engineering techniques, both of which are difficult subjects to approach. Instead, this work should be considered as a guide to using the Volatility memory analysis framework with respect to malware infection and analysis.

When working with or examining files and data generated using various Volatility plugins, the use of the terms processes, memory sample files and memory dump files are used interchangeably.

Finally, the use of masculine is employed throughout this text to simplify it.

## Target audience

The targeted audience for this report is the computer forensic investigator who assesses suspect computer memory images for evidence of infection. Although computer memory analysis is a new field within the realm of digital forensics, there are those who have been conducting malware analysis and software reverse engineering for years, long before it came to the attention of today's practitioners. Thus, those seasoned veterans are aptly skilled, having taken years to develop their abilities. As such, the Volatility framework, while capable of providing insight to novices, is all the more capable in expert hands.

The author has written this report for others who, like himself, are required from time to time to conduct memory malware assessments and investigations. However, the author, like many others, is not seasoned enough to take full advantage of Volatility's capabilities. As such, this report combines both traditional forensic investigative techniques, coupled with Volatility's non-expert (non-reverse engineering) plugins, in order to develop an investigative how-to for non-memory experts.

This page intentionally left blank.

# 1 Background

## 1.1 Objective

The objective of this report is to examine how a computer forensic investigator, without specialised computer memory or software reverse engineering knowledge, can successfully investigate a memory image suspected of infection. More specifically, this report provides both a methodological basis and demonstrable techniques that a novice memory analyst can use as a basis for investigating suspected memory images.

The work carried out herein is based on the publicly available memory image for Stuxnet. This document is the fourth in a series of many. Ultimately, these reports will provide a methodological and foundational framework that novice and experienced investigators alike can rely on for guidance.

## 1.2 Summary

While memory analysis has largely been carried out by software reverse engineers and malware analysts, the advent of memory analysis-based forensic frameworks such as Volatility has made it possible for non-memory specialists to engage in the forensic analysis of malware-infected memory images. By combining Volatility, data carving utilities and anti-virus scanners, novice analysts have all the necessary tools required for conducting non-reverse engineering memory-based investigations.

This report examines the investigative techniques necessary for an investigator to conduct such memory analyses on his own. The first report written on this topic by the author examined the Zeus Trojan horse, found in TM 2013-018 [1], while the second examined the Prolaco worm and SpyEye Trojan horse, found in TM 2013-155 [2]. The third report examined the R2D2 Trojan horse; it is available in TM 2013-177 [3].

This specific report examines the Stuxnet worm, a highly complex piece of malware, in order to complement the ongoing assembly of quality tutorials in order to build a compendium of knowledge that can be used by the Canadian Armed Forces and our partners as a basis for conducting their own investigations. This series of reports examines various Windows-based malware infected memory images. It is hoped that these documents will serve as a learning guide.

Although others have engaged in the analysis of many of these publicly available memory images, the author is of the opinion that these analyses are insufficient for use as learning guides. Specifically, these analyses are either too limited in their investigative scope or provide too little information to be of use to budding memory analysts. Moreover, many of the analyses leave the reader asking more questions than when he began, due to their overall lack of use of a comprehensive investigative context. Thus, the author has strived to ensure that his investigative actions and lines of inquiry were well documented, even if some portions of a given investigation are unsuccessful, in order to ensure that the investigative context used was coherent.

This work was carried out over a period of several months as part of the Live Computer Forensics project, an agreement between DRDC Valcartier and the RCMP (SRE-09-015, 31XF20).

The results of this project will also be of great interest to the Canadian Forces Network Operations Centre (CFNOC), the RCMP's Integrated Technological Crime Unit (ITCU), the Sûreté du Québec and other cyber investigation teams.

## 1.3 Why write new tutorials?

The purpose of writing new tutorials was addressed in the first report of this series. [1]

## 1.4 Infected memory image metadata

The infected memory image for Stuxnet was procured from the following location: http://code.google.com/p/volatility/wiki/PublicMemoryImages. Its metadata, in uncompressed form, is as follows:

*Table 1: Infected memory image metadata.*

| Memory image name | Size (in MiB) | SHA1 hash value |
|---|---|---|
| stuxnet.vmem | 512 (exactly) | 6783d95883a32762042cae731887ae3693b030c1 |

## 1.5 Data carving

An in-depth examination of data carving can be found in two memorandums written by the author, specifically [1, 4].

## 1.6 Malware and anti-virus scanners

### 1.6.1 Specifics

An examination of malware and anti-virus scanner specifics can be found in [1].

However, due to the complex and sophisticated hiding mechanisms of the Stuxnet worm, this investigation relies more heavily on scanners.

### 1.6.2 Caveat

An analysis concerning the caveats of using malware and anti-virus scanners was conducted in [1].

## 1.7 Detailed list of software tools used

### 1.7.1 Anti-virus scanners

This report makes use of six anti-virus scanners, the same six as those used in [3]. These six scanners continue to represent a diverse cross-section of various detection mechanisms necessary for the detection of diverse malware. Each scanner was updated September 17, 2013; the analysis was carried out October 2013. Scanner specifics are listed in the following table:

*Table 2: List of anti-virus scanners and their command line parameters.*

| Anti-virus scanner | Command line parameters |
|---|---|
| Avast v.1.3.0 command line scanner | avast -c |
| AVG 2013 command line scanner version 13.0.3114 | avgscan -H -P -p |
| BitDefender for Unices v7.90123 Linux-amd64 scanner command line | bdscan (no parameters used) |
| Comodo Antivirus Product Version 1.1.268025.1 / Virus Signature Database Version 16954 | cmdscan -v -s |
| FRISK F-Prot version 6.3.3.5015 command line scanner | fpscan -u 4 -s 4 -z 10 --adware --applications --nospin |
| McAfee VirusScan for Linux64 Version 6.0.3.356 command line scanner | uvscan --RECURSIVE --ANALYZE --MANALYZE --MIME --PANALYZE --UNZIP --VERBOSE |

### 1.7.2 Data carving software

Photorec was used for data carving. The specifics concerning program settings were examined in [1].

### 1.7.3 Volatility framework

An examination of Volatility, its capabilities, main authors and contributors is found in [1].

A list of Windows-specific plugins currently supported by this version of Volatility is described in Annex A.

## 1.8 Investigative methodology

The overall investigative methodology, first proposed in [1] updated in [2], has been further clarified and lightly amended in this report. It is an approach to handling forensic memory investigations for non-reverse engineers and memory specialists. As more memory infections are examined by the author in ensuing reports, this methodology will likely be updated to reflect additional techniques required to adequately investigate the underlying memory image.

The methodology is both deductive and inductive. Starting with results obtained through scanning, carving, application of various Volatility plugins and combinations thereof, deductive reasoning is applied in order to determine the source or trigger of the infection. Inductive reasoning is then applied to identified (through scanning, string or hexadecimal analysis) malware to determine how the infection took hold and compromised the memory image.

This methodology can be summarised using the following steps:

### Part 1 - Protecting the disk image:

Ensure that the memory image has been set as read-only to prevent accidental changes or modifications from occurring to the image:

1. This varies according to the operating system used to perform the analysis and the underlying features of the filesystem that the memory image resides upon.

2. UNIX and Linux provide root-enabled mechanisms for filesystem-based read-only enforcement.

3. The reader may choose to use non-UNIX/Linux filesystems.

### Part 2 - Preliminary memory image scanning:

Analyse the suspect memory image with multiple anti-virus scanners:

1. Some scanners can perform in-depth analysis of memory images and in many instances, determine the nature of the underlying infection.

    a. Some of the scanners require the use of command line parameters while for others it is either optional or unnecessary. Mileage will vary according to the scanner in question.

2. Save the output from the various scanners.

### Part 3 - Data carving of memory image:

Using an advanced data carving utility, carve all potential data and files from the suspect image:

1. Use one highly capable data-carving tool instead of several mediocre tools.

2. Perform hashing against all carved data memory files:

    a. SHA1 hashing:

        i. Determine if any SHA1 matches can be identified against known hash-sets (.e.g., NSRL or HashKeeper hash-sets). Save any identified hash-set matches.

            a) If known "good" files are identified, they can be excluded from subsequent analysis.

            b) If known "bad" files are identified, these may be further examined using subsequent analyses.

            c) The application of "good" and "bad" files is optional. If they are not applied, then all carved data memory files, upon hashing, are to be subject to further analyses.

    b. Fuzzy (CTPH) hashing:

        i. Conduct fuzzy hashing against all carved data files and save the fuzzy hashes for use in subsequent steps.

3. Run the anti-virus scanners against all carved data and files, with attention focused on correlating scanner results:

    a. Files identified as "good" can be excluded from this step, if such identifications were made.

    b. If multiple scanners indicate a carved data memory file contains suspicious or malicious content, then it should be considered of interest. The more scanners corroborate the file's suspicious or malicious nature, the more it should be considered relevant to the investigation.

    c. Files picked up by only one scanner, especially scanners prone to false positives, can be considered as false positives, due to the nature of carved data memory files. These files are often corrupt due to the manner in which carving finds and recovers detected file signatures and structures.

        i. False positives should nevertheless be examined using, at a minimum, strings-based analysis. If this reveals nothing then the file can be considered "harmless." When uncertain, the false positive should be sent for reverse engineering analysis.

    d. Save the results from the various scanners and correlate the results.

## Part 4 - Volatility plugin based memory analysis:

If a given memory image continues to remain suspect (i.e., evidence or indications of infection have been found) then use the Volatility memory analysis framework to determine more about the infection and possibly how it occurred:

1. For a corrupt or unreadable memory image:

    a. It is possible that investigative endeavours using Volatility will not yield tangible results.

    b. All efforts to date (carving, scanning, Volatility, etc.) should be documented as a Lessons Learned.

    c. **Terminate** the investigation at this point and leave the methodology.

2. For a functional memory image:

    a. Using non-reverse engineering Volatility analysis plugins, find and extract as much information as possible concerning the underlying system, processes and threads that were running, communications, registry settings (if applicable), open files, mutexes, handles, etc.

        i. There are many plugins to choose from and it is unlikely that they will all be of use. Start by using plugins that are of immediate use (e.g., *imageinfo*, *pslist*, *psxview*) before using more advanced plugins.

        ii. Plugins in of themselves are not likely sufficient to put together the pieces, unless something is particularly obvious. Analyses that are more complex will undoubtedly require that investigative techniques be used to assemble an understanding of what is infecting the memory image including potentially piecing together how it occurred.

    b. Once one or more suspected processes, threads, DLLs, drivers or data files have been identified using the various plugins and/or investigative techniques, it is important they be dumped from memory using appropriate Volatility plugins.

        i. Plugins exist to dump DLLs, processes, and process' memory space and drivers (currently applies to Volatility 2.2).

        ii. All dumped processes, process space, DLLs and drivers are to be scanned by all available scanners to determine if they are malicious or potentially suspicious. These files are then to be hashed (SHA1 and fuzzy) and compared against the hashes of known files and the hashes of the carved data memory files.

            a) Using all available scanners, determine which of the dumped memory sample files (DLLs, process, etc.) are identified as malicious or suspicious. Crossscanner correlation is important and more weight should be applied to those samples detected by more than one scanner.

b) SHA1 and fuzzy hash dumped memory samples are to be compared appropriate data sources including lists of known files hashes (i.e., NSRL, HashKeeper, etc.) and the SHA1 and fuzzy hashes of the carved data memory files.

   i) In so doing, it may be possible to corroborate some of the files obtained through the carving of the memory file, a technique that can be used against corrupted or unreadable files, and identify the similarities between the various files (i.e., those carved from the memory image vs. dumped DLLs vs. dumped processes vs. dumped drivers, etc.).

c) Based on the scanner results, work backwards to identify and correlate Volatility plugin results to scanner results and dumped memory samples. This is the basis of inductive reasoning.

iii. Any files dumped that the investigator deems potentially suspicious (i.e., DLLs linked to a highly suspicious process or a very odd named driver) that were not identified through scanning should be analysed using alternative means.

   a) Alternative means includes string and hexadecimal analysis. Reverse engineering efforts are not examined in this methodology. Certain file and hex editors provide the ability to parse and analyse files. Very often, malicious software (DLLs, drivers and processes) will contain suspicious keywords, values, strings or Windows API functions.

      i) Various whitelists and blacklists for Windows APIs are available. The user must be aware that these may be pertinent specific versions of Windows.

   b) If nothing is found using strings and hexadecimal analysis, then reverse engineering efforts may be required.

iv. If no suspicious or malicious content can be found in the dumped memory samples, whether through scanning, string or hexadecimal analysis and reverse engineering efforts are not possible, then cease further analysis and ensure that all work, analyses and results are documented.

   a) However, even if the malware is not in memory anymore, sometimes the cross-correlation of information from the various Volatility plugins may lead the investigator to suspect or determine that one or more disk-based files or network connections may have been responsible for the infection (or at least involved to some extent).

   b) Reasons why the malware may no longer be memory-resident:

    i) Some malware can force the system to page them out so that they cannot be dumped or analysed from memory. Direct pagefile analysis, without memory contents is very difficult and may be a good location to hide.

    ii) Some malware guard against memory dumping and have the ability to unload from memory in the event memory acquisition is detected.

  c) Cease further memory analysis and if possible forensically acquire the disk from whence the memory image was obtained.

    i) If it is not possible to obtain the disk image, **cease and terminate** the analysis and leave the methodology.

    ii) With the disk image in hand, it can be analysed for evidence of infection. This type of analysis is not examined herein.

## Part 5 - Windows registry analysis:

The Windows registry can be long and complex to analyse but Volatility provides various Windows-specific plugins to aid in its analysis:

1. Extract UserAssist keys from the memory image using the Volatility *userassist* plugin.

2. Determine which registry hives are available in the memory image.

3. Armed with a list of potential registry keys commonly used by malware, generate a script that queries the variously identified hives for the presence of these keys. Generating the script takes only a few minutes using command line data processing tools to create them.

    a. This assumes the reader has a thorough understanding of UNIX command line data processing tools and the piping of output/input between programs in order to auto-generate functional scripts.

        i. This is the approach taken by the author but the reader is free to choose whatever approach he is most comfortable using.

        ii. The author has provided a list of commonly used registry keys targeted by malware, which is frequently revised.

4. Running the auto-generated script takes a few minutes and requires no user intervention, assuming the script was produced correctly.

5. Reading the script's output typically takes only a few minutes. Registry keys containing evidence or indicators of compromise are not generally difficult to find. However, depending on the nature of the malware in question, and the manner in which data was encoded into a given key, it may take more time.

6. If no pertinent registry information can be obtained from the memory image, then *strings*-based extraction and analysis can be used:

a. Using *strings*-based extraction, find and extract all 7, 8, 16 and 32-bit strings from the memory image and all suspect or infected process-based dumps.

b. This type of analysis can very easily generate millions of strings, even for moderately sized memory images:

    i. In order to readily analyse this quantity of data, relevant and context appropriate keyword or wordlist searches are required.

        a) Selecting or choosing appropriate keywords or wordlists is not necessarily straightforward. At a minimum, they should be context sensitive and reflect the evidence thus far obtained.

c. Based on the results from keyword/wordlist searches, establish and determine (if possible) both the possible presence and behaviour of the malware.

## Part 6 - Miscellaneous (optional):

Once Volatility and/or other analyses have been completed, it may be in the interest of the investigation to conclude by identifying additional information about the memory image and its contents:

1. Encryption key detection:

    a. Various malware utilise various means for encrypting their network communications. Various FOSS and COTS software exist to identify and extract different types of encryptions keys embedded within a memory image:

        i. These include AES, RSA, Serpent and Twofish, as well as others.

        ii. The FOSS tools commonly used are *aeskeyfind*, *rsakeyfind* and *interrogate*.

        iii. COTS software typically find the keys and are then used to bypass disk encryption software.

        iv. In theory, any network-encrypted communications could be decrypted with the correct decryption key:

            a) This delves into cryptography and reverse engineering techniques, outside the scope of the methodology.

2. Establish the Windows devices used by a suspicious or malicious device driver:

    a. If a suspicious or malicious device driver is dumped from the memory image, using Volatility's *devicetree* plugin will enable investigators to determine what devices are used or created by the driver.

# 2 Memory investigation and analysis of Stuxnet

## 2.1 Background

### 2.1.1 Mise-en-scène

This analysis examines a memory image suspected of harbouring the Stuxnet worm, as based on the methodology put forward in Section 1.8. Much information is available on the web concerning this specific infection. References [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18] provide a wealth of information.

However, in contrast to previous reports by the author that used malware reports for information regarding the underlying infection [1, 2] this report does not. Instead, these reports and other sources of information are cited for the reader's use rather than for the author's investigative analysis. Moreover, as with previous analyses [1, 2, and 3], no use was made of existing Volatility analyses.

Specifically, in order to gain practical experience analysing memory images, the author is of the opinion that there is no substitute for applying keen attention to detail. This approach, while non-intuitive in nature, is adept at identifying out of the ordinary minutiae. Thus, this specific investigation, while applying the methodology outlined in Section 1.8, will also point out detected anomalies that may indicate potential indicators of compromise or other infection-based evidence.

### 2.1.2 About Stuxnet

The Stuxnet worm is an advanced Windows-specific malware that spreads via the network and USB flash/thumb drives. The origins of the worm are not known with certainty but it is alleged to be state-sponsored and to have been designed to target Iran's nuclear program, specifically its uranium enrichment centrifuges. Although it is not known with certainty who broke the story of the Stuxnet worm it is known that it was first identified by Belarusian anti-virus company VirusBlokAda, sometime in June 2010. It is also known that Kaspersky Labs and F-Secure were one of the very first virus labs to have received samples of the worm. Moreover, there are at least two known strains of the worm, with the second being more advanced and virulent than the first. The worm uses four specific Windows-based zero-day exploits to spread and perform privilege escalation attacks which have since that time been addressed by Microsoft through the application of various patches. [5, 6, 7, 8, 9 and 10]

The main objective of the worm is to infect Windows computer systems and networks with PLC-based SCADA systems. More specifically, it is thought that this worm specifically targets the SCADA control systems used by Iran for its uranium enrichment program, although it is possible that other non-centrifugal SCADA systems could become infected. [5, 6, 7, 8, 9 and 10]

Stuxnet is a very complex piece of malware, unlike anything else known up until the time of its discovery. Thus, attempting to define the worm's capabilities and the various mechanisms it uses to achieve its objective is complicated due to its many capabilities. However, it is important to emphasize that the worm is capable of delivering various payloads, some incorporated directly into the worm while most are downloaded via C&C servers. These payloads are then deployed by the worm to exploit the variously encountered environments and possible PLC and SCADA specific capabilities found on target systems and networks. [11, 12, 13, 14, 15, 16, 17 and 18]

## 2.2 Preliminary investigative steps

The steps examined in this subsection should be considered as required preliminary steps for examining a potentially infected memory image.

### 2.2.1 Safeguard the memory image

The memory image *stuxnet.vmem* was set to immutable atop an Ext4-based filesystem. The command used to perform this, carried out as the root user, was:

```
$ sudo chattr +i stuxnet.vmem
```

This results in a memory image that can no longer be modified, even by the root user. This is to prevent accidental modifications from occurring to this file.

### 2.2.2 Preliminary anti-virus scanning results

Scanning only the memory image itself with the six scanners outlined in Section 1.7.1, the only scanner that identified the memory image as infected was Avast. Its output was as follows:

```
./analysis/stuxnet.vmem    [infected   by   Win32:Small-HTYB
[Trj]]
```

Preliminary anti-virus scanner examination indicates that this memory image is likely infected with some malware. However, the type, as based on the message from Avast, is too generic for positive identification. It appears that Avast was the only scanner capable of at least partially examining the memory image’s internal structures. All anti-virus results were recorded and saved.

### 2.2.3 Data carving and file hashing

Photorec succeeded in recovering 2,544 files carved from the memory image as per the recommended Photorec settings put forward in Section 1.7.2. 82 duplicate files were found, thereby leaving 2,462 unique files recovered. Of the 2,544 recovered files, 864 were identified as PE-based files. Of those, 524 were identified as Windows 32-bit DLLs, while 340 were identified as standard Windows 32-bit PEs (no device drivers detected). No 64-bit PE-based files were identified. Three files were identified as UPX-based. Finally, five files were identified as MS-DOS 16-bit executables for Windows 3.x and one DOS batch file was identified.

Other file types were detected but were of no immediate use. However, their types were recorded and saved for possible future use within this analysis.

All recovered files were SHA1-hashed and then validated against NSRL hash-set 2.41 (June 2013). Results were stored for future use. Nineteen unique SHA1 hashes were confirmed as matching against the NSRL hash-set. However, in all 52 unique SHA1-based NSRL filenames matches were identified which are found in Annex B.

Finally, CTPH-based hashing (fuzzy hashing) was conducted using the *ssdeep* tool against the carved memory data files and the results stored for future use.

UPX-based unpacking failed as the executables were corrupted.

### 2.2.4 Anti-virus scanning results for carved memory data files

Using the six scanners and combining their output through UNIX command line processing tools (e.g., *cat*, *sort*, *find*, *tr*, *strings*, *awk*, *grep*, *uniq*, etc.), multiple multi-scanner matches were established. The match involving the most scanners was carved memory data file *f0785768.exe*. It was detected by Avast, AVG, BitDefender and Comodo. All matches are shown in the table below.

Specific logs for each scanner can be found in Annex C and matches are indicated accordingly therein. Only four of the six scanners (Avast, AVG, BitDefender and Comodo) succeeded in detecting one or more possible infections in the carved memory data files. In all, nine specific multi-scanner matches were established based on the scanners' results.

*Table 3: Matching of potentially infected carved memory data files vs. scanner.*

| Potentially infected file | Scanner matching |
|---|---|
| f0785768.exe | Avast |
| | AVG |
| | BitDefender |
| | Comodo |
| f0843952.swf | Avast |
| | AVG |
| f0595624.exe | AVG |
| | BitDefender |
| f0583552.dll | BitDefender |
| | Comodo |
| f0573960.dll | BitDefender |
| | Comodo |
| f0277432.dll | BitDefender |
| | Comodo |

| Potentially infected file | Scanner matching |
|---|---|
| f0264288.dll | Avast |
|  | AVG |
| f0264240.dll | Avast |
|  | AVG |
| f0262960.dll | AVG |
|  | BitDefender |

### 2.2.5 Concerning the discovery of a malicious Flash file

Scanner matching provided by Avast and AVG identified carved memory data file *f0843952.swf* as a malicious Flash file. Avast correctly recognized the file as CVE 2007-0071 (details available in Annex C). Further research has corroborated this file as the aforementioned vulnerability and reverse engineering efforts have substantiated the file's maliciousness. It carries out an integer overflow attack allowing an attacker to execute arbitrary code. However, not all versions of Flash are affected by this exploit. Modern implementations are no longer susceptible to it.

It is unknown how this file found itself in this infected memory image. The exploit is altogether unrelated to Stuxnet and no correlation could be found between it and the various malicious content discovered throughout this investigation specific to Stuxnet.

## 2.3 Volatility analysis

In order to investigate this specific memory image the use and output of various Volatility plugins are examined. All plugin-specific output was stored in appropriately named text files for possible future use, for preserving the analysis workflow and maintaining adequate documentation.

### 2.3.1 Step 1: Background information, process listings and analysis

This step examines the various Volatility plugins used to provide background information and context to the memory image. Process-based plugins are often able to provide important indicators of infection or compromise. However, they are not particularly helpful for determining if a computer system has been used inappropriately.

#### 2.3.1.1 Imageinfo plugin

The *imageinfo* plugin is used to provide basic contextual information about a suspect memory image. This should always be the first Volatility plugin used by an investigator.

Consider this plugin's output using command "*volatility -f stuxnet.vmem imageinfo*":

```
Determining profile based on KDBG search...

          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86
(Instantiated with WinXPSP2x86)
                     AS Layer1 : JKIA32PagedMemoryPae (Kernel
AS)
                     AS     Layer2     :     FileAddressSpace
(/media/scratch/Stuxnet_Report/stuxnet.vmem)
                      PAE type : PAE
                           DTB : 0x319000L
                          KDBG : 0x80545ae0
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000
             KUSER_SHARED_DATA : 0xffdf0000
           Image date and time : 2011-06-03 04:31:36 UTC+0000
     Image local date and time : 2011-06-03 00:31:36 -0400
```

This memory image appears to be running atop a 32-bit Windows XP computer system with Service Pack 3. It is equipped with one PAE-based processor (1 core) and the memory image is 512 MiB in size (based on the memory image's size determined using *ls -l*). The memory image was captured June 3, 2011 at 00:31:36 EDT.

### 2.3.1.2 Pslist plugin

The next step is to identify which processes are running within the memory image in order to determine if anything out of the ordinary is immediately visible. The *pslist* plugin provides a detailed process listing. It makes use of virtual memory addressing and offsets. This should always be the first Volatility process listing plugin used.

Consider this plugin's output using command "*volatility -f stuxnet.vmem pslist*":

*Table 4: Volatility Pslist plugin output (sorted by PID).*

| Offset(V) | Name | PID | PPID | Thds | Hnds | Sess | Wow64 | Start | Exit |
|---|---|---|---|---|---|---|---|---|---|
| 0x823c8830 | System | 4 | 0 | 59 | 403 | ------ | 0 | | |
| 0x8205ada0 | alg.exe | 188 | 668 | 6 | 107 | 0 | 0 | 2010-10-29 17:09:09 | |
| 0x81f14938 | ipconfig.exe | 304 | 968 | 0 | -------- | 0 | 0 | 2011-06-03 04:31:35 | 2011-06-03 04:31:36 |
| 0x81e86978 | TSVNCache.exe | 324 | 1196 | 7 | 54 | 0 | 0 | 2010-10-29 17:11:49 | |
| 0x820df020 | smss.exe | 376 | 4 | 3 | 19 | ------ | 0 | 2010-10-29 17:08:53 | |
| 0x821a2da0 | csrss.exe | 600 | 376 | 11 | 395 | 0 | 0 | 2010-10-29 17:08:54 | |
| 0x81da5650 | winlogon.exe | 624 | 376 | 19 | 570 | 0 | 0 | 2010-10-29 17:08:54 | |
| 0x81c543a0 | Procmon.exe | 660 | 1196 | 13 | 189 | 0 | 0 | 2011-06-03 04:25:56 | |
| 0x82073020 | services.exe | 668 | 624 | 21 | 431 | 0 | 0 | 2010-10-29 17:08:54 | |
| 0x81e70020 | lsass.exe | 680 | 624 | 19 | 342 | 0 | 0 | 2010-10-29 17:08:54 | |
| 0x82279998 | imapi.exe | 756 | 668 | 4 | 116 | 0 | 0 | 2010-10-29 17:11:54 | |
| 0x823315d8 | vmacthlp.exe | 844 | 668 | 1 | 25 | 0 | 0 | 2010-10-29 17:08:55 | |
| 0x81db8da0 | svchost.exe | 856 | 668 | 17 | 193 | 0 | 0 | 2010-10-29 17:08:55 | |

| Offset(V) | Name | PID | PPID | Thds | Hnds | Sess | Wow64 | Start | Exit |
|---|---|---|---|---|---|---|---|---|---|
| 0x81c498c8 | lsass.exe | 868 | 668 | 2 | 23 | 0 | 0 | 2011-06-03 04:26:55 | |
| 0x81e61da0 | svchost.exe | 940 | 668 | 13 | 312 | 0 | 0 | 2010-10-29 17:08:55 | |
| 0x81c0cda0 | cmd.exe | 968 | 1664 | 0 | -------- | 0 | 0 | 2011-06-03 04:31:35 | 2011-06-03 04:31:36 |
| 0x822b9a10 | wuauclt.exe | 976 | 1032 | 3 | 133 | 0 | 0 | 2010-10-29 17:12:03 | |
| 0x822843e8 | svchost.exe | 1032 | 668 | 61 | 1169 | 0 | 0 | 2010-10-29 17:08:55 | |
| 0x81e18b28 | svchost.exe | 1080 | 668 | 5 | 80 | 0 | 0 | 2010-10-29 17:08:55 | |
| 0x820ec7e8 | explorer.exe | 1196 | 1728 | 16 | 582 | 0 | 0 | 2010-10-29 17:11:49 | |
| 0x81ff7020 | svchost.exe | 1200 | 668 | 14 | 197 | 0 | 0 | 2010-10-29 17:08:55 | |
| 0x81e6b660 | VMwareUser.exe | 1356 | 1196 | 9 | 251 | 0 | 0 | 2010-10-29 17:11:50 | |
| 0x81fee8b0 | spoolsv.exe | 1412 | 668 | 10 | 118 | 0 | 0 | 2010-10-29 17:08:56 | |
| 0x81e0eda0 | jqs.exe | 1580 | 668 | 5 | 148 | 0 | 0 | 2010-10-29 17:09:05 | |
| 0x81fe52d0 | vmtoolsd.exe | 1664 | 668 | 5 | 284 | 0 | 0 | 2010-10-29 17:09:05 | |
| 0x8210d478 | jusched.exe | 1712 | 1196 | 1 | 26 | 0 | 0 | 2010-10-29 17:11:50 | |
| 0x821a0568 | VMUpgradeHelper | 1816 | 668 | 3 | 96 | 0 | 0 | 2010-10-29 17:09:08 | |
| 0x81fa5390 | wmiprvse.exe | 1872 | 856 | 5 | 134 | 0 | 0 | 2011-06-03 04:25:58 | |
| 0x81fc5da0 | VMwareTray.exe | 1912 | 1196 | 1 | 50 | 0 | 0 | 2010-10-29 17:11:50 | |
| 0x81c47c00 | lsass.exe | 1928 | 668 | 4 | 65 | 0 | 0 | 2011-06-03 04:26:55 | |
| 0x820ecc10 | wscntfy.exe | 2040 | 1032 | 1 | 28 | 0 | 0 | 2010-10-29 17:11:49 | |

Examining at the above listing, several issues have been highlighted (in red) as they represent potentially suspicious processes. Specifically, Windows systems should typically only have one instantiated process of *lsass.exe*. However, this system has three *lsass*-based processes running, specifically PIDs 680, 868 and 1928. Moreover, PIDs 868 and 1928 were spawned by PID 668 (*services.exe*) while PID 680 was spawned by PID 624 (*winlogon.exe*). Something may be amiss, as these two additional processes are not supposed to be spawned by *services.exe* but instead by *winlogon.exe*.

In addition, from a command shell (PID 968) process PID 304 (*ipconfig.exe*) was spawned. Finally, PID 660 (*Procmon.exe*) was spawned from PID 1196 (*explorer.exe*). These three processes, *cmd.exe*, *ipconfig.exe* and *Procmon.exe*, respectively, were likely initiated by the currently logged on user gives the impression that the user was searching for something to do with IP address configuration information (*ipconfig.exe*) or process/thread management (*Procmon.exe*).

Perhaps the *psscan* plugin will reveal additionally useful information.

#### 2.3.1.3 Psscan plugin

The *psscan* plugin uses physical memory addressing and scans memory images for _EPROCESS pool allocations, in contrast to the *pslist* plugin that uses virtual memory addressing and scans for EPROCESS lists. The benefit of using this plugin is that sometimes it succeeds in listing processes that cannot be found using other process listing plugins (i.e., *pslist* and *pstree*).

Consider the following output from this plugin, using command "*volatility -f stuxnet.vmem psscan*":

*Table 5: Volatility Psscan plugin output (sorted by PID).*

| Offset(P) | Name | PID | PPID | PDB | Time created | Time exited |
|---|---|---|---|---|---|---|
| 0x025c8830 | System | 4 | 0 | 0x00319000 | | |
| 0x0225ada0 | alg.exe | 188 | 668 | 0x0a940240 | 2010-10-29 17:09:09 | |
| 0x02114938 | ipconfig.exe | 304 | 968 | 0x0a940380 | 2011-06-03 04:31:35 | 2011-06-03 04:31:36 |
| 0x02086978 | TSVNCache.exe | 324 | 1196 | 0x0a940180 | 2010-10-29 17:11:49 | |
| 0x022df020 | smss.exe | 376 | 4 | 0x0a940020 | 2010-10-29 17:08:53 | |
| 0x023a2da0 | csrss.exe | 600 | 376 | 0x0a940040 | 2010-10-29 17:08:54 | |
| 0x01fa5650 | winlogon.exe | 624 | 376 | 0x0a940060 | 2010-10-29 17:08:54 | |
| 0x01e543a0 | Procmon.exe | 660 | 1196 | 0x0a940260 | 2011-06-03 04:25:56 | |
| 0x02273020 | services.exe | 668 | 624 | 0x0a940080 | 2010-10-29 17:08:54 | |
| 0x02070020 | lsass.exe | 680 | 624 | 0x0a9400a0 | 2010-10-29 17:08:54 | |
| 0x02479998 | imapi.exe | 756 | 668 | 0x0a940320 | 2010-10-29 17:11:54 | |
| 0x025315d8 | vmacthlp.exe | 844 | 668 | 0x0a9400c0 | 2010-10-29 17:08:55 | |
| 0x01fb8da0 | svchost.exe | 856 | 668 | 0x0a9400e0 | 2010-10-29 17:08:55 | |
| 0x01e498c8 | lsass.exe | 868 | 668 | 0x0a940360 | 2011-06-03 04:26:55 | |
| 0x02061da0 | svchost.exe | 940 | 668 | 0x0a940100 | 2010-10-29 17:08:55 | |
| 0x01e0cda0 | cmd.exe | 968 | 1664 | 0x0a9403a0 | 2011-06-03 04:31:35 | 2011-06-03 04:31:36 |
| 0x024b9a10 | wuauclt.exe | 976 | 1032 | 0x0a940340 | 2010-10-29 17:12:03 | |
| 0x024843e8 | svchost.exe | 1032 | 668 | 0x0a940120 | 2010-10-29 17:08:55 | |
| 0x02018b28 | svchost.exe | 1080 | 668 | 0x0a940140 | 2010-10-29 17:08:55 | |
| 0x022ec7e8 | explorer.exe | 1196 | 1728 | 0x0a940280 | 2010-10-29 17:11:49 | |
| 0x021f7020 | svchost.exe | 1200 | 668 | 0x0a940160 | 2010-10-29 17:08:55 | |
| 0x0206b660 | VMwareUser.exe | 1356 | 1196 | 0x0a9402e0 | 2010-10-29 17:11:50 | |
| 0x021ee8b0 | spoolsv.exe | 1412 | 668 | 0x0a9401a0 | 2010-10-29 17:08:56 | |
| 0x0200eda0 | jqs.exe | 1580 | 668 | 0x0a9401e0 | 2010-10-29 17:09:05 | |
| 0x021e52d0 | vmtoolsd.exe | 1664 | 668 | 0x0a940200 | 2010-10-29 17:09:05 | |
| 0x0230d478 | jusched.exe | 1712 | 1196 | 0x0a940300 | 2010-10-29 17:11:50 | |
| 0x023a0568 | VMUpgradeHelper | 1816 | 668 | 0x0a940220 | 2010-10-29 17:09:08 | |
| 0x021a5390 | wmiprvse.exe | 1872 | 856 | 0x0a9401c0 | 2011-06-03 04:25:58 | |
| 0x021c5da0 | VMwareTray.exe | 1912 | 1196 | 0x0a9402c0 | 2010-10-29 17:11:50 | |
| 0x01e47c00 | lsass.exe | 1928 | 668 | 0x0a9403c0 | 2011-06-03 04:26:55 | |
| 0x022ecc10 | wscntfy.exe | 2040 | 1032 | 0x0a9402a0 | 2010-10-29 17:11:49 | |

The information presented in this table appears the same as that presented by the *pslist* plugin. Differentiating their output may help to determine if there are any differences between them. This is done in the subsequent step.

#### 2.3.1.4 Differentiating the output between the pslist and psscan plugins

Distinguishing between the output of the *pslist* and *psscan* plugins may not be obvious at first glance. For this task, shell-based text processing is of significant use. By using the following command, it is readily possible to differentiate between the output of the two plugins:

```
$ cat pslist.txt psscan.txt | awk '{print $2"\t"$3}' | sort
| uniq -c | grep -v “    2”
```

This command results in the following output:

```
1 --------------------     ------
1 ----------------         ------
```

Thus, by using these commands, it was determined that there was no discernible difference in their output. Perhaps the next plugin, *psxview*, will be of more assistance.

#### 2.3.1.5 Psxview plugin

Volatility offers an additional capability for detecting hidden running processes. The *psxview* plugin provides a detailed listing of processes in a memory image by using five specific process detection methods. These include *pslist*, *psscan*, *thrdproc*, *pspcdid* and *csrss*. Moreover, the plugin makes use of physical memory addressing.

Using this plugin, a process may be considered “hidden” if a given detection mechanism lists it as FALSE. If it is listed as TRUE then it is visible to that mechanism. For a process to be considered hidden, it should be invisible to, at a minimum, any non-*csrss* detection mechanism but it may also be undetectable by additional mechanisms.

However, if a process is not seen by the *pslist* or *psscan* mechanisms then the process is without doubt hidden. Even so, this is not in of itself indicative of a process being suspicious or malicious. Instead, sometimes it has to do with how the process was spawned. Those processes listed as hidden by *thrdproc* or *pspcdid* carry far less weight if both *pslist* and *psscan* list them as “visible.” Investigators must consider many factors when deciding whether a given process is hidden, pseudo-hidden or visible and this will depend on which mechanisms see it and those which do not.

Sometimes processes may be marked as hidden by the *csrss* mechanism but they generally are not hidden. Therefore, any process marked as hidden using this method requires that at least one other mechanism detects it as hidden too. Consider that for Windows 7 and Vista systems, their list of internal processes is not generally available for direct consultation. For Windows XP, sometimes the required memory pages are swapped out. These various factors may affect the outcome of the *csrss* mechanism. [19]

For example, consider that PIDs 304 and 968 from the following table are listed as hidden by the *thrdproc* and *csrss* mechanisms but are visible to *pslist*, *psscan* and *pspcdid*. These processes are therefore considered as *visible*.

Consider the plugin’s output using command “*volatility -f stuxnet.vmem psxview*”:

*Table 6: Volatility Psxview plugin output (sorted by PID).*

| Offset(P) | Name | PID | pslist | psscan | thrdproc | pspcdid | csrss |
|---|---|---:|---|---|---|---|---|
| 0x025c8830 | System | 4 | TRUE | TRUE | TRUE | TRUE | FALSE |
| 0x0225ada0 | alg.exe | 188 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x02114938 | ipconfig.exe | 304 | TRUE | TRUE | FALSE | TRUE | FALSE |
| 0x02086978 | TSVNCache.exe | 324 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x022df020 | smss.exe | 376 | TRUE | TRUE | TRUE | TRUE | FALSE |
| 0x023a2da0 | csrss.exe | 600 | TRUE | TRUE | TRUE | TRUE | FALSE |
| 0x01fa5650 | winlogon.exe | 624 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x01e543a0 | Procmon.exe | 660 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x02273020 | services.exe | 668 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x02070020 | lsass.exe | 680 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x02479998 | imapi.exe | 756 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x025315d8 | vmacthlp.exe | 844 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x01fb8da0 | svchost.exe | 856 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x01e498c8 | lsass.exe | 868 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x02061da0 | svchost.exe | 940 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x01e0cda0 | cmd.exe | 968 | TRUE | TRUE | FALSE | TRUE | FALSE |
| 0x024b9a10 | wuauclt.exe | 976 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x024843e8 | svchost.exe | 1032 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x02018b28 | svchost.exe | 1080 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x022ec7e8 | explorer.exe | 1196 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x021f7020 | svchost.exe | 1200 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x0206b660 | VMwareUser.exe | 1356 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x021ee8b0 | spoolsv.exe | 1412 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x0200eda0 | jqs.exe | 1580 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x021e52d0 | vmtoolsd.exe | 1664 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x0230d478 | jusched.exe | 1712 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x023a0568 | VMUpgradeHelper | 1816 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x021a5390 | wmiprvse.exe | 1872 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x021c5da0 | VMwareTray.exe | 1912 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x01e47c00 | lsass.exe | 1928 | TRUE | TRUE | TRUE | TRUE | TRUE |
| 0x022ecc10 | wscntfy.exe | 2040 | TRUE | TRUE | TRUE | TRUE | TRUE |

Based on the plugin's output, no truly hidden processes were found for this memory image.

Although processes *cmd.exe* and *ipconfig.exe* are listed as hidden by the *thrdproc* detection mechanism no concern should be given here, as it is normal for console-based single-threaded programs instantiated directly from *cmd.exe* to be indiscernible via thread-based detection.

#### 2.3.1.6 Summary and analysis

The Volatility *pslist* and *psscan* plugins have potentially found indicators of compromise. More specifically, the detection of three *lsass.exe* based processes rather than the standard one process is suspicious. Additional analyses will be conducted in the subsequent step to determine if there are other indicators of compromise in this memory image.

### 2.3.2 Step 2: State-based information and analysis

This step examines various state-based plugins that can be used to establish additional evidence of infection. These plugins often provide information that process-listing plugins cannot.

#### 2.3.2.1 Cmdscan and consoles plugins

The *cmdscan* and *consoles* plugins may reveal additional information about commands typed into a command shell.

The *cmdscan* plugin is used to query the process memory of *csrss.exe* or *conhost.exe* for possible commands that may have been entered into the system shell (*cmd.exe*; i.e., PID 968) or through a backdoor or RDP session by an attacker. Specifically, it looks for COMMAND_HISTORY based structures left behind in memory. The scanning of *csrss.exe* applies to Windows XP, 2003, Vista and Server 2008 while the use of *conhost.exe* applies to Windows 7. The effect of this plugin against Windows 2000, 8 and Server 2012 is not currently known and has not been attempted by the author. [16]

The *consoles* plugin is similar to *cmdscan* except that it searches for CONSOLE_INFORMATION based data structures instead. More specifically, it provides the command history of commands fed to the system shell (*cmd.exe*; i.e., PID 968) or through backdoors and this data structure keeps both the input and output buffers for commands found using this plugin. [16]

To query a memory image using these two plugins, the following commands are issued:

```
$ volatility -f stuxnet.vmem cmdscan

$ volatility -f stuxnet.vmem consoles
```

These two commands resulted in no output whatsoever.

#### 2.3.2.2 Connscan plugin

The first network-based Volatility plugin that should be used is *connscan*. It is used to verify the existence of ongoing network connections and scans a memory image for current or recently terminated connections. This plugin makes uses of physical memory addressing.

Running command "*volatility -f stuxnet.vmem connscan*" resulted in no output whatsoever.

### 2.3.2.3 Connections plugin

The *connections* plugin is used to find evidence of both recently terminated and ongoing communications. It therefore makes sense to use this plugin as it may reveal additional network-based information. Moreover, this plugin supports both physical and virtual memory addresses.

Running command "*volatility -f stuxnet.vmem connections*" resulted no output whatsoever.

### 2.3.2.4 Sockets and sockscan plugins

Volatility offers two additional network-based plugins, *sockets* and *sockscan*. The *sockets* plugin lists open sockets and may provide additional information about covert network channels, while the *sockscan* plugin scans a suspect memory image for all TCP sockets. Generally, the output is the same for both plugins with the exception of memory addresses, where the *sockets* plugin uses virtual memory addressing while the *sockscan* plugin uses physical memory addressing.

Thus, using the following commands it will be possible to determine which processes are ready for an incoming connection:

```
$ volatility -f stuxnet.vmem sockets > sockets.txt

$ volatility -f stuxnet.vmem sockscan > sockscan.txt

$ cat sockets.txt sockscan.txt | awk '{$1="";print}' | sort
-n | uniq > sockets_sockscan.txt
```

The output of these commands appears as shown in the following table:

*Table 7: Volatility Sockets and Sockscan plugins output (sorted by PID).*

| PID | Port | Proto | Protocol | Address | Create Time |
|---:|---:|---:|---|---|---:|
| 4 | 445 | 17 | UDP | 0.0.0.0 | 2010-10-29 17:08:53 |
| 4 | 445 | 6 | TCP | 0.0.0.0 | 2010-10-29 17:08:53 |
| 188 | 1025 | 6 | TCP | 127.0.0.1 | 2010-10-29 17:09:09 |
| 680 | 0 | 255 | Reserved | 0.0.0.0 | 2010-10-29 17:09:05 |
| 680 | 4500 | 17 | UDP | 0.0.0.0 | 2010-10-29 17:09:05 |
| 680 | 500 | 17 | UDP | 0.0.0.0 | 2010-10-29 17:09:05 |
| 940 | 135 | 6 | TCP | 0.0.0.0 | 2010-10-29 17:08:55 |
| 1032 | 123 | 17 | UDP | 127.0.0.1 | 2011-06-03 04:25:47 |
| 1080 | 1141 | 17 | UDP | 0.0.0.0 | 2010-10-31 16:36:16 |
| 1080 | 1142 | 17 | UDP | 0.0.0.0 | 2010-10-31 16:36:16 |
| 1200 | 1900 | 17 | UDP | 127.0.0.1 | 2011-06-03 04:25:47 |
| 1580 | 5152 | 6 | TCP | 127.0.0.1 | 2010-10-29 17:09:05 |

Looking at this data it is not immediately possible for most investigators to discern legitimate network port usage from suspicious usage. However, several important issues were determined. Firstly, port 123 is open and attached to PID 1032 (*services.exe*), a port typically used for NTP-based network time. This behaviour is not necessarily suspicious on its own. Nevertheless, caution is advised as NTP is sometimes used for nefarious purposes. [20, 21 and 22]

PID 1080 (*svchost.exe*) is using ports 1141 and 1142, both of which are not part of the standard Windows configuration and based on the list of running system processes, should not be in use. Ports 1141 and 1142 are typically in use by Oracle User Service, not by the Windows *svchost.exe* process, unless some specific service was configured. However, since these details are not known, they may be considered moderately suspicious. [20, 21, 22 and 23]

PID 1580 (*jqs.exe*) is using port 5152, a port associated to with Java Quick Starter [26]. PID 1200 (*svchost.exe*) is using port 1900, a port sometimes used for network-enabled Plug'n'Play devices [21, 22]. These ports and processes are not suspicious, as based on this information.

Finally, PID 188 (*alg.exe*) has an open port of 1025. This port should not normally be open for a Windows XP-based system. This port could be in use if the system were a Windows Exchange Server [24] or running IIS or NFS [25]. However, this system is performing none of duties.

Thus, at a minimum, ports 123 (PID 1032), 1025 (PID 188) and 1141/1142 (PID 1080) are suspicious. The next subsection will examine the relationship between these processes and their ports.

#### 2.3.2.5 Examining the relationship between suspicious ports and processes

Based on the information established thus far, as per process-based listings, two additional instances of process *lsass.exe* were found (a standard Windows systems should only ever have one running instance). Moreover, based on the information obtained by examining and correlating the *sockets* and *sockscan* plugins' output, several suspicious ports were found. All these results have been compiled into the following table, which paints an interesting story.

*Table 8: Suspicious Port vs. suspicious PIDs and PPIDs (sorted by PID).*

| Port | PID | Process Name | PPID | Parent Process Name |
|---|---|---|---|---|
| 1025 | 188 | alg.exe | 668 | services.exe |
| N/A | 868 | lsass.exe | 668 | services.exe |
| 123 | 1032 | svchost.exe | 668 | services.exe |
| 1141 | 1080 | svchost.exe | 668 | services.exe |
| 1142 | 1080 | svchost.exe | 668 | services.exe |
| N/A | 1928 | lsass.exe | 668 | services.exe |

Examining the above table, it is apparent that PID 668 (*services.exe*) is at the centre of this system's odd behaviour. However, it is entirely normal that processes such as *svchost.exe* and *alg.exe* are spawned by *services.exe*. What is not normal, however, are some of ports, specifically 1025, 1141 and 1142 coupled with the fact that two *lsass*-based processes were spawned by *services.exe* and not *winlogon.exe*. The issue of NTP time-based services is yet to be considered suspect as many Windows systems use NTP to synchronize their clocks.

Additional evidence or indicators of compromise are required before proceeding with the dumping of any specific process, DLLs or drivers from the memory image.

#### 2.3.2.6 Filescan plugin

If an infection is active and does not show itself via the network then the *filescan* plugin may be of assistance as it may be able to find open file handles in memory. Unfortunately, no direct link to these files is possible as a physical disk image is not available for analysis. This plugin makes use of physical address offsets.

The preferred method for detecting indicators of compromise is twofold. First, using keywords (e.g., Stuxnet, infection, rootkit, worm, etc.) it may be possible to find the infection, as malware programmers do not always use innocent looking filenames. Of course, this is at best a hit and miss approach. Secondly, an investigator can attempt to detect suspicious files based on their names and locations. However, this requires that the investigator have a very good working knowledge of the underlying operating system. Just looking at filenames[1] and locations will not produce meaningful results, unless something really sticks out.

For this specific investigation, since emphasis is placed on detecting indicators of compromise without the use of external documentation (i.e., malware reports), the investigator must studiously examine this plugin's output. Thus, running command "*volatility -f stuxnet.vmem filescan*," after extensive verification against the NSRL and after having ruled out various development, debugging programs and other miscellaneous files found in the plugin's output, six files stood out. These files are listed in the following table:

*Table 9: Volatility Filescan plugin output for suspicious files.*

| Offset (P) | #Ptr | #Hnd | Access | Name |
|---|---|---|---|---|
| 0x01dfa028 | 1 | 0 | R--r-- | \Device\HarddiskVolume1\WINDOWS\inf\oem7A.PNF |
| 0x01e0d028 | 1 | 0 | -WD--- | \Device\HarddiskVolume1\WINDOWS\inf\mdmeric3.PNF |
| 0x021b53c8 | 1 | 0 | RW---- | \Device\HarddiskVolume1\WINDOWS\inf\mdmcpq3.PNF |
| 0x02340c30 | 1 | 0 | R--r-d | \Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop\74ddc49a7c121a61b8d06c03f92d0c13.exe |
| 0x01eac6c8 | 1 | 0 | -WD--- | \Device\HarddiskVolume1\WINDOWS\system32\drivers\mrxnet.sys |
| 0x02137d18 | 1 | 0 | R--r-d | \Device\HarddiskVolume1\WINDOWS\system32\drivers\mrxnet.sys |

[1] Recall that a reliable source of filenames is the NSRL hash-set. It can be broken down manually (using command line text processing tools) by software product and operating system.

| Offset (P) | #Ptr | #Hnd | Access | Name |
|---|---|---|---|---|
| 0x0218a028 | 1 | 0 | R--r-d | \Device\HarddiskVolume1\WINDOWS\system32 \drivers\mrxcls.sys |
| 0x0219d340 | 1 | 0 | -WD--- | \Device\HarddiskVolume1\WINDOWS\system32 \drivers\mrxcls.sys |

The information listed in the above table paints an interesting story. One PNF-based file, specifically file *oem7A.PNF* has a very similar name to legitimate file *oem7.PNF*, as based on the NSRL. The other two PNF files do not resemble any currently known NSRL-based files; therefore, they are suspect.

File *74ddc49a7c121a61b8d06c03f92d0c13.exe* is a suspiciously long non-descriptive filename that is reminiscent of a malware dropper. As previously shown in past analyses conducted by the author ([1, 2]), various malware configuration files and dropper filenames were found, as examined in Table 10 below.

Files *mrxnet.sys* and *mrxcls.sys*, while similar in names to Windows drivers *mrxdav.sys* and *mrxsmb.sys*, are not found in the NSRL, and are therefore suspect.

Thus, six suspicious files have been identified using the *filescan* plugin. It is possible that others were present but could not be readily identified. Moreover, the information established thus far indicates a possible malware dropper, possibly two malicious drivers and three unidentified PNF files. The case for suspicious file *74ddc49a7c121a61b8d06c03f92d0c13.exe* being used as a malware dropper is seen in the following table:

*Table 10: Filenames of past analyses concerning malware processes, configuration files and dropper.*

| Report | Filename |
|---|---|
| Zeus analysis [1] | ZeuS_binary_5767b2c6d84d87a47d12da03f4f376ad.exe |
| Prolaco [2] | 1_doc_RCData_61 |
| SpyEye [2] | \Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop\spyeye\spyeye\edc7c152759ba0482bd39db0ea2c4319 |
| SpyEye [2] | \Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop\spyeye\spyeye\2b8a408b56eaf3ce0198c9d1d8a75ec0 |

These filenames demonstrate a tendency for malware to use non-standard naming conventions, one that should not be ignored.

### 2.3.2.7 Mutantscan plugin

The *mutantscan* plugin can sometimes reveal interesting information about Windows thread-based mutexes in memory. This plugin makes use of physical offset addressing.

Using command "*volatility -f stuxnet.vmem mutantscan*" yielded the following pertinent information after hours of pruning the output and validating suspicious mutexes against numerous web-based searches:

*Table 11: Volatility Mutantscan plugin output of suspicious mutexes.*

| Offset (P) | #Ptr | #Hnd | Signal | Thread | CID | Name |
|---|---|---|---|---|---|---|
| 0x01e4dbe0 | 2 | 1 | 1 | 0x00000000 |  | _!SHMSFTHISTORY!_ |
| 0x0205eae0 | 2 | 1 | 0 | 0x81fd8020 | 1032:1948 | Instance0: ESENT Performance Data Schema Version 40 |
| 0x020e3980 | 14 | 13 | 1 | 0x00000000 |  | SHIMLIB_LOG_MUTEX |
| 0x02108bb0 | 2 | 1 | 0 | 0x81fc0020 | 668:568 | PrefetchFileCacheOwner |
| 0x0217e138 | 2 | 1 | 1 | 0x00000000 |  | HGFSMUTEX00000000000003e7 |
| 0x0228cb48 | 3 | 2 | 1 | 0x00000000 |  | HGFSMUTEX0000000000029b4c |
| 0x023b75f8 | 2 | 1 | 0 | 0x81c6d180 | 668:476 | {E41362C3-F75C-4ec2-AF49-3CB6BCA591CA} |

Although the output above was thoroughly examined and revised several times, it is likely that other suspicious mutexes went unrecognized. Recognizing a suspicious mutex is very difficult without an exceedingly strong knowledge of Windows-based reverse engineering.

A web search for strings "*services.exe*" and "ESENT Performance Data Schema Version 40" reveals many virus and malware reports. The same occurred for strings "_!SHMSFTHISTORY!_", "SHIMLIB_LOG_MUTEX", "PrefetchFileCacheOwner", "HGFSMUTEX00000000000003e7", "HGFSMUTEX0000000000029b4c" and "{E41362C3-F75C-4ec2-AF49-3CB6BCA591CA}". Many of these searches revealed information about Stuxnet-specific infections while others were more generic.

When considering that CID 1032 was spawned by PID 668 process *services.exe* is again at the centre of things.

While these mutexes are likely indicative of malicious software lurking somewhere in memory, some are more suggestive than others are. There is no doubt that understanding the various mutexes in use at any given time within a given Windows system is a complex endeavour. Unfortunately, there is no available mutex whitelist or blacklist. The investigator, until he has sufficient reverse engineering skills, will have to rely on various web searches, which can take many hours to complete and may reveal too much or too little information, all of which must be evaluated by the investigator given the current investigative context and information on hand.

Running the *handles* plugin next may identify additional clues that may provide further context to these suspicious mutexes.

#### 2.3.2.8 Handles plugin

The *handles* plugin can reveal interesting information about processes and the resources attached or associated to them that might not be found using the previously examined plugins. This plugin makes use of virtual memory addressing.

Using command "*volatility -f stuxnet.vmem handles*," the following pruned output is of interest to the investigation and is as follows:

*Table 12: Volatility Handles plugin output for suspicious handles (sorted by PID).*

| Offset (V) | PID | Handle | Access | Type | Details |
|---|---|---|---|---|---|
| 0x82279998 | 668 | 0x200 | 0x1f0fff | Process | imapi.exe(756) |
| 0x823315d8 | 668 | 0x270 | 0x1f0fff | Process | vmacthlp.exe(844) |
| 0x81db8da0 | 668 | 0x284 | 0x1f0fff | Process | svchost.exe(856) |
| 0x81e61da0 | 668 | 0x330 | 0x1f0fff | Process | svchost.exe(940) |
| 0x822843e8 | 668 | 0x33c | 0x1f0fff | Process | svchost.exe(1032) |
| 0x81e18b28 | 668 | 0x378 | 0x1f0fff | Process | svchost.exe(1080) |
| 0x81ff7020 | 668 | 0x38c | 0x1f0fff | Process | svchost.exe(1200) |
| 0x81fee8b0 | 668 | 0x398 | 0x1f0fff | Process | spoolsv.exe(1412) |
| 0x81fe52d0 | 668 | 0x3b8 | 0x1f0fff | Process | vmtoolsd.exe(1664) |
| 0x81e0eda0 | 668 | 0x3bc | 0x1f0fff | Process | jqs.exe(1580) |
| 0x8205ada0 | 668 | 0x49c | 0x1f0fff | Process | alg.exe(188) |
| 0x81c498c8 | 668 | 0x654 | 0x1f0fff | Process | lsass.exe(868) |
| 0x81c47c00 | 668 | 0x660 | 0x1f0fff | Process | lsass.exe(1928) |
| 0x8210e3a8 | 1032 | 0x870 | 0x1f0003 | Event | W32TIME_NAMED_EVENT_ SYSTIME_NOT_CORRECT |
| 0x81dbc7b0 | 1032 | 0x888 | 0x12019f | File | \Device\NamedPipe\W32TIME |
| 0x81dbc568 | 1032 | 0x88c | 0x12019f | File | \Device\NamedPipe\W32TIME |

Examining the above table, the many processes associated and instantiated by PID 668 (*services.exe*) becomes apparent. Handles associated with PID 668 were flagged due to the evidence thus far accumulated signifying something likely nefarious about it.

Although PID 1032 (*svchost.exe*) was spawned by PID 668, suspicion is not drawn to it for this reason but rather due to its handles' association with NTP (network port 123). The three handles for this process are indicative of network time-related services. Moreover, handle *W32TIME_NAMED_EVENT_SYSTIME_NOT_CORRECT* is synonymous with malware and Windows errors, as based on various web searches. Thus, this handle should also be flagged. The two *\Device\NamedPipe\W32TIME* handles are not particularly worrisome.

It is likely that other suspicious handles were present but were not flagged due to the lack of appropriate context in which to evaluate them.

#### 2.3.2.9 Threads plugin

Two Volatility plugins are used in this section, specifically the *threads*. This plugin will be used to process the results of the *mutantscan* and *handles* plugins as more information may be determined concerning these processes and their threads.

The *threads* plugin searches for _ETHREADS and _KTHREADS data structures and uses virtual memory addressing.

Because there are many processes to examine, keen evaluation of these plugins' output of these is required as there are potentially hundreds of threads to examine where each thread's output consists of many lines of text. Thus, in order to maximize both the use of the *threads* plugin and UNIX command line processing tools, the following commands were issued to obtain information concerning the threads from PIDs 668, 868, 1032 and 1928 (*services.exe*, *lsass.exe*, *svchost.exe* and *lsass.exe*, respectively):

```
$ volatility -f stuxnet.vmem threads –p 668,868,1032,1928 |
grep Priority | grep –v BasePriority > threads_priority.txt

$ volatility -f stuxnet.vmem threads –p 668,868,1032,1928 |
grep BasePriority > threads_basepriority.txt

$ volatility -f stuxnet.vmem threads –p 668,868,1032,1928 |
grep ETHREAD > threads_ETHREAD.txt

$ pr -m -J -t threads_ETHREAD.txt threads_basepriority.txt
threads_priority.txt | awk '{print $2"   "$4"   "$6"   "$8"
"$10}' | grep -v "0x8   0x8" | grep -v "0x9   0x9" | grep
-v "0xf   0xf" > threads_merged_suspicious.txt
```

The output from these commands has been compiled into the following table:

*Table 13: Threads plugin output for BasePriority vs. Priority (sorted by PID/TID).*

| Offset (V) | PID | TID | BasePriority | Priority |
|---|---|---|---|---|
| 0x81fce5e8 | 668 | 220 | 0x9 | 0xa |
| 0x81fcdda8 | 668 | 348 | 0x9 | 0x10 |
| 0x81fb3448 | 668 | 472 | 0x9 | 0xa |
| 0x820f0da8 | 668 | 824 | 0x9 | 0xa |

| Offset (V) | PID | TID | BasePriority | Priority |
|---|---|---|---|---|
| 0x82331858 | 668 | 840 | 0x9 | 0x10 |
| 0x8220e3e8 | 668 | 872 | 0x9 | 0xa |
| 0x82249658 | 668 | 928 | 0x9 | 0xa |
| 0x8208f608 | 668 | 1088 | 0x9 | 0xa |
| 0x82284b58 | 668 | 1092 | 0x9 | 0xa |
| 0x82126bf0 | 668 | 1420 | 0x9 | 0xb |
| 0x81dc8478 | 668 | 1484 | 0x9 | 0xa |
| 0x81f492b0 | 668 | 1552 | 0x9 | 0x10 |
| 0x81deada8 | 668 | 1604 | 0x9 | 0xa |
| 0x81e0e400 | 668 | 1720 | 0x9 | 0xa |
| 0x8205a6f8 | 668 | 2008 | 0x9 | 0x10 |
| 0x822bbda8 | 868 | 1884 | 0x8 | 0x9 |
| 0x01e58aa0 | 1032 | 732 | 0x8 | 0x10 |
| 0x81fd6020 | 1032 | 744 | 0x8 | 0x9 |
| 0x021af468 | 1032 | 908 | 0x8 | 0x10 |
| 0x01e63da8 | 1032 | 1028 | 0x8 | 0x10 |
| 0x81ee01d0 | 1032 | 1044 | 0x8 | 0x9 |
| 0x81fa5758 | 1032 | 1056 | 0x8 | 0x9 |
| 0x8226f4d8 | 1032 | 1128 | 0x8 | 0x9 |
| 0x02331ae0 | 1032 | 1136 | 0x8 | 0x10 |
| 0x81c02208 | 1032 | 1228 | 0x9 | 0xb |
| 0x81fa8858 | 1032 | 1304 | 0x8 | 0x9 |
| 0x81f65548 | 1032 | 1312 | 0x8 | 0x10 |
| 0x81ff8450 | 1032 | 1324 | 0x8 | 0x9 |
| 0x81cb0b30 | 1032 | 1388 | 0x8 | 0x9 |
| 0x81feeda8 | 1032 | 1404 | 0x8 | 0xa |
| 0x81feeb30 | 1032 | 1408 | 0x8 | 0x9 |
| 0x81e0e020 | 1032 | 1576 | 0x8 | 0x9 |
| 0x821b27f8 | 1032 | 1660 | 0x8 | 0x10 |
| 0x81ca5020 | 1032 | 1752 | 0x8 | 0x9 |

| Offset (V) | PID | TID | BasePriority | Priority |
|---|---|---|---|---|
| 0x81daa668 | 1032 | 1796 | 0x8 | 0x9 |
| 0x81e62668 | 1032 | 1800 | 0x8 | 0x9 |
| 0x81e62240 | 1032 | 1804 | 0x8 | 0x10 |
| 0x821a0c28 | 1032 | 1824 | 0x8 | 0x10 |
| 0x82081460 | 1032 | 1828 | 0x8 | 0x9 |
| 0x81c71b40 | 1032 | 1888 | 0x8 | 0x9 |
| 0x81eebc28 | 1032 | 1892 | 0x8 | 0x10 |
| 0x8205a460 | 1032 | 1896 | 0x8 | 0x10 |
| 0x82270808 | 1032 | 1924 | 0x8 | 0xa |
| 0x81fd35c8 | 1032 | 2000 | 0x8 | 0x9 |
| 0x821a3da8 | 1032 | 2004 | 0x8 | 0x9 |
| 0x81d9e2d0 | 1032 | 2012 | 0x8 | 0x9 |
| 0x82059c18 | 1032 | 2016 | 0x8 | 0x9 |
| 0x81e14248 | 1928 | 416 | 0x8 | 0x9 |
| 0x81fed2c0 | 1928 | 780 | 0x8 | 0x10 |

The objective of this exercise is to determine which threads have had their *Priority* or *BasePriority* modified. Typically, these two values are the same, unless someone (i.e., the user) or something (i.e., malicious code) changes them. Characteristic values for *Priority* and *BasePriority* are typically *0x8* or *0x9*.

Thus, the issued commands have successfully isolated all threads that do not share the same *Priority* and *BasePriority* values. Of course, there are times when these values may legitimately be different from their typical values. However, looking at the table, especially for PIDs 668 and 1032, there are too many “priority adjustments” to be considered qualitatively “normal.”

While it is not possible to directly dump threads (and even if were possible it would not be advisable because threads are typically very small units of work) this table provides additional potential indicators of compromise.

### 2.3.2.10 Driverscan and DriverIRP plugins

The *driverscan* plugin scans a memory image for driver objects while the *driverirp* plugin scans for IRP hooks, often indicative of malicious software. The former plugin uses physical memory addressing while the latter uses neither virtual nor physical memory addressing but instead accepts KDBG and KPCR addresses.

Through these plugins, it may be possible to find the specific driver alluded to by *filescan* plugin. The following commands were issued to query for evidence about the two malicious drivers:

```
$ volatility -f stuxnet.vmem driverscan

$ volatility -f stuxnet.vmem driverirp
```

The output from these commands was pruned for pertinence.

The following relevant output for the *driverscan* plugin is as follows:

*Table 14: Volatility Driverscan plugin output of suspicious drivers.*

| Offset (P) | #Ptr | #Hnd | Start | Size (in hex) | Service Key | Name | Driver Name |
|---|---|---|---|---|---|---|---|
| 0x02126870 | 3 | 0 | 0xf895a000 | 0x4d80 | MRxCls | MRxCls | \Driver\MRxCls |
| 0x022e54f8 | 14 | 0 | 0xb21d8000 | 0x2980 | MRxNet | MRxNet | \Driver\MRxNet |

The following relevant output for the *driverirp* plugin is as follows:

```
DriverName: MRxCls
DriverStart: 0xf895a000
DriverSize: 0x4d80
DriverStartIo: 0x0
   0 IRP_MJ_CREATE                      0xf895a9e8 mrxcls.sys
   1 IRP_MJ_CREATE_NAMED_PIPE           0x804f354a ntoskrnl.exe
   2 IRP_MJ_CLOSE                       0xf895a9e8 mrxcls.sys
   3 IRP_MJ_READ                        0x804f354a ntoskrnl.exe
   4 IRP_MJ_WRITE                       0x804f354a ntoskrnl.exe
   5 IRP_MJ_QUERY_INFORMATION           0x804f354a ntoskrnl.exe
   6 IRP_MJ_SET_INFORMATION             0x804f354a ntoskrnl.exe
   7 IRP_MJ_QUERY_EA                    0x804f354a ntoskrnl.exe
   8 IRP_MJ_SET_EA                      0x804f354a ntoskrnl.exe
   9 IRP_MJ_FLUSH_BUFFERS               0x804f354a ntoskrnl.exe
  10 IRP_MJ_QUERY_VOLUME_INFORMATION    0x804f354a ntoskrnl.exe
  11 IRP_MJ_SET_VOLUME_INFORMATION      0x804f354a ntoskrnl.exe
  12 IRP_MJ_DIRECTORY_CONTROL           0x804f354a ntoskrnl.exe
  13 IRP_MJ_FILE_SYSTEM_CONTROL         0x804f354a ntoskrnl.exe
  14 IRP_MJ_DEVICE_CONTROL              0xf895aa04 mrxcls.sys
  15 IRP_MJ_INTERNAL_DEVICE_CONTROL     0x804f354a ntoskrnl.exe
  16 IRP_MJ_SHUTDOWN                    0x804f354a ntoskrnl.exe
  17 IRP_MJ_LOCK_CONTROL                0x804f354a ntoskrnl.exe
  18 IRP_MJ_CLEANUP                     0x804f354a ntoskrnl.exe
  19 IRP_MJ_CREATE_MAILSLOT             0x804f354a ntoskrnl.exe
  20 IRP_MJ_QUERY_SECURITY              0x804f354a ntoskrnl.exe
  21 IRP_MJ_SET_SECURITY                0x804f354a ntoskrnl.exe
  22 IRP_MJ_POWER                       0x804f354a ntoskrnl.exe
  23 IRP_MJ_SYSTEM_CONTROL              0x804f354a ntoskrnl.exe
  24 IRP_MJ_DEVICE_CHANGE               0x804f354a ntoskrnl.exe
  25 IRP_MJ_QUERY_QUOTA                 0x804f354a ntoskrnl.exe
  26 IRP_MJ_SET_QUOTA                   0x804f354a ntoskrnl.exe
  27 IRP_MJ_PNP                         0x804f354a ntoskrnl.exe

DriverName: MRxNet
DriverStart: 0xb21d8000
DriverSize: 0x2980
DriverStartIo: 0x0
   0 IRP_MJ_CREATE                      0xb21d8486 mrxnet.sys
   1 IRP_MJ_CREATE_NAMED_PIPE           0xb21d8486 mrxnet.sys
   2 IRP_MJ_CLOSE                        0xb21d8486 mrxnet.sys
   3 IRP_MJ_READ                         0xb21d8486 mrxnet.sys
   4 IRP_MJ_WRITE                        0xb21d8486 mrxnet.sys
   5 IRP_MJ_QUERY_INFORMATION            0xb21d8486 mrxnet.sys
   6 IRP_MJ_SET_INFORMATION              0xb21d8486 mrxnet.sys
   7 IRP_MJ_QUERY_EA                     0xb21d8486 mrxnet.sys
   8 IRP_MJ_SET_EA                       0xb21d8486 mrxnet.sys
   9 IRP_MJ_FLUSH_BUFFERS                0xb21d8486 mrxnet.sys
  10 IRP_MJ_QUERY_VOLUME_INFORMATION     0xb21d8486 mrxnet.sys
  11 IRP_MJ_SET_VOLUME_INFORMATION       0xb21d8486 mrxnet.sys
  12 IRP_MJ_DIRECTORY_CONTROL            0xb21d84ec mrxnet.sys
  13 IRP_MJ_FILE_SYSTEM_CONTROL          0xb21d8496 mrxnet.sys
  14 IRP_MJ_DEVICE_CONTROL               0xb21d8486 mrxnet.sys
  15 IRP_MJ_INTERNAL_DEVICE_CONTROL      0xb21d8486 mrxnet.sys
  16 IRP_MJ_SHUTDOWN                     0xb21d8486 mrxnet.sys
  17 IRP_MJ_LOCK_CONTROL                 0xb21d8486 mrxnet.sys
  18 IRP_MJ_CLEANUP                      0xb21d8486 mrxnet.sys
  19 IRP_MJ_CREATE_MAILSLOT              0xb21d8486 mrxnet.sys
  20 IRP_MJ_QUERY_SECURITY               0xb21d8486 mrxnet.sys
  21 IRP_MJ_SET_SECURITY                 0xb21d8486 mrxnet.sys
  22 IRP_MJ_POWER                        0xb21d8486 mrxnet.sys
  23 IRP_MJ_SYSTEM_CONTROL               0xb21d8486 mrxnet.sys
  24 IRP_MJ_DEVICE_CHANGE                0xb21d8486 mrxnet.sys
  25 IRP_MJ_QUERY_QUOTA                  0xb21d8486 mrxnet.sys
  26 IRP_MJ_SET_QUOTA                    0xb21d8486 mrxnet.sys
  27 IRP_MJ_PNP                          0xb21d8486 mrxnet.sys
```

Upon examination of the plugins' output, it is not readily possible to determine if these two drivers, *mrxcls.sys* and *mrxnet.sys*, are malicious. However, various details are now known about them including their location in memory and which driver IRP function codes[2] they are using. However, for non-reverse engineers it is not obvious to determine which codes are typically used for device drivers and which are used for malware as no whitelist or blacklist is readily available.

#### 2.3.2.11 Svcscan plugin

The *svcscan* Volatility plugin scans a memory image for Windows services. The drivers for a typical Windows system are generally registered as services, although valid exceptions exist. For instance, filter drivers are not commonly registered as services and these can include network sniffer drivers, certain filesystem drivers and network drivers. Thus, the claim that a driver not associated to a service is malicious or suspicious is not valid; however, when discovered it may be worth investigating a little further. Unfortunately, Volatility does not yet provide a plugin that can differentiate between registered and unregistered driver-based services, thus this remains a manual analysis.

Running the command "*volatility -f stuxnet.vmem svcscan*" did not produce any information concerning the two previously identified suspicious drivers, *mrxcls.sys* and *mrxnet.sys*. However, information concerning valid Windows drivers *mrxdav.sys* and *mrxsmb.sys*, which are very closely related in name to these two suspicious drivers, are registered as Windows services, as seen from the following output:

```
Offset: 0x385d28
Order: 112
```

[2] An IRP function code is denoted by *IRP_MJ*.

```
Process ID: -
Service Name: MRxDAV
Display Name: WebDav Client Redirector
Service Type: SERVICE_FILE_SYSTEM_DRIVER
Service State: SERVICE_RUNNING
Binary Path: \FileSystem\MRxDAV

Offset: 0x385db8
Order: 113
Process ID: -
Service Name: MRxSmb
Display Name: MRxSmb
Service Type: SERVICE_FILE_SYSTEM_DRIVER
Service State: SERVICE_RUNNING
Binary Path: \FileSystem\MRxSmb
```

### 2.3.2.12 Ldrmodules plugin

The *ldrmodules* plugin scans a memory image for signs of unlinked files (such as DLLs) in memory. These may be indicative of suspicious or malicious files lurking in memory. Although no suspicious DLLs have been found thus far, it does not preclude them from existing. Moreover, this plugin can also find other types of hidden files in memory including executables, libraries and configuration files.

To find potentially suspicious unlinked files, command "*volatility -f stuxnet.vmem ldrmodules | grep False*" was issued which generated the following output:

*Table 15: Volatility Ldrmodules plugin output (sorted by PID).*

| PID | Process | Base | InLoad | InInit | InMem | MappedPath |
|---|---|---|---|---|---|---|
| 4 | System | 0x7c900000 | False | False | False | \WINDOWS\system32\ntdll.dll |
| 188 | alg.exe | 0x01000000 | True | False | True | \WINDOWS\system32\alg.exe |
| 324 | TSVNCache.exe | 0x00400000 | True | False | True | \Program Files\TortoiseSVN\bin\TSVNCache.exe |
| 376 | smss.exe | 0x48580000 | True | False | True | \WINDOWS\system32\smss.exe |
| 600 | csrss.exe | 0x00460000 | False | False | False | \WINDOWS\Fonts\vgasys.fon |
| 600 | csrss.exe | 0x00f90000 | False | False | False | \WINDOWS\Fonts\vgaoem.fon |
| 600 | csrss.exe | 0x4a680000 | True | False | True | \WINDOWS\system32\csrss.exe |
| 600 | csrss.exe | 0x01350000 | False | False | False | \WINDOWS\Fonts\sserife.fon |
| 624 | winlogon.exe | 0x01000000 | True | False | True | \WINDOWS\system32\winlogon.exe |
| 660 | Procmon.exe | 0x00400000 | True | False | True | \Documents and Settings\Administrator\Desktop\SysinternalsSuite\Procmon.exe |
| 668 | services.exe | 0x01000000 | True | False | True | \WINDOWS\system32\services.exe |

| PID | Process | Base | InLoad | InInit | InMem | MappedPath |
|---|---|---|---|---|---|---|
| 680 | lsass.exe | 0x01000000 | True | False | True | \WINDOWS\system32\lsass.exe |
| 756 | imapi.exe | 0x01000000 | True | False | True | \WINDOWS\system32\imapi.exe |
| 844 | vmacthlp.exe | 0x00400000 | True | False | True | \Program Files\VMware\VMware Tools\vmacthlp.exe |
| 856 | svchost.exe | 0x01000000 | True | False | True | \WINDOWS\system32\svchost.exe |
| 868 | lsass.exe | 0x00080000 | False | False | False | - |
| 868 | lsass.exe | 0x01000000 | True | False | True | - |
| 940 | svchost.exe | 0x01000000 | True | False | True | \WINDOWS\system32\svchost.exe |
| 976 | wuauclt.exe | 0x00400000 | True | False | True | \WINDOWS\system32\wuauclt.exe |
| 1032 | svchost.exe | 0x01000000 | True | False | True | \WINDOWS\system32\svchost.exe |
| 1080 | svchost.exe | 0x01000000 | True | False | True | \WINDOWS\system32\svchost.exe |
| 1196 | explorer.exe | 0x01000000 | True | False | True | \WINDOWS\explorer.exe |
| 1196 | explorer.exe | 0x01760000 | False | False | False | \WINDOWS\Resources\Themes\Luna\Shell\NormalColor\shellstyle.dll |
| 1200 | svchost.exe | 0x01000000 | True | False | True | \WINDOWS\system32\svchost.exe |
| 1356 | VMwareUser.exe | 0x00400000 | True | False | True | \Program Files\VMware\VMware Tools\VMwareUser.exe |
| 1412 | spoolsv.exe | 0x01000000 | True | False | True | \WINDOWS\system32\spoolsv.exe |
| 1580 | jqs.exe | 0x00400000 | True | False | True | \Program Files\Java\jre6\bin\jqs.exe |
| 1664 | vmtoolsd.exe | 0x00400000 | True | False | True | \Program Files\VMware\VMware Tools\vmtoolsd.exe |
| 1712 | jusched.exe | 0x00400000 | True | False | True | \Program Files\Common Files\Java\Java Update\jusched.exe |
| 1816 | VMUpgradeHelper | 0x00400000 | True | False | True | \Program Files\VMware\VMware Tools\VMUpgradeHelper.exe |
| 1872 | wmiprvse.exe | 0x01000000 | True | False | True | \WINDOWS\system32\wbem\wmiprvse.exe |
| 1912 | VMwareTray.exe | 0x00400000 | True | False | True | \Program Files\VMware\VMware Tools\VMwareTray.exe |
| 1928 | lsass.exe | 0x00080000 | False | False | False | - |
| 1928 | lsass.exe | 0x01000000 | True | False | True | - |
| 2040 | wscntfy.exe | 0x01000000 | True | False | True | \WINDOWS\system32\wscntfy.exe |

Although most of the relevant output generated by this plugin is typical, four stood out from the rest (highlighted in red). Flags were raised for the *lsass* processes (PIDs 868 and 1928) which again were spawned by *services.exe* (PID 668), which has already been deemed highly suspicious. However, the fact that these two instances of *lsass* have unknown unlinked files makes them even more suspicious.

Rerunning the plugin using the verbose mode (parameter *-v*), as per command "*volatility -f stuxnet.vmem ldrmodules -v | grep -v -P '(True|False)'| sort | uniq*" enables the investigator to list all detected unlinked files. This command generates lots of output, far more than the standard *ldrmodules* plugin command. Thus, a list of known files (i.e., NSRL) is very useful when examining this output in order to hone in on unknown files. In so doing, three unidentified files were discovered and they are as follows:

```
Init   Path:   C:\WINDOWS\system32\KERNEL32.DLL.ASLR.0360b7ab   :
KERNEL32.DLL.ASLR.0360b7ab

Init   Path:   C:\WINDOWS\system32\KERNEL32.DLL.ASLR.0360c5e2   :
KERNEL32.DLL.ASLR.0360c5e2

Init   Path:   C:\WINDOWS\system32\KERNEL32.DLL.ASLR.0360c8ee   :
KERNEL32.DLL.ASLR.0360c8ee
```

However, no such Windows files are known to exist. Based on the full output from this command (not listed here) file *KERNEL32.DLL.ASLR.0360c5e2* is associated to PID 668 (*services.exe*) while file *KERNEL32.DLL.ASLR.0360c8ee* is associated to PID 940 (*svchost.exe*) and file *KERNEL32.DLL.ASLR.0360b7ab* is associated to PID 1928 (*lsass.exe*). Again, all three processes are interrelated to PID 668. This information is made clearer through the following table:

*Table 16: Verbose listing for plugin Ldrmodules specific to unknown KERNEL32.DLL.ASLR* filenames (sorted by PID).*

| Suspicious Filename | PID | PPID |
|---|---|---|
| KERNEL32.DLL.ASLR.0360c5e2 | 668 | 668 |
| KERNEL32.DLL.ASLR.0360c8ee | 940 | 668 |
| KERNEL32.DLL.ASLR.0360b7ab | 1928 | 668 |

### 2.3.2.13 Dlllist plugin

The *dlllist* plugin is primarily used to determine which DLLs are loaded for a given process. However, it can also be used to identify all DLLs loaded into a memory image. Running command "*volatility -f stuxnet.vmem dlllist*" identified, in total, 1252 DLLs loaded into memory.

Based on the *dlllist*-determined list of loaded DLLs, suspicious DLLs *KERNEL32.DLL.ASLR** were found within the process space of PIDs 668, 940 and 1928 (*services.exe*, *svchost.exe* and *lsass.exe*, respectively). These DLLs have the following plugin-related information:

*Table 17: Volatility Dlllist plugin output for suspicious DLLs KERNEL32.DLL.ASLR* (sorted by PID).*

| Filename | PID | Base address | Size (in hex) | Path |
|---|---|---|---|---|
| KERNEL32.DLL.ASLR.0360c5e2 | 668 | 0x013f0000 | 0x138000 | C:\WINDOWS\system32 |
| KERNEL32.DLL.ASLR.0360c8ee | 940 | 0x00d00000 | 0x138000 | C:\WINDOWS\system32 |
| KERNEL32.DLL.ASLR.0360b7ab | 1928 | 0x00870000 | 0x138000 | C:\WINDOWS\system32 |

Suspicious process *lsass.exe* (PID 868) does not show up in the above table because it does not have an associated *KERNEL32.DLL.ASLR** file.

Based on additional information provided by this plugin, further details concerning PIDs 668, 868, 940 and 1928 are shown below. Anomalies and points of interests have been highlighted (in red).

Details concerning PID 668 are as follows:

```
services.exe pid:    668
Command line : C:\WINDOWS\system32\services.exe
Service Pack 3

Base             Size Path
---------- ---------- ----
0x01000000    0x1c000 C:\WINDOWS\system32\services.exe
0x7c900000    0xaf000 C:\WINDOWS\system32\ntdll.dll
0x7c800000    0xf6000 C:\WINDOWS\system32\kernel32.dll
0x77dd0000    0x9b000 C:\WINDOWS\system32\ADVAPI32.dll
0x77e70000    0x92000 C:\WINDOWS\system32\RPCRT4.dll
0x77fe0000    0x11000 C:\WINDOWS\system32\Secur32.dll
0x77c10000    0x58000 C:\WINDOWS\system32\msvcrt.dll
0x5f770000     0xc000 C:\WINDOWS\system32\NCObjAPI.DLL
0x76080000    0x65000 C:\WINDOWS\system32\MSVCP60.dll
0x7dbd0000    0x51000 C:\WINDOWS\system32\SCESRV.dll
0x776c0000    0x12000 C:\WINDOWS\system32\AUTHZ.dll
0x7e410000    0x91000 C:\WINDOWS\system32\USER32.dll
0x77f10000    0x49000 C:\WINDOWS\system32\GDI32.dll
0x769c0000    0xb4000 C:\WINDOWS\system32\USERENV.dll
0x7dba0000    0x21000 C:\WINDOWS\system32\umpnpmgr.dll
0x76360000    0x10000 C:\WINDOWS\system32\WINSTA.dll
0x5b860000    0x55000 C:\WINDOWS\system32\NETAPI32.dll
0x5cb70000    0x26000 C:\WINDOWS\system32\ShimEng.dll
0x47260000     0xf000 C:\WINDOWS\AppPatch\AcAdProc.dll
0x77b40000    0x22000 C:\WINDOWS\system32\Apphelp.dll
0x77c00000     0x8000 C:\WINDOWS\system32\VERSION.dll
0x77b70000    0x11000 C:\WINDOWS\system32\eventlog.dll
0x76bf0000     0xb000 C:\WINDOWS\system32\PSAPI.DLL
0x71ab0000    0x17000 C:\WINDOWS\system32\WS2_32.dll
0x71aa0000     0x8000 C:\WINDOWS\system32\WS2HELP.dll
0x76f50000     0x8000 C:\WINDOWS\system32\wtsapi32.dll
0x76c30000    0x2e000 C:\WINDOWS\system32\WINTRUST.dll
0x77a80000    0x95000 C:\WINDOWS\system32\CRYPT32.dll
0x77b20000    0x12000 C:\WINDOWS\system32\MSASN1.dll
0x76c90000    0x28000 C:\WINDOWS\system32\IMAGEHLP.dll
0x01020000   0x2c5000 C:\WINDOWS\system32\xpsp2res.dll
0x68000000    0x36000 C:\WINDOWS\system32\rsaenh.dll
0x5ad70000    0x38000 C:\WINDOWS\system32\uxtheme.dll
0x75150000    0x13000 C:\WINDOWS\system32\Cabinet.dll
0x774e0000   0x13d000 C:\WINDOWS\system32\ole32.dll
0x013f0000                                        0x138000
C:\WINDOWS\system32\KERNEL32.DLL.ASLR.0360c5e2
0x76f20000    0x27000 C:\WINDOWS\system32\DNSAPI.dll
0x76d60000    0x19000 C:\WINDOWS\system32\IPHLPAPI.DLL
```

```
0x77120000    0x8b000 C:\WINDOWS\system32\OLEAUT32.dll
0x7c9c0000   0x817000 C:\WINDOWS\system32\SHELL32.dll
0x77f60000    0x76000 C:\WINDOWS\system32\SHLWAPI.dll
0x771b0000    0xaa000 C:\WINDOWS\system32\WININET.dll
0x71ad0000     0x9000 C:\WINDOWS\system32\WSOCK32.dll
0x773d0000                                         0x103000
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.2600.5512_x-
ww_35d4ce83\comctl32.dll
0x5d090000    0x9a000 C:\WINDOWS\system32\comctl32.dll
```

Details concerning PID 868 are as follows:

```
lsass.exe pid:    868
Command line : "C:\WINDOWS\\system32\\lsass.exe"
Service Pack 3

Base             Size Path
---------- ---------- ----
0x01000000     0x6000 C:\WINDOWS\system32\lsass.exe
0x7c900000    0xaf000 C:\WINDOWS\system32\ntdll.dll
0x7c800000    0xf6000 C:\WINDOWS\system32\kernel32.dll
0x77dd0000    0x9b000 C:\WINDOWS\system32\ADVAPI32.dll
0x77e70000    0x92000 C:\WINDOWS\system32\RPCRT4.dll
0x77fe0000    0x11000 C:\WINDOWS\system32\Secur32.dll
0x7e410000    0x91000 C:\WINDOWS\system32\USER32.dll
0x77f10000    0x49000 C:\WINDOWS\system32\GDI32.dll
```

Details concerning PID 940 are as follows:

```
svchost.exe pid:    940
Command line : C:\WINDOWS\system32\svchost -k rpcss
Service Pack 3

Base             Size Path
---------- ---------- ----
0x01000000     0x6000 C:\WINDOWS\system32\svchost.exe
0x7c900000    0xaf000 C:\WINDOWS\system32\ntdll.dll
0x7c800000    0xf6000 C:\WINDOWS\system32\kernel32.dll
0x77dd0000    0x9b000 C:\WINDOWS\system32\ADVAPI32.dll
0x77e70000    0x92000 C:\WINDOWS\system32\RPCRT4.dll
0x77fe0000    0x11000 C:\WINDOWS\system32\Secur32.dll
0x5cb70000    0x26000 C:\WINDOWS\system32\ShimEng.dll
0x6f880000   0x1ca000 C:\WINDOWS\AppPatch\AcGenral.DLL
0x7e410000    0x91000 C:\WINDOWS\system32\USER32.dll
0x77f10000    0x49000 C:\WINDOWS\system32\GDI32.dll
0x76b40000    0x2d000 C:\WINDOWS\system32\WINMM.dll
0x774e0000   0x13d000 C:\WINDOWS\system32\ole32.dll
0x77c10000    0x58000 C:\WINDOWS\system32\msvcrt.dll
0x77120000    0x8b000 C:\WINDOWS\system32\OLEAUT32.dll
0x77be0000    0x15000 C:\WINDOWS\system32\MSACM32.dll
0x77c00000     0x8000 C:\WINDOWS\system32\VERSION.dll
```

```
0x7c9c0000   0x817000 C:\WINDOWS\system32\SHELL32.dll
0x77f60000    0x76000 C:\WINDOWS\system32\SHLWAPI.dll
0x769c0000    0xb4000 C:\WINDOWS\system32\USERENV.dll
0x5ad70000    0x38000 C:\WINDOWS\system32\UxTheme.dll
0x773d0000                                  0x103000
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.2600.5512_x-
ww_35d4ce83\comctl32.dll
0x5d090000    0x9a000 C:\WINDOWS\system32\comctl32.dll
0x76a80000    0x64000 c:\windows\system32\rpcss.dll
0x71ab0000    0x17000 c:\windows\system32\WS2_32.dll
0x71aa0000     0x8000 c:\windows\system32\WS2HELP.dll
0x00670000   0x2c5000 C:\WINDOWS\system32\xpsp2res.dll
0x68000000    0x36000 C:\WINDOWS\system32\rsaenh.dll
0x71a50000    0x3f000 C:\WINDOWS\system32\mswsock.dll
0x662b0000    0x58000 C:\WINDOWS\system32\hnetcfg.dll
0x71a90000     0x8000 C:\WINDOWS\System32\wshtcpip.dll
0x76f20000    0x27000 C:\WINDOWS\system32\DNSAPI.dll
0x76d60000    0x19000 C:\WINDOWS\system32\iphlpapi.dll
0x76fb0000     0x8000 C:\WINDOWS\System32\winrnr.dll
0x76f60000    0x2c000 C:\WINDOWS\system32\WLDAP32.dll
0x76fc0000     0x6000 C:\WINDOWS\system32\rasadhlp.dll
0x76fd0000    0x7f000 C:\WINDOWS\system32\CLBCATQ.DLL
0x77050000    0xc5000 C:\WINDOWS\system32\COMRes.dll
0x00d00000                                  0x138000
C:\WINDOWS\system32\KERNEL32.DLL.ASLR.0360c8ee
0x5b860000    0x55000 C:\WINDOWS\system32\NETAPI32.dll
0x76bf0000     0xb000 C:\WINDOWS\system32\PSAPI.DLL
0x771b0000    0xaa000 C:\WINDOWS\system32\WININET.dll
0x77a80000    0x95000 C:\WINDOWS\system32\CRYPT32.dll
0x77b20000    0x12000 C:\WINDOWS\system32\MSASN1.dll
0x71ad0000     0x9000 C:\WINDOWS\system32\WSOCK32.dll
```

Details concerning PID 1928 are as follows:

```
lsass.exe pid:   1928
Command line : "C:\WINDOWS\\system32\\lsass.exe"
Service Pack 3

Base             Size Path
---------- ---------- ----
0x01000000     0x6000 C:\WINDOWS\system32\lsass.exe
0x7c900000    0xaf000 C:\WINDOWS\system32\ntdll.dll
0x7c800000    0xf6000 C:\WINDOWS\system32\kernel32.dll
0x77dd0000    0x9b000 C:\WINDOWS\system32\ADVAPI32.dll
0x77e70000    0x92000 C:\WINDOWS\system32\RPCRT4.dll
0x77fe0000    0x11000 C:\WINDOWS\system32\Secur32.dll
0x7e410000    0x91000 C:\WINDOWS\system32\USER32.dll
0x77f10000    0x49000 C:\WINDOWS\system32\GDI32.dll
0x00870000                                  0x138000
C:\WINDOWS\system32\KERNEL32.DLL.ASLR.0360b7ab
0x76f20000    0x27000 C:\WINDOWS\system32\DNSAPI.dll
0x77c10000    0x58000 C:\WINDOWS\system32\msvcrt.dll
0x71ab0000    0x17000 C:\WINDOWS\system32\WS2_32.dll
0x71aa0000     0x8000 C:\WINDOWS\system32\WS2HELP.dll
```

```
0x76d60000    0x19000 C:\WINDOWS\system32\IPHLPAPI.DLL
0x5b860000    0x55000 C:\WINDOWS\system32\NETAPI32.dll
0x774e0000   0x13d000 C:\WINDOWS\system32\ole32.dll
0x77120000    0x8b000 C:\WINDOWS\system32\OLEAUT32.dll
0x76bf0000     0xb000 C:\WINDOWS\system32\PSAPI.DLL
0x7c9c0000   0x817000 C:\WINDOWS\system32\SHELL32.dll
0x77f60000    0x76000 C:\WINDOWS\system32\SHLWAPI.dll
0x769c0000    0xb4000 C:\WINDOWS\system32\USERENV.dll
0x77c00000     0x8000 C:\WINDOWS\system32\VERSION.dll
0x771b0000    0xaa000 C:\WINDOWS\system32\WININET.dll
0x77a80000    0x95000 C:\WINDOWS\system32\CRYPT32.dll
0x77b20000    0x12000 C:\WINDOWS\system32\MSASN1.dll
0x71ad0000     0x9000 C:\WINDOWS\system32\WSOCK32.dll
0x773d0000                                      0x103000
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.2600.5512_x-
ww_35d4ce83\comctl32.dll
0x5d090000    0x9a000 C:\WINDOWS\system32\comctl32.dll
```

For three of the above listed outputs (PIDs 668, 940 and 1928) *KERNEL32.DLL.ASLR** files were identified in the list of process-associated DLLs. As for PID 868, too few DLLs were found associated with it as compared to PID 1928.

Finally, while searching the raw output from this plugin, the following two lines of evidence were identified as suspicious when considering that all files listed in the output only used one "\". Consider the following evidence for PIDs 868 and 1928 where the command lines for instantiating *lsass.exe* were found to be:

```
Command line : "C:\WINDOWS\\system32\\lsass.exe"
Command line : "C:\WINDOWS\\system32\\lsass.exe"
```

Versus the following command line's output for PID 680 which thus far is considered uninfected and uncompromised:

```
Command line : C:\WINDOWS\system32\lsass.exe
```

### 2.3.2.14 Summary and analysis

Based on the information, evidence and various indicators of compromise, there is little doubt that this memory image is uninfected.

Potentially suspicious network ports have been found associated with Windows processes they ordinarily would not be.

Furthermore, through the application of various plugins examined in this step, two highly suspicious device drivers have been identified which closely resemble in name two known Windows devices drivers. Moreover, a potential malware dropper has been found. In addition, three unidentified PNF files were discovered; again, one of these files very closely resembles a known Windows PNF file.

Finally, three very suspicious Windows DLLs have been found associated to three processes currently under suspicion of infection or compromise and two of the commands used to spawn *lsass.exe* were found to be abnormal.

### 2.3.3 Step 3: Detection and analysis of suspicious processes, DLLs and drivers

Sufficient evidence has been established indicating that suspicious or possibly malicious processes, DLLs or drivers are hiding in memory. The next phase is to dump them from memory so that they can be further analysed. This specific step examines how to dump them from memory and corroborate them with the evidence thus far obtained.

#### 2.3.3.1 Create data directories

Create directories *malfind*, *dlldump* and *moddump* for storing memory samples that are to be dumped from the memory image using Volatility. This is done using the following commands:

```
$ mkdir malfind
$ mkdir dlldump
$ mkdir moddump
```

#### 2.3.3.2 Malfind plugin

The evidence and potential indicators of compromise thus far demonstrated in Step 2 indicate that maliciously injected code may be in play, due the identification of various suspicious DLLs. The use of the *malfind* plugin and subsequent analysis of dumped memory samples may be able to identify which processes were subjected to code injection.

##### 2.3.3.2.1 Running the plugin

Volatility's *malfind* plugin was specifically designed to search for malicious code hidden through code injection. If memory address offsets are specified they must be physical memory addresses.

Because many much of the available evidence indicates infection, it makes more sense, from the perspective of the author, to conduct an at large analysis using this plugin rather than target several PIDs since so many processes were instantiated by suspicious process *services.exe* (PID 668).

Thus, the following command was run against the entire memory image to determine which processes were likely subverted through code injection:

```
$ volatility -f stuxnet.vmem malfind --dump-dir=malfind
```

This command succeeded in dumping 14 sample files from memory. Looking at only the textual output generated by the plugin, as found in Annex D, several are likely indicative of code injection. Nevertheless, subsequent analyses will confirm or rule them out from involvement in this investigation.

#### 2.3.3.2.2 AV scanning and file type determination

All 14 samples were scanned using the six aforementioned scanners. Of these 14 samples, eight were found to be infected by one or more scanners and many multi-scanner matches were established between samples, as shown in the following table:

*Table 18: Scanners results for Malfind-dumped samples (sorted by scanner).*

| Scanner | Filename | Infection Identification | Matches |
|---|---|---|---|
| Avast | process.0x81e61da0.0xd00000.dmp | Win32:Duqu-K [Rtk]<br>Win32:Duqu-F [Rtk]<br>Win32:Malware-gen<br>Win32:Stuxnet-C [Wrm]<br>Win32:Stuxnet [Rtk] | Match 1 |
| | process.0x81c47c00.0x1000000.dmp | Win32:Duqu-F [Rtk] | Match 2 |
| | process.0x81c47c00.0x870000.dmp | Win32:Duqu-K [Rtk]<br>Win32:Duqu-F [Rtk]<br>Win32:Malware-gen<br>Win32:Stuxnet-C [Wrm]<br>Win32:Stuxnet [Rtk] | Match 3 |
| | process.0x82073020.0x13f0000.dmp | Win32:Duqu-K [Rtk]<br>Win32:Duqu-F [Rtk]<br>Win32:Malware-gen<br>Win32:Stuxnet-C [Wrm]<br>Win32:Stuxnet [Rtk] | Match 4 |
| | process.0x81c498c8.0x80000.dmp | Win32:Malware-gen | Match 5 |
| | process.0x81c47c00.0x80000.dmp | Win32:Malware-gen | Match 6 |
| | process.0x81c498c8.0x1000000.dmp | Win32:Duqu-F [Rtk] | Match 7 |
| AVG | process.0x81c47c00.0x1000000.dmp | Trojan horse Duqu.A | Match 2 |
| | process.0x81e61da0.0xd00000.dmp | Trojan horse Agent3.CMIC | Match 1 |
| | process.0x81c47c00.0x870000.dmp | Trojan horse Agent3.CLEU | Match 3 |
| | process.0x82073020.0x13f0000.dmp | Trojan horse Agent3.CITI | Match 4 |
| | process.0x81c498c8.0x80000.dmp | Trojan horse Hider.IRJ | Match 5 |
| | process.0x81c47c00.0x80000.dmp | Trojan horse Hider.IRJ | Match 6 |
| | process.0x81c498c8.0x1000000.dmp | Trojan horse Duqu.A | Match 7 |

| Scanner | Filename | Infection Identification | Matches |
|---|---|---|---|
| BitDefender | process.0x81e61da0.0xd00000.dmp | Gen:Variant.Graftor.Elzob.17846 | Match 1 |
| | process.0x81c47c00.0x1000000.dmp | Trojan.Generic.7868042 | Match 2 |
| | process.0x81c47c00.0x870000.dmp | Gen:Variant.Graftor.Elzob.17846 | Match 3 |
| | process.0x82073020.0x13f0000.dmp | Gen:Variant.Graftor.Elzob.17846 | Match 4 |
| | process.0x81c498c8.0x80000.dmp | Backdoor.Generic.577628 | Match 5 |
| | process.0x81c47c00.0x6f0000.dmp | Backdoor.Generic.577628 | Match 8 |
| | process.0x81c47c00.0x80000.dmp | Backdoor.Generic.577628 | Match 6 |
| | process.0x81c498c8.0x1000000.dmp | Trojan.Generic.8217115 | Match 7 |
| | process.0x81e61da0.0xb70000.dmp | Backdoor.Generic.577628 | |
| Comodo | process.0x81e61da0.0xd00000.dmp | Packed.Win32.MUPX.Gen | Match 1 |
| | process.0x81c47c00.0x1000000.dmp | Malware | Match 2 |
| | process.0x81c47c00.0x870000.dmp | Packed.Win32.MUPX.Gen | Match 3 |
| | process.0x82073020.0x13f0000.dmp | Packed.Win32.MUPX.Gen | Match 4 |
| | process.0x81c498c8.0x80000.dmp | Worm.Win32.Stuxnet.K | Match 5 |
| | process.0x81c47c00.0x6f0000.dmp | Malware | Match 8 |
| | process.0x81c47c00.0x80000.dmp | Worm.Win32.Stuxnet.K | Match 6 |
| | process.0x81c498c8.0x1000000.dmp | Malware | Match 7 |
| FRISK | process.0x81c47c00.0x80000.dmp | W32/MalwareF.JBBO (exact) | Match 6 |
| | process.0x81c498c8.0x80000.dmp | W32/MalwareF.JBBO (exact) | Match 5 |
| McAfee | process.0x81c47c00.0x80000.dmp | Generic.dx!7CBDEFE442A1 trojan | Match 6 |
| | process.0x81c498c8.0x1000000.dmp | Generic.dx!1A97C7987EAC trojan | Match 7 |
| | process.0x81c498c8.0x80000.dmp | Generic.dx!7CBDEFE442A1 trojan | Match 5 |

Based on these results, both BitDefender and Comodo were the most sensitive to detecting infections from the dumped samples, each detecting eight of the fourteen as infected and each detected the very same samples as infected, thereby adding significant credence to their results.

Avast and AVG were close seconds in terms of infection detection, each detecting the same seven infections. Moreover, these seven correspond to seven of the eight detected by BitDefender and Comodo, again adding significant weight to these results.

Finally, F-Prot and McAfee detected the fewest infections at two and three, respectively. Their detected infections corresponded to those already picked by the previous four scanners – that is to say, they picked up nothing new.

Thus, the following eight dumped samples are very likely infected:

```
process.0x82073020.0x13f0000.dmp
process.0x81e61da0.0xd00000.dmp
process.0x81c47c00.0x1000000.dmp
process.0x81c47c00.0x870000.dmp
process.0x81c47c00.0x6f0000.dmp
process.0x81c47c00.0x80000.dmp
process.0x81c498c8.0x80000.dmp
process.0x81c498c8.0x1000000.dmp
```

Of these eight samples, they were examined using the *file* command to determine their file type. Based on its results, five were detected as 32-bit UPX compressed executables while two were detected as standard 32-bit Windows executables and one as an unknown data file, as per the following table:

*Table 19: File type determination for infected memory samples (sorted by filename).*

| **Filename** | **File Type (as per *file* command)** |
|---|---|
| process.0x81c47c00.0x1000000.dmp | 32-bit Windows PE executable |
| process.0x81c47c00.0x6f0000.dmp | Unknown data |
| process.0x81c47c00.0x80000.dmp | 32-bit UPX compressed executable |
| process.0x81c47c00.0x870000.dmp | 32-bit UPX compressed executable |
| process.0x81c498c8.0x1000000.dmp | 32-bit Windows PE executable |
| process.0x81c498c8.0x80000.dmp | 32-bit UPX compressed executable |
| process.0x81e61da0.0xd00000.dmp | 32-bit UPX compressed executable |
| process.0x82073020.0x13f0000.dmp | 32-bit UPX compressed executable |

Upon closer inspection, however, file *process.0x81c47c00.0x6f0000.dmp* was identified as an UPX compressed executable.

Finally, these samples can be attributed back to known processes as shown below:

*Table 20: PID attribution concerning scanner detected infected files for Malfind-dumped samples (sorted by PID).*

| **Filename** | **Actual Process** | **PID** | **PPID** |
|---|---|---|---|
| process.0x82073020.0x13f0000.dmp | services.exe | 668 | 668 |
| process.0x81c498c8.0x80000.dmp | lsass.exe | 868 | 668 |
| process.0x81c498c8.0x1000000.dmp | lsass.exe | 868 | 668 |
| process.0x81e61da0.0xd00000.dmp | svchost.exe | 940 | 668 |

| Filename | Actual Process | PID | PPID |
|---|---|---|---|
| process.0x81c47c00.0x1000000.dmp | lsass.exe | 1928 | 668 |
| process.0x81c47c00.0x870000.dmp | lsass.exe | 1928 | 668 |
| process.0x81c47c00.0x6f0000.dmp | lsass.exe | 1928 | 668 |
| process.0x81c47c00.0x80000.dmp | lsass.exe | 1928 | 668 |

The evidence and information presented thus far paint a very compelling image with respect to this infection.

#### 2.3.3.2.3 UPX decompression and brief analysis

The aforementioned executables identified as UPX-based were passed through a UPX decompressor. Of the six UPX files, only two were successfully decompressed, specifically files *process.0x81c47c00.0x80000.dmp* and *process.0x81c498c8.0x80000.dmp*.

Scanner identification has confirmed that these two decompressed UPX files are in fact the Stuxnet worm. Additional analysis by the author, outside the scope of this report, was conducted against these two files that has confirmed that they are in fact the worm. Moreover, these two files share the same SHA1 hash.

Further analysis and explanation will not be examined in this report, as the goal is to inform and guide budding memory analysts how to maximize the use of Volatility in collecting as much information as possible from a memory image. Cutting the analysis short at this point would not be, in the opinion of the author, of net benefit. Thus, their SHA1 and fuzzy hash values will not be included in the remainder of the analysis.

#### 2.3.3.2.4 SHA1 and fuzzy hashes

All 14 dumped files were hashed using the *sha1sum* command to determine their SHA1 signatures. Files *process.0x81c47c00.0x80000.dmp* and *process.0x81c498c8.0x80000.dmp* were found to be identical (found below in bold). The other 12 dumped samples were unique with respect to one another.

The *malfind*-dumped memory samples were then fuzzy hashed against one another to determine their similarities between one another. This analysis revealed that in all there were 19 matches between the memory samples, as described in the following table:

*Table 21: Fuzzy hash matches for Malfind-dumped memory samples (sorted by %).*

| Matched Filename #1 | Matched Filename #2 | Match (in %) |
|---|---|---|
| process.0x81c47c00.0x680000.dmp | process.0x81c47c00.0x1000000.dmp | 49 |
| process.0x81c498c8.0x1000000.dmp | process.0x81c47c00.0x680000.dmp | 49 |

| Matched Filename #1 | Matched Filename #2 | Match (in %) |
|---|---|---|
| process.0x82073020.0x940000.dmp | process.0x81c47c00.0x1000000.dmp | 52 |
| process.0x82073020.0x940000.dmp | process.0x81c498c8.0x1000000.dmp | 52 |
| process.0x81e61da0.0xbf0000.dmp | process.0x81c47c00.0x1000000.dmp | 54 |
| process.0x81e61da0.0xbf0000.dmp | process.0x81c498c8.0x1000000.dmp | 54 |
| process.0x81e61da0.0xd00000.dmp | process.0x81c47c00.0x870000.dmp | 66 |
| process.0x82073020.0x13f0000.dmp | process.0x81e61da0.0xd00000.dmp | 69 |
| process.0x82073020.0x13f0000.dmp | process.0x81c47c00.0x870000.dmp | 71 |
| process.0x81e61da0.0xbf0000.dmp | process.0x81c47c00.0x680000.dmp | 96 |
| process.0x82073020.0x940000.dmp | process.0x81c47c00.0x680000.dmp | 96 |
| process.0x82073020.0x940000.dmp | process.0x81e61da0.0xbf0000.dmp | 96 |
| process.0x81c47c00.0x80000.dmp | process.0x81c47c00.0x6f0000.dmp | 99 |
| process.0x81c498c8.0x80000.dmp | process.0x81c47c00.0x6f0000.dmp | 99 |
| process.0x81e61da0.0xb70000.dmp | process.0x81c47c00.0x6f0000.dmp | 99 |
| process.0x81e61da0.0xb70000.dmp | process.0x81c47c00.0x80000.dmp | 99 |
| process.0x81e61da0.0xb70000.dmp | process.0x81c498c8.0x80000.dmp | 99 |
| process.0x81c498c8.0x1000000.dmp | process.0x81c47c00.0x1000000.dmp | 100 |
| **process.0x81c498c8.0x80000.dmp** | **process.0x81c47c00.0x80000.dmp** | **100** |

Note, that for some of these files their similarities were very high, with two of them found to be 100% similar. However, their SHA1 hashes tell a different story. Only the last two files listed in the above table (last match) are actually identical. The previous 100% match is very similar, perhaps differing by only a few bytes, but their SHA1 hashes are not identical.

The SHA1 and fuzzy hashes of the *malfind*-dumped memory samples were then compared against those of the carved memory data files. No identical SHA1 hashes were established but two fuzzy hash matches were found:

```
./malfind/process.0x81c47c00.0x1000000.dmp          matches
./carving/f0219248.dll (58)

./malfind/process.0x81c498c8.0x1000000.dmp          matches
./carving/f0219248.dll (58)
```

Finally, the SHA1 hashes were compared against the NSRL 2.41 hash-set but no matches were identified.

#### 2.3.3.2.5 Summary

The *malfind* plugin succeeded in identifying multiple instances of code injection with respect to processes *services.exe*, *lsass.exe*, *svchost.exe* and *lsass.exe* (PIDs 668, 868, 940 and 1928, respectively).

Moreover, an identical match was established (see Table 22 for details) between two *malfind*-dumped samples. Furthermore, various partial and nearly identical matches were obtained between many of the dumped samples. Finally, two dumped samples were found partially matching some of the carved data memory files.

In short, eight samples were found to be infected representing four specific processes, all of which are associated with PID 668 (*services.exe*).

### 2.3.3.3 Dlldump plugin

Based on the evidence and information established thus far, it is clear that PID 668 is the root of the infection. Moreover, it appears that PIDs 668, 868, 940 and 1928 are infected or have been compromised through code injection.

Using the *dlldump* plugin, it will possible to dump all DLLs still resident in memory associated with suspicious processes to disk for further analysis.

#### 2.3.3.3.1 Running the plugin

Volatility's *dlldump* plugin was specifically designed to dump DLLs from memory to disk. If memory address offsets are specified then they must be physical memory addresses.

The following command was issued to dump DLLs from PIDs 668, 868, 940 and 1928:

```
$ volatility -f stuxnet.vmem -p 668,868,940,1928 --dump-
dir=dlldump
```

Upon running this command, the following numbers of DLLs were dumped for the specified PIDs, as per the following table:

*Table 22: Number of DLLs dumped per specified PID for Dlldump plugin (sorted by PID).*

| PID dumped | Number of dumped DLLs |
|---|---|
| 868 | 8 |
| 668 | 45 |
| 940 | 44 |
| 1928 | 28 |

In all, 125 DLLs were successfully dumped to disk and none with none of the DLLs having been paged out from memory. The textual results of this command are found in Annex E.

#### 2.3.3.3.2 AV scanning and result analysis

All 125 samples were scanned using the six aforementioned scanners. Of these samples, eight were found infected by one or more scanners and many multi-scanner matches were established between samples, as shown in the following colour-coded table:

*Table 23: Scanners results for dumped Dlldump-based memory samples (sorted by scanner).*

| Scanner | Filename | Infection Identification | Matches |
|---|---|---|---|
| Avast | module.668.2273020.13f0000.dll | Win32:Duqu-F [Rtk]<br>Win32:Stuxnet-C [Wrm]<br>Win32:Malware-gen<br>Win32:StuxX-A [Wrm]<br>Win32:Stuxnet [Rtk]<br>Win32:Duqu-F [Rtk] | Match 1 |
| | module.940.2061da0.d00000.dll | Win32:Duqu-F [Rtk]<br>Win32:Stuxnet-C [Wrm]<br>Win32:Malware-gen<br>Win32:StuxX-A [Wrm]<br>Win32:Stuxnet [Rtk]<br>Win32:Malware-gen<br>Win32:Duqu-F [Rtk] | Match 2 |
| | module.1928.1e47c00.1000000.dll | Win32:Duqu-F [Rtk] | Match 3 |
| | module.1928.1e47c00.870000.dll | Win32:Duqu-F [Rtk]<br>Win32:Stuxnet-C [Wrm]<br>Win32:Malware-gen<br>Win32:StuxX-A [Wrm]<br>Win32:Stuxnet [Rtk]<br>Win32:Malware-gen<br>Win32:Duqu-F [Rtk] | Match 4 |
| | module.868.1e498c8.1000000.dll | Win32:Duqu-F [Rtk] | Match 5 |
| AVG | module.1928.1e47c00.1000000.dll | Trojan horse Duqu.A | Match 3 |
| | module.1928.1e47c00.870000.dll | Trojan horse Generic_r.OD<br>Trojan horse Generic_r.OD.dropper | Match 4 |
| | module.668.2273020.13f0000.dll | Trojan horse Generic_r.OD<br>Trojan horse Generic_r.OD.dropper | Match 1 |
| | module.868.1e498c8.1000000.dll | Trojan horse Duqu.A | Match 5 |
| | module.940.2061da0.d00000.dll | Trojan horse Generic_r.OD<br>Trojan horse Generic_r.OD.dropper | Match 2 |

| Scanner | Filename | Infection Identification | Matches |
|---|---|---|---|
| BitDefender | module.1928.1e47c00.1000000.dll | Trojan.Generic.KDV.649803 | Match 3 |
| | module.1928.1e47c00.870000.dll | Trojan.Generic.KDV.564268 | Match 4 |
| | module.868.1e498c8.1000000.dll | Trojan.Generic.KDV.649803 | Match 5 |
| | module.940.2061da0.d00000.dll | Trojan.Generic.KDV.786223 | Match 2 |
| Comodo | module.1928.1e47c00.1000000.dll | Malware | Match 3 |
| | module.1928.1e47c00.870000.dll | Malware | Match 4 |
| | module.668.2273020.1020000.dll | Malware | |
| | module.868.1e498c8.1000000.dll | Malware | Match 5 |
| | module.940.2061da0.d00000.dll | Malware | Match 2 |
| FRISK | module.668.2273020.13f0000.dll | <W32/Dropper.gen8!Maximus> | Match 1 |
| | module.1928.1e47c00.870000.dll | <W32/Dropper.gen8!Maximus> | Match 4 |
| | module.940.2061da0.d00000.dll | <W32/Dropper.gen8!Maximus> | Match 2 |
| McAfee | module.668.2273020.13f0000.dll | Stuxnet trojan | Match 1 |
| | module.1928.1e47c00.870000.dll | Stuxnet trojan | Match 4 |
| | module.940.2061da0.d00000.dll | Stuxnet trojan | Match 2 |

In the above listed malware identified by the various scanners, five matches in all were established between the scanners. Some dumped DLLs were more readily detected by the various scanners than others, and one specific dumped DLL was only detected by one scanner, *module.668.2273020.1020000.dll* by Comodo. Although a *strings* analysis did not directly reveal that this file was infected or malicious, the following table reveals the association between the dumped DLL memory samples, their associated PIDs and the DLL names they represent.

*Table 24: Association between scanner-identified malware for Dlldump-based memory samples and PID/Process name (sorted by PID).*

| Process Name | DLL/EXE Name | PID | Dlldump Memory Sample Name |
|---|---|---|---|
| services.exe | xpsp2res.dll | 668 | module.668.2273020.1020000.dll |
| services.exe | KERNEL32.DLL.ASLR.0360c5e2 | 668 | module.668.2273020.13f0000.dll |
| svchost.exe | KERNEL32.DLL.ASLR.0360c8ee | 940 | module.940.2061da0.d00000.dll |
| lsass.exe | lsass.exe | 868 | module.868.1e498c8.1000000.dll |
| lsass.exe | lsass.exe | 1928 | module.1928.1e47c00.1000000.dll |
| lsass.exe | KERNEL32.DLL.ASLR.0360b7ab | 1928 | module.1928.1e47c00.870000.dll |

Based on these two tables, all dumped DLL memory samples detected by the various scanners as infected, with the exception of sample *module.668.2273020.1020000.dll*, fits the current set of facts. It is very likely that sample *module.668.2273020.1020000.dll* is a false positive as there is currently no indication that file *xpsp2res.dll* is infected, as based on the evidence obtained thus far in this investigation.

Note that the three suspicious DLL files *KERNEL32.DLL.ASLR** have been accurately identified as malicious by the various scanners. Moreover, even the two suspicious instances of *lsass.exe* (PIDs 868 and 1928) were directly identified as infected by the various scanners as per the information determined through Table 24.

#### 2.3.3.3.3 SHA1 and fuzzy hashes

All 125 *dlldump*-based memory samples were hashed using the *sha1sum* command to identify similarities between each other and the carved memory data files. Files *module.868.1e498c8.77fe0000.dll* and *module.1928.1e47c00.77fe0000.dll* were found to be identical. The remaining 123 DLLs were found to be distinct from one another.

The SHA1 hashes were then compared against the NSRL but no matches were established. The hashes were then compared against the carved memory data files and again no matches were identified.

Fuzzy hashing was then carried out between the dumped DLL memory samples to identify similarities. In all, six 100% matches were established through fuzzy hashing, but because only two SHA1 hashes were identical (*module.868.1e498c8.77fe0000.dll* and *module.1928.1e47c00.77fe0000.dll*), the other four fuzzy hash matches were very similar, perhaps differing only by a few bytes. A full listing of these matches is available in Annex F.1. In all, 103 matches and partial matches were identified.

Finally, fuzzy hash matching was conducted between the dumped memory samples and the carved memory data files. No identical matches were found. However, in all, 61 partial matches were established. A full listing of these matches is available in Annex F.2.

#### 2.3.3.3.4 Summary

As established through DLL-based dumping and analysis of suspicious processes, it has been determined that the Stuxnet infection had infected four specific processes, specifically PIDs 668, 868, 940 and 1928. As portrayed in Table 24, there is little doubt concerning which DLLs and executables were directly involved in the infection, with the exception of *xpsp2res.dll* which was a false positive.

Moreover, coupled with the information obtained using the *malfind* plugin and its subsequent analyses, these four processes contain malicious DLLs. The information and evidence thus paints a telling story of this particular infection.

Although some may have preferred to dump all drivers from the memory image and then validate them through AV scanning and hash analysis, this would have introduced a great deal of analytical overhead. Moreover, the information and evidence established thus far has not indicated that this was a necessary investigative endeavour.

### 2.3.3.4 Moddump plugin

Now that the presence of injected code and malicious DLLs has been established for this memory image, it is time to dump and analyse the two suspicious device drivers identified in Step 2. These two drivers are using names very similar to known Windows device drivers.

#### 2.3.3.4.1 Running the plugin

Volatility's *moddump* plugin was specifically designed to dump drivers from memory to disk. If memory address offsets are specified then the *Start* address found in Table 14 obtained from the *driverscan* plugin should be used.

To dump drivers *MRxCLS* and *MRxNet* from the memory image, the following two commands were issued:

```
$ volatility -f stuxnet.vmem moddump -b 0xf895a000 --dump-
dir=moddump

$ volatility -f stuxnet.vmem moddump -b 0xb21d8000 --dump-
dir=moddump
```

The two dumped drivers, files *driver.f895a000.sys* and *driver.b21d8000.sys* had the following metadata:

*Table 25: Metadata concerning Moddump-based driver MRxCls.*

| | |
|---|---|
| **Filename** | moddump/driver.f895a000.sys |
| **Size** | 19,840 bytes |
| **SHA1 hash** | a83a1b3d565611d68a3ab8b93648d30bf715f56a |
| **Fuzzy hash** | 384:GHjgXHujOpb6Rl3qdcr7mj3eSW0lGYaWd7pxW3KzM:5XHLpO/+0mTeSJ/7p+K |

*Table 26: Metadata concerning Moddump-based driver MRxNet.*

| | |
|---|---|
| **Filename** | moddump/driver.b21d8000.sys |
| **Size** | 10,624 bytes |
| **SHA1 hash** | 7918300a71a9c5bf55fbe95b93fd8d2b79a7cf97 |
| **Fuzzy hash** | 96:myL+XFVckoY+H0Si6R2HOzopMZG4+oGYZsYoDKMRV7/tsM/JfNvRc7d5DNSdMe4:mg+X5/OzoIVFZsheCVxNZUDNTL |

#### 2.3.3.4.2 AV scanning

Using the aforementioned AV scanners against the dumped drivers, four and two of the six scanners detected them as infected, respectively. Specifics are listed in the following table:

*Table 27: AV scanner detection of Moddump-based driver MRxCls.*

| Scanner | Detected as |
|---|---|
| Avast | Win32:Duqu-K [Rtk] |
| AVG | Trojan horse Rootkit-Pakes.AJ |
| **Scanner** | **Detected as** |
| BitDefender | Rootkit.51232 |
| Comodo | TrojWare.Win32.Rootkit.Stuxnet.A |
| F-Prot | N/A |
| Mcafee | N/A |

*Table 28: AV scanner detection of Moddump-based driver MRxNet.*

| Scanner | Detected as |
|---|---|
| Avast | N/A |
| AVG | N/A |
| BitDefender | Trojan.Generic.6534646 |
| Comodo | TrojWare.Win32.Rootkit.Stuxnet.B |
| F-Prot | N/A |
| Mcafee | N/A |

Based on the scanner analyses, these two drivers are associated with the infection. Although driver *MRxCls* was detected by four of the six scanners whereas driver *MRxNet* was identified by only two of the scanners, the information provided by these scanners ties them in with the *malfind*and *dlldump*dumped samples thus far established.

#### 2.3.3.4.3 SHA1 and fuzzy hashes

The SHA1 hashes for the two malicious drivers were compared against those of the NSRL and the carved memory data files but no matches were identified.

Fuzzy hash comparisons against the carved memory data files yielded two partial matches:

```
./moddump/driver.b21d8000.sys 41% match ./carving/f0174648.exe

./moddump/driver.f895a000.sys 36% match./carving/f0933680.exe
```

#### 2.3.3.4.4 Summary

The two suspicious drivers identified in Step 2 have been found to not only be malicious but are part of the infection plaguing this memory image.

#### 2.3.3.5 Summary and analysis

This step confirms the variously identified suspicious processes (Step 1) and files, DLLs, drivers, and threads (Step 2). The existence of injected code can be inferred from the presence of suspicious DLLs coupled with the fact that suspicious handles, threads, and mutexes have been identified. Whether these inferences are certain will not be known until a full reverse engineering effort is undertaken including a full analysis of the underlying disk image.

The objective of the two device drivers is not yet known although it is likely they play a role in code injection. To better understand the role of the drivers a reverse engineering is required.

One false positive was found in file *xpsp2res.dll* identified solely by Comodo.

Finally, it has been ascertained that process *services.exe* (PID 668) is at the heart of the infection and that one malicious DLL (*KERNEL32.DLL.ASLR.0360c5e2*) was subverting it through code injection. The other affected processes PIDs 868, 940 and 1928 followed suite.

### 2.3.4 Step 4: Registry

The Windows registry serves to both complicate and facilitate the investigator's work. It is commonly used by malware to configure system settings for permanent infection. However, the difficulty in working with the registry lies in knowing where to look. The registry is spread out across many data files (commonly known as registry hives) in various locations and each serves a specific purpose with respect to system, application and user configurations. Annex G provides a listing of registry keys commonly used by malware. The list has had several entries added to it since report [3].

#### 2.3.4.1 Hivelist plugin

The purpose of using the *hivelist* plugin is to determine which registry hives[3] are available in the memory image.

Consider the plugin's output, using command "*volatility -f stuxnet.vmem hivelist*":

*Table 29: Volatility Hivelist plugin output.*

| Virtual Address | Physical Address | Filename and Location |
|---|---|---|
| 0xe1069008 | 0x14b8d008 | \Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat |
| 0xe1077758 | 0x152b7758 | \Device\HarddiskVolume1\Documents and Settings\Administrator\NTUSER.DAT |
| 0xe1bdb9e8 | 0x0e1959e8 | \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat |

[3] A registry hive denotes the actual disk file and its location on disk.

| Virtual Address | Physical Address | Filename and Location |
|---|---|---|
| 0xe1bd5b60 | 0x0e027b60 | \Device\HarddiskVolume1\Documents and Settings\LocalService\NTUSER.DAT |
| 0xe1bc26d8 | 0x0de626d8 | \Device\HarddiskVolume1\Documents and Settings\NetworkService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat |
| 0xe1bb5758 | 0x0df10758 | \Device\HarddiskVolume1\Documents and Settings\NetworkService\NTUSER.DAT |
| 0xe1628b60 | 0x0a768b60 | \Device\HarddiskVolume1\WINDOWS\system32\config\software |
| 0xe16386b8 | 0x0a7a06b8 | \Device\HarddiskVolume1\WINDOWS\system32\config\default |
| 0xe1638b60 | 0x0a7a0b60 | \Device\HarddiskVolume1\WINDOWS\system32\config\SAM |
| 0xe1628008 | 0x0a768008 | \Device\HarddiskVolume1\WINDOWS\system32\config\SECURITY |
| 0xe13feb60 | 0x02e6ab60 | [no name] |
| 0xe1035b60 | 0x02a9eb60 | \Device\HarddiskVolume1\WINDOWS\system32\config\system |
| 0xe102e008 | 0x02a98008 | [no name] |
| 0x80670a0c | 0x00670a0c | [no name] |

#### 2.3.4.2 Printkey plugin

Using all proposed registry keys identified in Annex E, 1120 Volatility *printkey* commands were issued via a script to query the memory image for information pertaining to traces of this malware's activities. Building such a script takes only a few minutes. Based on the physical memory addresses listed in the above table, used in conjunction with various command line tools including *cat*, *awk* and *sed*, it is quickly assembled.

All output generated by the script was captured and stored to a text file for subsequent analysis.

After running the script, the following pertinent information was identified:

```
Registry: User Specified
Key name: MRxNet (S)
Last updated: 2011-06-03 04:26:47

Subkeys:
  (V) Enum

Values:
REG_SZ       Description    : (S) MRXNET
REG_SZ       DisplayName    : (S) MRXNET
REG_DWORD    ErrorControl   : (S) 0
REG_SZ       Group          : (S) Network
REG_SZ       ImagePath      : (S) \??\C:\WINDOWS\system32\Drivers\mrxnet.sys
```

```
REG_DWORD    Start          : (S) 1
REG_DWORD    Type           : (S) 1
Legend: (S) = Stable   (V) = Volatile

----------------------------

Registry: User Specified
Key name: MRxCls (S)
Last updated: 2011-06-03 04:26:47

Subkeys:
  (V) Enum

Values:
REG_SZ       Description    : (S) MRXCLS
REG_SZ       DisplayName    : (S) MRXCLS
REG_DWORD    ErrorControl   : (S) 0
REG_SZ       Group          : (S) Network
REG_SZ       ImagePath      : (S) \??\C:\WINDOWS\system32\Drivers\mrxcls.sys
REG_DWORD    Start          : (S) 1
REG_DWORD    Type           : (S) 1
REG_BINARY   Data           : (S)
0x00000000  8f 1f f7 6d 7d b1 c9 09 9d cc 24 7a c6 9f fb 23   ...m}.....$z...#
0x00000010  90 bd 9d bf f1 d4 51 92 2a b4 1f 6a 2e a6 4f b3   ......Q.*..j..O.
0x00000020  cb 69 7c 0b 92 3b 1b c0 d7 75 17 a9 e3 33 48 dc   .i|..;...u...3H.
0x00000030  ad f6 da ea 2f 87 10 c4 21 81 a5 75 68 00 2e b1   ..../...!..uh...
0x00000040  c2 7b eb dd bb 72 47 dc 87 91 14 a5 f3 c4 32 b0   .{...rG.......2.
0x00000050  cc 93 38 36 6b 49 0a f2 6f 1f 1d a1 4a 15 05 80   ..86kI..o...J...
0x00000060  4b 13 a8 aa 82 41 4b 89 dc 89 24 a2 ed 16 37 f3   K....AK...$...7.
0x00000070  42 a9 a0 6a 7f 82 cd 90 e5 3c 49 cc b2 97 ca cb   B..j.....<I.....
0x00000080  7b 64 c1 48 b2 4c f5 ae 54 42 74 0f 00 31 fd 80   {d.H.L..TBt..1..
0x00000090  e8 7e 0e 69 12 42 3a ec 0f 6f 03 b8 46 9c 68 97   .~.i.B:..o..F.h.
0x000000a0  ac 62 16 fb 1a 1b d9 33 6c e8 f9 93 c3 56 54 a1   .b.....3l....VT.
0x000000b0  89 7a 7b 77 ce ba 0d 95 a7 0f ab 5e 1c 3c 18 63   .z{w.......^.<.c
0x000000c0  ae 3e 60 a6 81 bc fa 85 fb 37 a0 0a 57 f9 c9 d3   .>`......7..W...
0x000000d0  cf 6b 41 d9 6d cd 39 71 c5 11 83 f1 d9 f3 7d b7   .kA.m.9q......}.
0x000000e0  91 f7 70 46 c2 24 f7 b9 0f 2d b2 60 72 1c 8f f9   ..pF.$...-.`r...
0x000000f0  98 16 34 52 4b 7d 5f 81 5f 35 fd 8b 3e 78 b1 0b   ..4RK}_._5..>x..
0x00000100  0a 90 5a d8 30 5a 56 90 9a c0 c1 0f eb 95 d5 2f   ..Z.0ZV......../
0x00000110  b7 c5 8d 2b 3f 49 41 8b 86 b4 db 71 67 69 e6 e8   ...+?IA....qgi..
0x00000120  69 77 29 77 18 82 11 8b d7 5d 26 e4 5a 5c 2c 46   iw)w.....]&.Z\,F
0x00000130  c2 f0 02 28 d8 ea 4b 95 9c 3a 3c 12 da c4 87 21   ...(..K..:<....!
0x00000140  91 4f d0 6e fa c4 dd b7 c9 af e2 ae fe 14 0f 53   .O.n...........S
0x00000150  c4 ba dd 31 1a 38 7b 37 c0 9e 83 ff 2c b2 4c 88   ...1.8{7....,.L.
0x00000160  33 c1 89 e5 ca 68 31 2d 20 ce 50 64 7b 39 c7 fb   3....h1-..Pd{9..
0x00000170  b1 9f a9 0d 6c 2a 82 ae 7f 25 43 a7 a2 28 eb 27   ....l*...%C..(.'
0x00000180  73 c9 45 f9 fd 53 a8 f4 a7 fd b4 90 b2 28 d8 0c   s.E..S.......(..
0x00000190  5a a8 84 d0 7f ed 99 25 18 fe b8 4c 48 66 8d 59   Z......%...LHf.Y
0x000001a0  40 f6 cc 30 a6 f4 04 e8 76 9c ea 0e f6 a4 4a ce   @..0....v.....J.
0x000001b0  d2
```

These registry entries pertain to the two malicious device drivers, *MRxNet* and *MRxCls*. No information could be found concerning malicious DLLs *KERNEL32.DLL.ASLR**, indicating that these DLLs are likely loaded into memory by one or both of these drivers in order to carry out code injection.

Thus, the persistence of this infection was made possible through the Windows registry which loads these two device drivers in order to perpetuate the infection.

#### 2.3.4.3 Userassist plugin

The final registry-based Volatility plugin run against the memory image was *userassist*. This plugin has the potential to provide, among other things, registry-based information pertaining to programs run and files opened by the user.

This plugin identified the following information likely relevant to the infection:

```
REG_BINARY                  UEME_RUNPATH:C:\Documents    and
Settings\Administrator\Desktop\74ddc49a7c121a61b8d06c03f92d
0c13.exe :
ID:             6
Count:          1
Last updated:   2011-06-03 04:26:46
0x00000000  06 00 00 00 06 00 00 00 80 1e e0 72 a6 21 cc 01
...........r.!..
```

This UserAssist key is reminiscent of the output identified by the filescan plugin in Step 2. This highly suspicious executable is likely the malware dropper responsible for the infection.

### 2.3.5 Step 5: Miscellaneous

This final step examines two additional lines of inquiry, although they are optional.

Specifically, it may be possible to determine if encryption was used by the malware to secure its communications and to identify specifics concerning the two malicious device drivers.

#### 2.3.5.1 Devicetree

The Volatility *devicetree* plugin is used to determine the relationship between drivers and their required Windows devices. In so doing, it may be possible to determine what device, and hence purpose, of a malicious device driver.

Running command “*volatility -f stuxnet.vmem devicetree*,” after pruning, generated the following output:

```
DRV 0x0205e5a8 \FileSystem\vmhgfs
---| DEV 0x820f0030 hgfsInternal UNKNOWN
---| DEV 0x821a1030 HGFS FILE_DEVICE_NETWORK_FILE_SYSTEM
------|   ATT   0x81f5d020   HGFS   -   \FileSystem\FltMgr
FILE_DEVICE_NETWORK_FILE_SYSTEM
```

```
---------|   ATT   0x821354b8   HGFS   -   \Driver\MRxNet
FILE_DEVICE_NETWORK_FILE_SYSTEM


DRV 0x020d2f38 \FileSystem\FltMgr
---| DEV 0x8206b628  FILE_DEVICE_CD_ROM_FILE_SYSTEM
---| DEV 0x81ead318  FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x81f47020  FILE_DEVICE_DISK_FILE_SYSTEM
------|     ATT     0x81fb9680          -     \Driver\MRxNet
FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x81e859c8  FILE_DEVICE_DISK_FILE_SYSTEM
------|     ATT     0x81f0ab90          -     \Driver\MRxNet
FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x81fac548  FILE_DEVICE_CD_ROM_FILE_SYSTEM
------|     ATT     0x8226ef10          -     \Driver\MRxNet
FILE_DEVICE_CD_ROM_FILE_SYSTEM
---| DEV 0x81f5d020  FILE_DEVICE_NETWORK_FILE_SYSTEM
------|     ATT     0x821354b8          -     \Driver\MRxNet
FILE_DEVICE_NETWORK_FILE_SYSTEM
---| DEV 0x81bf1020  FILE_DEVICE_NETWORK_FILE_SYSTEM
------|     ATT     0x81f0fc58          -     \Driver\MRxNet
FILE_DEVICE_NETWORK_FILE_SYSTEM
---| DEV 0x82135d10  FILE_DEVICE_NETWORK_FILE_SYSTEM
------|     ATT     0x81c0a910          -     \Driver\MRxNet
FILE_DEVICE_NETWORK_FILE_SYSTEM
---| DEV 0x8226ccd0 FltMgrMsg UNKNOWN
---| DEV 0x8233d390 FltMgr FILE_DEVICE_DISK_FILE_SYSTEM


DRV 0x02126870 \Driver\MRxCls
---| DEV 0x81bdbeb0 MRxClsDvX FILE_DEVICE_UNKNOWN


DRV 0x02296b20 \FileSystem\sr
---| DEV 0x8228c6b0  FILE_DEVICE_DISK_FILE_SYSTEM
------|    ATT    0x81f47020         -    \FileSystem\FltMgr
FILE_DEVICE_DISK_FILE_SYSTEM
---------|    ATT    0x81fb9680         -    \Driver\MRxNet
FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x81eecdd0  FILE_DEVICE_DISK_FILE_SYSTEM
------|    ATT    0x81e859c8         -    \FileSystem\FltMgr
FILE_DEVICE_DISK_FILE_SYSTEM
---------|    ATT    0x81f0ab90         -    \Driver\MRxNet
FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x823df450 SystemRestore FILE_DEVICE_UNKNOWN


DRV 0x022e1c08 \FileSystem\MRxDAV
---|        DEV        0x81caca58        WebDavRedirector
FILE_DEVICE_NETWORK_FILE_SYSTEM
------|     ATT     0x82135d10     WebDavRedirector     -
\FileSystem\FltMgr FILE_DEVICE_NETWORK_FILE_SYSTEM
---------| ATT 0x81c0a910 WebDavRedirector - \Driver\MRxNet
FILE_DEVICE_NETWORK_FILE_SYSTEM
```

```
DRV 0x022e54f8 \Driver\MRxNet
---| DEV 0x82125f10  FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x81dc49c0  FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x81fd59c0  FILE_DEVICE_CD_ROM_FILE_SYSTEM
---| DEV 0x81c8b500  FILE_DEVICE_CD_ROM_FILE_SYSTEM
---| DEV 0x821354b8  FILE_DEVICE_NETWORK_FILE_SYSTEM
---| DEV 0x81f0fc58  FILE_DEVICE_NETWORK_FILE_SYSTEM
---| DEV 0x81c0a910  FILE_DEVICE_NETWORK_FILE_SYSTEM
---| DEV 0x8226ef10  FILE_DEVICE_CD_ROM_FILE_SYSTEM
---| DEV 0x81f0ab90  FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x81fb9680  FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x82104700  FILE_DEVICE_DISK_FILE_SYSTEM


DRV 0x023ae880 \FileSystem\MRxSmb
---|       DEV       0x81da95d0       LanmanDatagramReceiver
FILE_DEVICE_NETWORK_BROWSER
---|         DEV         0x81ee5030         LanmanRedirector
FILE_DEVICE_NETWORK_FILE_SYSTEM
------|      ATT      0x81bf1020      LanmanRedirector      -
\FileSystem\FltMgr FILE_DEVICE_NETWORK_FILE_SYSTEM
---------| ATT 0x81f0fc58 LanmanRedirector - \Driver\MRxNet
FILE_DEVICE_NETWORK_FILE_SYSTEM


DRV 0x02476da0 \FileSystem\Cdfs
---| DEV 0x81e636c8 Cdfs FILE_DEVICE_CD_ROM_FILE_SYSTEM
------|    ATT    0x81fac548    Cdfs    -    \FileSystem\FltMgr
FILE_DEVICE_CD_ROM_FILE_SYSTEM
---------|    ATT    0x8226ef10    Cdfs    -    \Driver\MRxNet
FILE_DEVICE_CD_ROM_FILE_SYSTEM


DRV 0x02526f38 \FileSystem\Fs_Rec
---|         DEV         0x8205ac78        FatCdRomRecognizer
FILE_DEVICE_CD_ROM_FILE_SYSTEM
------| ATT 0x81c8b500 FatCdRomRecognizer - \Driver\MRxNet
FILE_DEVICE_CD_ROM_FILE_SYSTEM
---|         DEV         0x81d9e5c0         FatDiskRecognizer
FILE_DEVICE_DISK_FILE_SYSTEM
------| ATT 0x81dc49c0 FatDiskRecognizer - \Driver\MRxNet
FILE_DEVICE_DISK_FILE_SYSTEM
---|         DEV         0x81d9ef08        UdfsDiskRecognizer
FILE_DEVICE_DISK_FILE_SYSTEM
------| ATT 0x82125f10 UdfsDiskRecognizer - \Driver\MRxNet
FILE_DEVICE_DISK_FILE_SYSTEM
---|         DEV         0x81e5d428        UdfsCdRomRecognizer
FILE_DEVICE_CD_ROM_FILE_SYSTEM
------| ATT 0x81fd59c0 UdfsCdRomRecognizer - \Driver\MRxNet
FILE_DEVICE_CD_ROM_FILE_SYSTEM
---|          DEV          0x81e63ae0          CdfsRecognizer
FILE_DEVICE_CD_ROM_FILE_SYSTEM
```

```
DRV 0x0253d180 \FileSystem\Ntfs
---| DEV 0x82166020  FILE_DEVICE_DISK_FILE_SYSTEM
------|     ATT     0x8228c6b0          -     \FileSystem\sr
FILE_DEVICE_DISK_FILE_SYSTEM
---------|    ATT    0x81f47020       -    \FileSystem\FltMgr
FILE_DEVICE_DISK_FILE_SYSTEM
------------|    ATT    0x81fb9680       -    \Driver\MRxNet
FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x8224f790 Ntfs FILE_DEVICE_DISK_FILE_SYSTEM
------|    ATT    0x81eecdd0    Ntfs    -    \FileSystem\sr
FILE_DEVICE_DISK_FILE_SYSTEM
---------|   ATT   0x81e859c8   Ntfs   -   \FileSystem\FltMgr
FILE_DEVICE_DISK_FILE_SYSTEM
------------|   ATT   0x81f0ab90   Ntfs   -   \Driver\MRxNet
FILE_DEVICE_DISK_FILE_SYSTEM
```

The above output demonstrates how pervasive the Stuxnet infection was. The infection has embedded itself into many various filesystem and network related system services. Moreover, device driver *MRxNet* appears to redefine the system's usage of disk, network and optical devices, probably for the purposes of hiding data and spreading the infection. This specific device driver is particularly prevalent.

Device driver *MRxCls* make use of some unknown device whose purpose is not entirely clear at this point.

#### 2.3.5.2 Extract encryption keys

While the documentation detailing the Stuxnet infection provided herein [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18] does not specify the use of encryption keys, nevertheless an AES 256-bit key was identified within this memory image. Of course, it is possible that the Stuxnet infection does not use encryption and that it is the result of some other application that was in use within this memory image.

Two readily useable FOSS-based encryption detection and extraction tools include *aeskeyfind*[4] and *interrogate*[5]. Both tools are easy to use. Running either command will reveal that an AES encryption key that was in use is readily identifiable and has been identified as:

```
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1
d1e1f
```

#### 2.3.5.3 Summary and analysis

Although this step was brief, it was demonstrated that AES encryption can be detected and extracted from memory. However, it cannot be readily confirmed whether the Stuxnet infection made use of AES encryption.

[4] Aeskeyfind can be found at https://citp.princeton.edu/research/memory/code/.
[5] Interrogate can be found at https://github.com/carmaa/interrogate.

Finally, using the Volatility *devicetree* plugin it was possible to determine that malicious device driver *MRxNet* made significant changes to the operating system's use of file, network and optical filesystems. Moreover, device driver *MRxCls* introduced an unknown file-based device of unknown capability.

# 3 Conclusion

It can be concluded from this work that using sound investigative footwork, combined with the capabilities of the Volatility memory analysis framework, investigators can readily analyse and investigate suspected memory-based infections.

The Stuxnet worm is a persistent, readily replicating advanced malware. Its intentions are known - it actively seeks out and targets SCADA-based systems. However, its origins are not known with certainty. The malware has demonstrated its ability to infect multiple processes and take hold of the operating system soon after system boot through the loading of two malicious device drivers.

In comparison to past investigations conducted by the author in this series of reports, Stuxnet was by far the most complex to analyse. However, it did not make any particular effort to hide itself as it left two highly suspicious processes (*lsass.exe*) running which had the potential to draw unwanted attention to the infection. As advanced as the claims concerning this infection have made it out to be, it did not succeed in camouflaging itself very well.

Throughout this document, based on the clarified methodology put forward in Section 1.8, the author has demonstrated the manner in which a forensic memory analysis can be conducted by non-memory specialists. Thus, even novice memory investigators can successfully conduct complex memory analyses, when equipped with a straightforward methodology, techniques and tools.

Although much information was available concerning Stuxnet, investigators will not always be able to rely on such well-prepared reports. This is why this investigation did not make direct use of them during the analysis of this memory image. The techniques and methodology presented herein will be of use, to varying extents, against newer and more difficult to analyse malware.

This document is the fourth in a series of many. It is hoped that subsequent reports will be possible in order to continue building a sufficient compendium of knowledge for memory analysis for use by novice and expert memory analysts alike. While the degree of difficulty varies substantially from case to case, the Volatility framework, when combined with investigative knowhow, tools, techniques and methodology is a highly adept analysis-based framework.

# References

[1] Carbone, Richard. Malware memory analysis for non-specialists: Investigating a publicly available memory image of the Zeus Trojan horse. Technical Memorandum. Defence R&D Canada – Valcartier. TM 2013-018. April 2013.

[2] Carbone, Richard. Malware memory analysis for non-specialists: Investigating publicly available memory images for Prolaco and SpyEye. Technical Memorandum. Defence R&D Canada – Valcartier. TM 2013-155. October 2013.

[3] Carbone, Richard. Malware memory analysis for non-specialists: Investigating publicly available memory image 0zapftis (R2D2). Technical Memorandum. Defence R&D Canada – Valcartier. TM 2013-177. October 2013.

[4] Carbone, Richard. File recovery and data extraction using automated data recovery tools: A balanced approach using Windows and Linux when working with an unknown disk image and filesystem. Technical Memorandum. TM 2009-161. Defence R&D Canada – Valcartier. January 2013. http://cradpdf.drdc-rddc.gc.ca/PDFS/unc122/p531895_A1b.pdf.

[5] Kushner, David. The Real Story of Stuxnet: How Kaspersky Lab tracked down the malware that stymied Iran's nuclear-fuel enrichment program. Online article. IEEE Spectrum Magazine. February 2013. http://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet.

[6] Wikipedia. Stuxnet. Online encyclopaedic entry. Wikipedia. Wikimedia Foundation Inc. October 2013. http://en.wikipedia.org/wiki/Stuxnet.

[7] Wikipedia. VirusBlokAda. Online encyclopaedic entry. Wikipedia. Wikimedia Foundation Inc. May 2013. http://en.wikipedia.org/wiki/VirusBlokAda.

[8] Keizer, Gregg. Is Stuxnet the 'best' malware ever? Online article. Infoworld.com. September 2010. http://www.infoworld.com/print/137598.

[9] Albright, David; Brannan, Paul and Walrond, Christina. Did Stuxnet Take Out 1,000 Centrifuges at the Natanz Enrichment Plant? Assessment/Position paper. Institute for Science and International Security. December 2010. http://isis-online.org/uploads/isis-reports/documents/stuxnet_FEP_22Dec2010.pdf.

[10] Albright, David; Brannan, Paul and Walrond, Christina. Stuxnet Malware and Natanz: Update of ISIS December 22, 2010 Report. Assessment/Position paper. Institute for Science and International Security. February 2011. http://isis-online.org/uploads/isis-reports/documents/stuxnet_update_15Feb2011.pdf.

[11] Matrosov, Aleksandr; Rodionov, Eugene, et al. Stuxnet Under the Microscope. Technical report. Revision 1.31. Unknown date. http://www.eset.com/us/resources/white-papers/Stuxnet_Under_the_Microscope.pdf.

[12] Falliere, Nicolas; O Murchu, Liam, and Chien, Eric. W32.Stuxnet Dossier. Technical report. Version 1.4. Symantec. February 2011. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.

[13] Cyber Security Forum Initiative. Preliminary Stuxnet Report v.1.0. Technical report. Version 1.0. Cyber Security Forum Initiative. Unknown date. http://www.iamit.org/blog/wp-content/uploads/2010/10/CSFI_Stuxnet_Report_V1.pdf.

[14] Mueller, Paul and Yadegari, Babak. The Stuxnet Worm. Technical report. Department of Computer Science, University of Arizona. April 2012. http://www.cs.arizona.edu/~collberg/Teaching/466-566/2012/Resources/presentations/2012/topic9-final/report.pdf.

[15] McDonald, Geoff; O Murchu, Liam; et al. Stuxnet 0.5: The Missing Link. Technical report. Symantec. Unknown date. http://www2.gwu.edu/~nsarchiv/NSAEBB/NSAEBB424/docs/Cyber-088.pdf.

[16] Byres, Eric; Ginter, Andrew, and Langill, Joel. How Stuxnet Spreads – A Study of Infection Paths in Best Practice Systems. White paper. Tofino Security, Abterra Technologies and ScadaHacker.com. February 2011. http://abterra.ca/papers/How-Stuxnet-Spreads.pdf.

[17] Ginter, Andrew. The Stuxnet Worm and Options for Remediation. Technical report. Industrial Defender. August 2010. https://www.scadahacker.com/library/Documents/ICS_Events/Stuxnet%20Worm%20and%20Options%20for%20Remediation%20(Industrial%20Defender).pdf.

[18] Thabet, Amr. Stuxnet Malware Analysis Paper. Technical report. Codeproject.com. Unknown date. http://www.codeproject.com/KB/web-security/StuxnetMalware/Stuxnet_Malware_Analysis_Paper.pdf.

[19] Volatility. CommandReference: Example usage cases and output for Volatility 2.0 commands. Online command reference. Volatility. February 2012. http://code.google.com/p/volatility/wiki/CommandReference.

[20] AnswersThatWork. List of Common TCP/IP port numbers. Technical reference. AnswersThatWork.com. September 2008. http://www.answersthatwork.com/Download_Area/ATW_Library/Networking/Network__2-List_of_Common_TCPIP_port_numbers.pdf.

[21] Microsoft TechNet. Network Ports Used by Key Microsoft Server Products. Support article. Microsoft. 2013. http://technet.microsoft.com/en-us/library/cc875824.aspx.

[22] Microsoft TechNet. Port Assignment for Commonly-Used Services. Support article. Microsoft. 2013. http://technet.microsoft.com/en-us/library/cc959833.aspx.

[23] Oracle. Oracle User Messaging Service. Chapter 56: Oracle Fusion Middleware Developer's guide for Oracle SOA Suite 11g Release 1 (11.1.1). Oracle product documentation. Oracle. http://docs.oracle.com/cd/E15523_01/integration.1111/e10224/ns_intro.htm.

[24] Microsoft Support. Exchange Server static port mappings. Support article. Microsoft. 2013. http://support.microsoft.com/kb/270836.

[25] Wikipedia. List of TCP and UDP port numbers. Online encyclopaedic entry. Wikimedia Foundation Inc. October 2013. http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers.

[26] OTN Community. JQS "Java Quick Starter." Technical blog/question and answer. March 2009. Oracle.com. https://forums.oracle.com/thread/1240373.

This page intentionally left blank.

# Annex A Volatility Windows-based plugins

The following is a complete list of the default Windows-based plugins provided with Volatility version 2.2:

*Table 30: List of Volatility 2.2 plugins.*

| **Plugin** | **Capability (as per *Volatility -help* output)** |
|---|---|
| apihooks | Detect API hooks in process and kernel memory |
| atoms | Print session and window station atom tables |
| atomscan | Pool scanner for _RTL_ATOM_TABLE |
| bioskbd | Reads the keyboard buffer from Real Mode memory |
| callbacks | Print system-wide notification routines |
| clipboard | Extract the contents of the windows clipboard |
| cmdscan | Extract command history by scanning for _COMMAND_HISTORY |
| connections | Print list of open connections [Windows XP and 2003 Only] |
| connscan | Scan Physical memory for _TCPT_OBJECT objects (tcp connections) |
| consoles | Extract command history by scanning for _CONSOLE_INFORMATION |
| crashinfo | Dump crash-dump information |
| deskscan | Poolscaner for tagDESKTOP (desktops) |
| devicetree | Show device tree |
| dlldump | Dump DLLs from a process address space |
| dlllist | Print list of loaded dlls for each process |
| driverirp | Driver IRP hook detection |
| driverscan | Scan for driver objects _DRIVER_OBJECT |
| envars | Display process environment variables |
| eventhooks | Print details on windows event hooks |
| evtlogs | Extract Windows Event Logs (XP/2003 only) |
| filescan | Scan Physical memory for _FILE_OBJECT pool allocations |
| gahti | Dump the USER handle type information |
| gditimers | Print installed GDI timers and callbacks |
| gdt | Display Global Descriptor Table |

| **Plugin** | **Capability (as per *Volatility -help* output)** |
|---|---|
| getservicesids | Get the names of services in the Registry and return Calculated SID |
| getsids | Print the SIDs owning each process |
| handles | Print list of open handles for each process |
| hashdump | Dumps passwords hashes (LM/NTLM) from memory |
| hibinfo | Dump hibernation file information |
| hivedump | Prints out a hive |
| hivelist | Print list of registry hives |
| hivescan | Scan Physical memory for _CMHIVE objects (registry hives) |
| idt | Display Interrupt Descriptor Table |
| imagecopy | Copies a physical address space out as a raw DD image |
| imageinfo | Identify information for the image |
| impscan | Scan for calls to imported functions |
| kdbgscan | Search for and dump potential KDBG values |
| kpcrscan | Search for and dump potential KPCR values |
| ldrmodules | Detect unlinked DLLs |
| lsadump | Dump (decrypted) LSA secrets from the registry |
| malfind | Find hidden and injected code |
| memdump | Dump the addressable memory for a process |
| memmap | Print the memory map |
| messagehooks | List desktop and thread window message hooks |
| moddump | Dump a kernel driver to an executable file sample |
| modscan | Scan Physical memory for _LDR_DATA_TABLE_ENTRY objects |
| modules | Print list of loaded modules |
| mutantscan | Scan for mutant objects _KMUTANT |
| patcher | Patches memory based on page scans |
| printkey | Print a registry key, and its subkeys and values |
| procexedump | Dump a process to an executable file sample |
| procmemdump | Dump a process to an executable memory sample |

| **Plugin** | **Capability (as per *Volatility -help* output)** |
|---|---|
| pslist | Print all running processes by following the EPROCESS lists |
| psscan | Scan Physical memory for _EPROCESS pool allocations |
| pstree | Print process list as a tree |
| psxview | Find hidden processes with various process listings |
| raw2dmp | Converts a physical memory sample to a windbg crash dump |
| screenshot | Save a pseudo-screenshot based on GDI windows |
| sessions | List details on _MM_SESSION_SPACE (user logon sessions) |
| shimcache | Parses the Application Compatibility Shim Cache registry key |
| sockets | Print list of open sockets |
| sockscan | Scan Physical memory for _ADDRESS_OBJECT objects (tcp sockets) |
| ssdt | Display SSDT entries |
| strings | Match physical offsets to virtual addresses (may take a while, VERY verbose) |
| svcscan | Scan for Windows services |
| symlinkscan | Scan for symbolic link objects |
| thrdscan | Scan physical memory for _ETHREAD objects |
| threads | Investigate _ETHREAD and _KTHREADs |
| timers | Print kernel timers and associated module DPCs |
| userassist | Print userassist registry keys and information |
| userhandles | Dump the USER handle tables |
| vaddump | Dumps out the vad sections to a file |
| vadinfo | Dump the VAD info |
| vadtree | Walk the VAD tree and display in tree format |
| vadwalk | Walk the VAD tree |
| volshell | Shell in the memory image |
| windows | Print Desktop Windows (verbose details) |
| wintree | Print Z-Order Desktop Windows Tree |
| wndscan | Pool scanner for tagWINDOWSTATION (window stations) |
| yarascan | Scan process or kernel memory with Yara signatures |

This page intentionally left blank.

# Annex B NSRL file hash matches for carved memory data files

This annex provides a listing of those carved memory data files obtained in Section 2.2.3 that matched the SHA1 hashes of the NSRL hash-set 2.41 (June 2013). In total, nineteen unique NSRL SHA1 hashes were found matching the various carved memory data files. However, based on these hashes, it was established that the NSRL contained 52 unique SHA1-filename matches as shown in the following table:

*Table 31: SHA1 hash vs. NSRL filename for carved memory data files.*

| SHA1 | Filename |
|---|---|
| 016C1CE4119A884C002C83D40B3D8B73648E9FC3 | _endian.py.0160FC08_F3D9_4869_9D41_C611C16F42D5 |
| 059EDA50F187D66B3E47A391359099B72576C7A1 | comctl.man |
| 15740B197555BA8E162C37A60BA655151E3BEBAE | index.dat |
| 417F05853C3816F74D6E965694ECA28BCC72AC6F | _0E9D9F5076994D5FA6E423CC70A0C264 |
| 417F05853C3816F74D6E965694ECA28BCC72AC6F | _2DB29F1250A3472AA2BC66491ACE1A5A |
| 417F05853C3816F74D6E965694ECA28BCC72AC6F | _51C3399A8598E1CC1A30AFAB6B273444 |
| 417F05853C3816F74D6E965694ECA28BCC72AC6F | _7354554BF43E4E4D81AA053284C7ECA3 |
| 417F05853C3816F74D6E965694ECA28BCC72AC6F | flavormap.properties |
| 417F05853C3816F74D6E965694ECA28BCC72AC6F | flavormap.properties1 |
| 417F05853C3816F74D6E965694ECA28BCC72AC6F | flavormap.properties.134A883B_933C_41F1_9DC7_7271371486B8 |
| 417F05853C3816F74D6E965694ECA28BCC72AC6F | flavormap.properties.2B708BC3_5B4D_47C0_BCC5_3E1BD2C51E5B |
| 417F05853C3816F74D6E965694ECA28BCC72AC6F | flavormap.properties.2DA786B9_56F1_4FBC_B649_5C7711252559 |
| 417F05853C3816F74D6E965694ECA28BCC72AC6F | PTC24.F |
| 5082B30587F959A74C2BC359502F12454B1697A5 | __0X0050 |
| 59903E96E1EDC257A4850D45AD8C63F17454AE9D | riched32.dll |
| 59903E96E1EDC257A4850D45AD8C63F17454AE9D | RICHED32.DLL |
| 6475D55C14B2DE8F2EDD558C728F1FD41FB63F16 | controls.man |
| 6F9F663CDFBC2592EAB4C43FEE359EFFD37D60F2 | dxgthk.sys |
| 6F9F663CDFBC2592EAB4C43FEE359EFFD37D60F2 | DXGTHK.SYS |
| 80EB8A76E5579B0136281E4DD4E2D4E56B249E4C | null.sys |
| 80EB8A76E5579B0136281E4DD4E2D4E56B249E4C | NULL.SYS |
| 9B4081066DE8FDBEF545D4B5DB62538B2A8A6538 | policy.30729.4974.policy_9_0_Microsoft_VC90_CRT_x86.QFE |

| SHA1 | Filename |
|---|---|
| 9B4081066DE8FDBEF545D4B5DB62538B2A8A6538 | ul_policy.30729.4974.policy_9_0_Microsoft_VC90_CRT_x86.QFE |
| A8139A5A5BCC413090176ECAF41510AA0FFBB987 | Windows Catalog.lnk |
| B70BAFF604434E0485A28660535764C55176C925 | _171A289D2DFB4F40989EDF4E6A83AA76 |
| B70BAFF604434E0485A28660535764C55176C925 | _2A25C71D65FD247CF791F3263F21771E |
| B70BAFF604434E0485A28660535764C55176C925 | _48844EAD7DD64BD486DC283B761DF04A |
| B70BAFF604434E0485A28660535764C55176C925 | _6E6756BD6BF24A588F7AB18B55524BBA |
| B70BAFF604434E0485A28660535764C55176C925 | _7060DD9824C94DA5B35173482221E1DD |
| B70BAFF604434E0485A28660535764C55176C925 | _79DD1B5FA6B548F78FE66C71C084A18B |
| B70BAFF604434E0485A28660535764C55176C925 | cfbddd223bc84ff401e9d37367c36b40 |
| B70BAFF604434E0485A28660535764C55176C925 | cursors.properties |
| B70BAFF604434E0485A28660535764C55176C925 | cursors.properties1 |
| B70BAFF604434E0485A28660535764C55176C925 | cursors.properties.134A883B_933C_41F1_9DC7_7271371486B8 |
| B70BAFF604434E0485A28660535764C55176C925 | cursors.properties.2B708BC3_5B4D_47C0_BCC5_3E1BD2C51E5B |
| B70BAFF604434E0485A28660535764C55176C925 | cursors.properties.2DA786B9_56F1_4FBC_B649_5C7711252559 |
| B70BAFF604434E0485A28660535764C55176C925 | F2978_cursors.properties |
| B70BAFF604434E0485A28660535764C55176C925 | PTC35.F |
| BDB6DB39832DF1DCE10E8050E04AD3FCECCCFA30 | __0X0054 |
| C75D4C6E53A497C4DC1DF1F50BBEF08AC625A3D8 | hosts |
| C75D4C6E53A497C4DC1DF1F50BBEF08AC625A3D8 | HOSTS |
| C75D4C6E53A497C4DC1DF1F50BBEF08AC625A3D8 | x86_microsoft-windows-w..nfrastructure-other_31bf3856ad364e35_6.0.5384.4_none_3285630929235d47_hosts_d78df635 |
| D1531EAABD403C811DFBFB17985A97DBB0C3E534 | kbdclass.sys |
| DF9E8A2D18AEDD359476C1A45877F0614ECF4993 | fdc.sys |
| DFC37F6C15612F7AB155E53A028A69FB5987199A | Program Compatibility Wizard.lnk |
| E07EE000BC06B455534D8A517305C1208D30306B | audstub.sys |
| FB33FD00711440B9D0F3B3D526A753ED75640797 | navstart.wav |
| FB33FD00711440B9D0F3B3D526A753ED75640797 | Windows Navigation Start.wav |
| FB33FD00711440B9D0F3B3D526A753ED75640797 | Windows XP Start.wav |
| FB33FD00711440B9D0F3B3D526A753ED75640797 | xpstart.wa! |
| FB33FD00711440B9D0F3B3D526A753ED75640797 | xpstart.wav |
| FB33FD00711440B9D0F3B3D526A753ED75640797 | XPStart.wav |

# Annex C Anti-virus scanner logs for carved memory data files

In all, nine virus matches were identified between the various scanners. These matches are indicated below.

## C.1 Avast

```
./recup_dir.5/f0972904.exe      [infected by: Win32:Duqu-F [Rtk]]
./recup_dir.5/f0841616.exe      [infected by: Win32:Duqu-F [Rtk]]
./recup_dir.5/f0898328.dll      [infected by: Win32:Duqu-K [Rtk]]
./recup_dir.5/f0869280.dll      [infected by: Win32:Duqu-F [Rtk]]
./recup_dir.4/f0809656.pyc      [infected by: Win32:Duqu-F [Rtk]]
./recup_dir.4/f0843952.swf      [infected by: SWF:CVE-2007-0071 [Expl]] <- Match 2
./recup_dir.4/f0861008.exe      [infected by: Win32:Duqu-K [Rtk]]
./recup_dir.6/f0161192.exe      [infected by: Win32:Duqu-F [Rtk]]
./recup_dir.6/f0163032.dll      [infected by: Win32:Duqu-F [Rtk]]
./recup_dir.6/f0165472.dll      [infected by: Win32:Duqu-F [Rtk]]
./recup_dir.6/f0262544.dll      [infected by: Win32:Duqu-F [Rtk]]
./recup_dir.1/f0277688.dll      [infected by: Win32:Duqu-K [Rtk]]
./recup_dir.1/f0304160.dll/[Embedded_Ix#296e8]      [infected by: Win32:MalOb-GX [Cryp]]
./recup_dir.1/f0264240.dll      [infected by: Win32:Duqu-F [Rtk]] <- Match 8
./recup_dir.1/f0264288.dll      [infected by: Win32:Duqu-F [Rtk]] <- Match 7
./recup_dir.1/f0225968.exe      [infected by: Win32:Duqu-K [Rtk]]
./recup_dir.3/f0785768.exe      [infected by: Win32:StuxX-B [Wrm]] <- Match 1
```

## C.2 AVG

```
recup_dir.5/f0903856.dll      Virus found Win32/Heur
recup_dir.5/f0890376.exe      Virus found Win32/Heur
recup_dir.5/f0889112.dll      Virus found Win32/Heur
recup_dir.5/f0893696.exe      Virus found Win32/Heur
recup_dir.5/f0933680.exe      Trojan horse Rootkit-Pakes.AJ
recup_dir.4/f0816768.exe      Virus found Win32/Heur
recup_dir.4/f0806584.exe      Virus found Win32/Heur
recup_dir.4/f0843952.swf      Virus identified SWF/Exploit.F <- Match 2
recup_dir.4/f0842256.exe      Virus found Win32/Heur
recup_dir.4/f0832936.exe      Virus found Win32/Heur
recup_dir.4/f0865624.exe      Virus found Win32/Heur
recup_dir.4/f0805448.exe      Virus found Win32/Heur
recup_dir.4/f0805968.dll      Virus found Win32/Heur
recup_dir.4/f0825728.dll      Virus found Win32/Heur
recup_dir.6/f0161784.dll      Virus found Win32/Heur
```

| | |
|---|---|
| recup_dir.2/f0563568.exe | Virus found Win32/Heur |
| recup_dir.2/f0341176.exe | Trojan horse Rootkit-Pakes.AJ |
| recup_dir.2/f0608344.dll | Virus found Win32/Heur |
| recup_dir.2/f0572856.dll | Virus found Win32/Heur |
| recup_dir.2/f0595624.exe | Trojan horse Rootkit-Pakes.AE **<- Match 3** |
| recup_dir.2/f0459912.exe | Trojan horse Rootkit-Pakes.AJ |
| recup_dir.1/f0262712.dll | Virus found Win32/Heur |
| recup_dir.1/f0245496.dll | Virus found Win32/Heur |
| recup_dir.1/f0262960.dll | Virus found Win32/Heur **<- Match 9** |
| recup_dir.1/f0277128.exe | Virus found Win32/Heur |
| recup_dir.1/f0262824.dll | Virus found Win32/Heur |
| recup_dir.1/f0262944.dll | Virus found Win32/Heur |
| recup_dir.1/f0264240.dll | Virus found Win32/Heur **<- Match 8** |
| recup_dir.1/f0263040.dll | Virus found Win32/Heur |
| recup_dir.1/f0262632.dll | Virus found Win32/Heur |
| recup_dir.1/f0226264.dll | Virus found Win32/Heur |
| recup_dir.1/f0264288.dll | Virus found Win32/Heur **<- Match 7** |
| recup_dir.1/f0262728.dll | Virus found Win32/Heur |
| recup_dir.1/f0172584.dll | Virus found Win32/Heur |
| recup_dir.1/f0182168.dll | Virus found Win32/Heur |
| recup_dir.1/f0262792.dll | Virus found Win32/Heur |
| recup_dir.3/f0743744.dll | Virus found Win32/Heur |
| recup_dir.3/f0626480.dll | Virus found Win32/Heur |
| recup_dir.3/f0640880.exe | Virus found Win32/Heur |
| recup_dir.3/f0654984.dll | Virus found Win32/Heur |
| recup_dir.3/f0785768.exe | Trojan horse SHeur3.XLI **<- Match 1** |
| recup_dir.3/f0646224.exe | Virus found Win32/Heur |

## C.3 BitDefender

| | |
|---|---|
| recup_dir.3/f0785768.exe | infected: Gen:Variant.NSAnti.1 **<- Match 1** |
| recup_dir.3/f0770824.exe | infected: Trojan.Zlob.1.Gen |
| recup_dir.1/f0264256.exe | infected: Gen:Variant.FakeAlert.47 |
| recup_dir.1/f0262960.dll | infected: Gen:Heur.Conjar.5 **<- Match 9** |
| recup_dir.1/f0277432.dll | infected: Gen:Variant.Graftor.Elzob.17846 **<- Match 6** |
| recup_dir.2/f0573960.dll | infected: Gen:Variant.Graftor.Elzob.17846 **<- Match 5** |
| recup_dir.2/f0583552.dll | infected: Gen:Variant.Graftor.Elzob.17846 **<- Match 4** |
| recup_dir.2/f0595624.exe | infected: Gen:Variant.NSAnti.1 **<- Match 3** |

## C.4 Comodo

| | |
|---|---|
| recup_dir.4/f0857456.exe | Found Virus, Malware Name is TrojWare.Win32.FraudPack.P |
| recup_dir.6/f0582768.dll | Found Virus, Malware Name is TrojWare.Win32.FraudPack.P |
| recup_dir.2/f0436400.exe | Found Virus, Malware Name is TrojWare.Win32.FraudPack.P |

```
recup_dir.2/f0420688.exe      Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
recup_dir.2/f0573960.dll      Found  Virus,  Malware  Name  is  Packed.Win32.MUPX.Gen  <-
Match 5
recup_dir.2/f0583552.dll      Found  Virus,  Malware  Name  is  Packed.Win32.MUPX.Gen  <-
Match 4
recup_dir.1/f0093328.exe      Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
recup_dir.1/f0277432.dll      Found  Virus,  Malware  Name  is  Packed.Win32.MUPX.Gen  <-
Match 6
recup_dir.1/f0263784.dll      Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
recup_dir.3/f0719832.dll      Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
recup_dir.3/f0750168.dll      Found Virus, Malware Name is Heur.Packed.Unknown
recup_dir.3/f0613336.dll      Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
recup_dir.3/f0785768.exe      Found Virus, Malware Name is Worm.Win32.Stuxnet.a <- Match
1
recup_dir.3/f0730008.dll      Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
```

## C.5 F-Prot

F-Prot was the first of two anti-virus scanners unable to detect any malware whatsoever for the carved memory data files recovered.

## C.6 McAfee

McAfee was the second of two anti-virus scanners unable to detect any malware whatsoever for the carved memory data files recovered.

This page intentionally left blank.

# Annex D Textual output from the malfind plugin

The following output was generated by the malfind plugin having been run against the Stuxnet memory image, Stuxnet.vmem. The output is as follows:

```
Process: csrss.exe Pid: 600 Address: 0x7f6f0000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x7f6f0000  c8 00 00 00 1f 01 00 00 ff ee ff ee 08 70 00 00   .............p..
0x7f6f0010  08 00 00 00 00 fe 00 00 00 00 10 00 00 20 00 00   ................
0x7f6f0020  00 02 00 00 00 20 00 00 8d 01 00 00 ff ef fd 7f   ................
0x7f6f0030  03 00 08 06 00 00 00 00 00 00 00 00 00 00 00 00   ................

0x7f6f0000 c8000000         ENTER 0x0, 0x0
0x7f6f0004 1f               POP DS
0x7f6f0005 0100             ADD [EAX], EAX
0x7f6f0007 00ff             ADD BH, BH
0x7f6f0009 ee               OUT DX, AL
0x7f6f000a ff               DB 0xff
0x7f6f000b ee               OUT DX, AL
0x7f6f000c 087000           OR [EAX+0x0], DH
0x7f6f000f 0008             ADD [EAX], CL
0x7f6f0011 0000             ADD [EAX], AL
0x7f6f0013 0000             ADD [EAX], AL
0x7f6f0015 fe00             INC BYTE [EAX]
0x7f6f0017 0000             ADD [EAX], AL
0x7f6f0019 0010             ADD [EAX], DL
0x7f6f001b 0000             ADD [EAX], AL
0x7f6f001d 2000             AND [EAX], AL
0x7f6f001f 0000             ADD [EAX], AL
0x7f6f0021 0200             ADD AL, [EAX]
0x7f6f0023 0000             ADD [EAX], AL
0x7f6f0025 2000             AND [EAX], AL
0x7f6f0027 008d010000ff     ADD [EBP-0xffffff], CL
0x7f6f002d ef               OUT DX, EAX
0x7f6f002e fd               STD
0x7f6f002f 7f03             JG 0x7f6f0034
0x7f6f0031 0008             ADD [EAX], CL
0x7f6f0033 06               PUSH ES
0x7f6f0034 0000             ADD [EAX], AL
0x7f6f0036 0000             ADD [EAX], AL
0x7f6f0038 0000             ADD [EAX], AL
0x7f6f003a 0000             ADD [EAX], AL
0x7f6f003c 0000             ADD [EAX], AL
0x7f6f003e 0000             ADD [EAX], AL

Process: services.exe Pid: 668 Address: 0x940000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x00940000  90 06 94 00 c6 07 94 00 24 00 94 00 a5 04 00 00   ........$.......
0x00940010  f2 04 94 00 48 06 00 00 c9 04 94 00 29 00 00 00   ....H.......)...
0x00940020  00 00 c5 00 e8 13 00 00 00 5a 77 4d 61 70 56 69   .........ZwMapVi
0x00940030  65 77 4f 66 53 65 63 74 69 6f 6e 00 5a 51 81 c1   ewOfSection.ZQ..

0x940000 90             NOP
0x940001 06             PUSH ES
0x940002 94             XCHG ESP, EAX
0x940003 00c6           ADD DH, AL
0x940005 07             POP ES
0x940006 94             XCHG ESP, EAX
0x940007 002400         ADD [EAX+EAX], AH
0x94000a 94             XCHG ESP, EAX
0x94000b 00a5040000f2   ADD [EBP-0xdffffc], AH
0x940011 0494           ADD AL, 0x94
0x940013 004806         ADD [EAX+0x6], CL
```

```
0x940016 0000             ADD [EAX], AL
0x940018 c9               LEAVE
0x940019 0494             ADD AL, 0x94
0x94001b 0029             ADD [ECX], CH
0x94001d 0000             ADD [EAX], AL
0x94001f 0000             ADD [EAX], AL
0x940021 00c5             ADD CH, AL
0x940023 00e8             ADD AL, CH
0x940025 1300             ADC EAX, [EAX]
0x940027 0000             ADD [EAX], AL
0x940029 5a               POP EDX
0x94002a 774d             JA 0x940079
0x94002c 61               POPA
0x94002d 7056             JO 0x940085
0x94002f 6965774f665365   IMUL ESP, [EBP+0x77], 0x6553664f
0x940036 6374696f         ARPL [ECX+EBP*2+0x6f], SI
0x94003a 6e               OUTS DX, BYTE [ESI]
0x94003b 005a51           ADD [EDX+0x51], BL
0x94003e 81               DB 0x81
0x94003f c1               DB 0xc1

Process: services.exe Pid: 668 Address: 0x13f0000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x013f0000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x013f0010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x013f0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x013f0030  00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00   ................

0x13f0000 4d               DEC EBP
0x13f0001 5a               POP EDX
0x13f0002 90               NOP
0x13f0003 0003             ADD [EBX], AL
0x13f0005 0000             ADD [EAX], AL
0x13f0007 000400           ADD [EAX+EAX], AL
0x13f000a 0000             ADD [EAX], AL
0x13f000c ff               DB 0xff
0x13f000d ff00             INC DWORD [EAX]
0x13f000f 00b800000000     ADD [EAX+0x0], BH
0x13f0015 0000             ADD [EAX], AL
0x13f0017 004000           ADD [EAX+0x0], AL
0x13f001a 0000             ADD [EAX], AL
0x13f001c 0000             ADD [EAX], AL
0x13f001e 0000             ADD [EAX], AL
0x13f0020 0000             ADD [EAX], AL
0x13f0022 0000             ADD [EAX], AL
0x13f0024 0000             ADD [EAX], AL
0x13f0026 0000             ADD [EAX], AL
0x13f0028 0000             ADD [EAX], AL
0x13f002a 0000             ADD [EAX], AL
0x13f002c 0000             ADD [EAX], AL
0x13f002e 0000             ADD [EAX], AL
0x13f0030 0000             ADD [EAX], AL
0x13f0032 0000             ADD [EAX], AL
0x13f0034 0000             ADD [EAX], AL
0x13f0036 0000             ADD [EAX], AL
0x13f0038 0000             ADD [EAX], AL
0x13f003a 0000             ADD [EAX], AL
0x13f003c 0801             OR [ECX], AL
0x13f003e 0000             ADD [EAX], AL

Process: svchost.exe Pid: 940 Address: 0xb70000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x00b70000  29 87 7f ae 00 00 00 00 ff ff ff ff 77 35 00 01   )...........w5..
0x00b70010  4b 00 45 00 52 00 4e 00 45 00 4c 00 33 00 32 00   K.E.R.N.E.L.3.2.
0x00b70020  2e 00 44 00 4c 00 4c 00 2e 00 41 00 53 00 4c 00   ..D.L.L...A.S.L.
0x00b70030  52 00 2e 00 30 00 33 00 36 00 30 00 63 00 38 00   R...0.3.6.0.c.8.

0xb70000 29877fae0000     SUB [EDI+0xae7f], EAX
```

```
0xb70006 0000             ADD [EAX], AL
0xb70008 ff               DB 0xff
0xb70009 ff               DB 0xff
0xb7000a ff               DB 0xff
0xb7000b ff7735           PUSH DWORD [EDI+0x35]
0xb7000e 0001             ADD [ECX], AL
0xb70010 4b               DEC EBX
0xb70011 004500           ADD [EBP+0x0], AL
0xb70014 52               PUSH EDX
0xb70015 004e00           ADD [ESI+0x0], CL
0xb70018 45               INC EBP
0xb70019 004c0033         ADD [EAX+EAX+0x33], CL
0xb7001d 0032             ADD [EDX], DH
0xb7001f 002e             ADD [ESI], CH
0xb70021 0044004c         ADD [EAX+EAX+0x4c], AL
0xb70025 004c002e         ADD [EAX+EAX+0x2e], CL
0xb70029 004100           ADD [ECX+0x0], AL
0xb7002c 53               PUSH EBX
0xb7002d 004c0052         ADD [EAX+EAX+0x52], CL
0xb70031 002e             ADD [ESI], CH
0xb70033 0030             ADD [EAX], DH
0xb70035 0033             ADD [EBX], DH
0xb70037 0036             ADD [ESI], DH
0xb70039 0030             ADD [EAX], DH
0xb7003b 006300           ADD [EBX+0x0], AH
0xb7003e 3800             CMP [EAX], AL

Process: svchost.exe Pid: 940 Address: 0xbf0000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x00bf0000  90 06 bf 00 c6 07 bf 00 24 00 bf 00 a5 04 00 00   ........$.......
0x00bf0010  f2 04 bf 00 48 06 00 00 c9 04 bf 00 29 00 00 00   ....H.......)...
0x00bf0020  00 00 b7 00 e8 13 00 00 00 5a 77 4d 61 70 56 69   .........ZwMapVi
0x00bf0030  65 77 4f 66 53 65 63 74 69 6f 6e 00 5a 51 81 c1   ewOfSection.ZQ..

0xbf0000 90               NOP
0xbf0001 06               PUSH ES
0xbf0002 bf00c607bf       MOV EDI, 0xbf07c600
0xbf0007 002400           ADD [EAX+EAX], AH
0xbf000a bf00a50400       MOV EDI, 0x4a500
0xbf000f 00f2             ADD DL, DH
0xbf0011 04bf             ADD AL, 0xbf
0xbf0013 004806           ADD [EAX+0x6], CL
0xbf0016 0000             ADD [EAX], AL
0xbf0018 c9               LEAVE
0xbf0019 04bf             ADD AL, 0xbf
0xbf001b 0029             ADD [ECX], CH
0xbf001d 0000             ADD [EAX], AL
0xbf001f 0000             ADD [EAX], AL
0xbf0021 00b700e81300     ADD [EDI+0x13e800], DH
0xbf0027 0000             ADD [EAX], AL
0xbf0029 5a               POP EDX
0xbf002a 774d             JA 0xbf0079
0xbf002c 61               POPA
0xbf002d 7056             JO 0xbf0085
0xbf002f 6965774f665365   IMUL ESP, [EBP+0x77], 0x6553664f
0xbf0036 6374696f         ARPL [ECX+EBP*2+0x6f], SI
0xbf003a 6e               OUTS DX, BYTE [ESI]
0xbf003b 005a51           ADD [EDX+0x51], BL
0xbf003e 81               DB 0x81
0xbf003f c1               DB 0xc1

Process: svchost.exe Pid: 940 Address: 0xd00000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x00d00000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x00d00010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x00d00020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00d00030  00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00   ................
```

```
0xd00000 4d                 DEC EBP
0xd00001 5a                 POP EDX
0xd00002 90                 NOP
0xd00003 0003               ADD [EBX], AL
0xd00005 0000               ADD [EAX], AL
0xd00007 000400             ADD [EAX+EAX], AL
0xd0000a 0000               ADD [EAX], AL
0xd0000c ff                 DB 0xff
0xd0000d ff00               INC DWORD [EAX]
0xd0000f 00b800000000       ADD [EAX+0x0], BH
0xd00015 0000               ADD [EAX], AL
0xd00017 004000             ADD [EAX+0x0], AL
0xd0001a 0000               ADD [EAX], AL
0xd0001c 0000               ADD [EAX], AL
0xd0001e 0000               ADD [EAX], AL
0xd00020 0000               ADD [EAX], AL
0xd00022 0000               ADD [EAX], AL
0xd00024 0000               ADD [EAX], AL
0xd00026 0000               ADD [EAX], AL
0xd00028 0000               ADD [EAX], AL
0xd0002a 0000               ADD [EAX], AL
0xd0002c 0000               ADD [EAX], AL
0xd0002e 0000               ADD [EAX], AL
0xd00030 0000               ADD [EAX], AL
0xd00032 0000               ADD [EAX], AL
0xd00034 0000               ADD [EAX], AL
0xd00036 0000               ADD [EAX], AL
0xd00038 0000               ADD [EAX], AL
0xd0003a 0000               ADD [EAX], AL
0xd0003c 0801               OR [ECX], AL
0xd0003e 0000               ADD [EAX], AL

Process: explorer.exe Pid: 1196 Address: 0x2550000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x02550000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x02550010  00 00 55 02 00 00 00 00 00 00 00 00 00 00 00 00   ..U.............
0x02550020  10 00 55 02 00 00 00 00 00 00 00 00 00 00 00 00   ..U.............
0x02550030  20 00 55 02 00 00 00 00 00 00 00 00 00 00 00 00   ..U.............

0x2550000 0000              ADD [EAX], AL
0x2550002 0000              ADD [EAX], AL
0x2550004 0000              ADD [EAX], AL
0x2550006 0000              ADD [EAX], AL
0x2550008 0000              ADD [EAX], AL
0x255000a 0000              ADD [EAX], AL
0x255000c 0000              ADD [EAX], AL
0x255000e 0000              ADD [EAX], AL
0x2550010 0000              ADD [EAX], AL
0x2550012 55                PUSH EBP
0x2550013 0200              ADD AL, [EAX]
0x2550015 0000              ADD [EAX], AL
0x2550017 0000              ADD [EAX], AL
0x2550019 0000              ADD [EAX], AL
0x255001b 0000              ADD [EAX], AL
0x255001d 0000              ADD [EAX], AL
0x255001f 0010              ADD [EAX], DL
0x2550021 005502            ADD [EBP+0x2], DL
0x2550024 0000              ADD [EAX], AL
0x2550026 0000              ADD [EAX], AL
0x2550028 0000              ADD [EAX], AL
0x255002a 0000              ADD [EAX], AL
0x255002c 0000              ADD [EAX], AL
0x255002e 0000              ADD [EAX], AL
0x2550030 2000              AND [EAX], AL
0x2550032 55                PUSH EBP
0x2550033 0200              ADD AL, [EAX]
0x2550035 0000              ADD [EAX], AL
0x2550037 0000              ADD [EAX], AL
0x2550039 0000              ADD [EAX], AL
0x255003b 0000              ADD [EAX], AL
```

```
0x255003d 0000             ADD [EAX], AL
0x255003f 00               DB 0x0

Process: lsass.exe Pid: 868 Address: 0x80000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x00080000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x00080010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x00080020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00080030  00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00   ................

0x80000 4d               DEC EBP
0x80001 5a               POP EDX
0x80002 90               NOP
0x80003 0003             ADD [EBX], AL
0x80005 0000             ADD [EAX], AL
0x80007 000400           ADD [EAX+EAX], AL
0x8000a 0000             ADD [EAX], AL
0x8000c ff               DB 0xff
0x8000d ff00             INC DWORD [EAX]
0x8000f 00b800000000     ADD [EAX+0x0], BH
0x80015 0000             ADD [EAX], AL
0x80017 004000           ADD [EAX+0x0], AL
0x8001a 0000             ADD [EAX], AL
0x8001c 0000             ADD [EAX], AL
0x8001e 0000             ADD [EAX], AL
0x80020 0000             ADD [EAX], AL
0x80022 0000             ADD [EAX], AL
0x80024 0000             ADD [EAX], AL
0x80026 0000             ADD [EAX], AL
0x80028 0000             ADD [EAX], AL
0x8002a 0000             ADD [EAX], AL
0x8002c 0000             ADD [EAX], AL
0x8002e 0000             ADD [EAX], AL
0x80030 0000             ADD [EAX], AL
0x80032 0000             ADD [EAX], AL
0x80034 0000             ADD [EAX], AL
0x80036 0000             ADD [EAX], AL
0x80038 0000             ADD [EAX], AL
0x8003a 0000             ADD [EAX], AL
0x8003c 0801             OR [ECX], AL
0x8003e 0000             ADD [EAX], AL

Process: lsass.exe Pid: 868 Address: 0x1000000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 2, Protection: 6

0x01000000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x01000010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x01000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x01000030  00 00 00 00 00 00 00 00 00 00 00 00 d0 00 00 00   ................

0x1000000 4d               DEC EBP
0x1000001 5a               POP EDX
0x1000002 90               NOP
0x1000003 0003             ADD [EBX], AL
0x1000005 0000             ADD [EAX], AL
0x1000007 000400           ADD [EAX+EAX], AL
0x100000a 0000             ADD [EAX], AL
0x100000c ff               DB 0xff
0x100000d ff00             INC DWORD [EAX]
0x100000f 00b800000000     ADD [EAX+0x0], BH
0x1000015 0000             ADD [EAX], AL
0x1000017 004000           ADD [EAX+0x0], AL
0x100001a 0000             ADD [EAX], AL
0x100001c 0000             ADD [EAX], AL
0x100001e 0000             ADD [EAX], AL
0x1000020 0000             ADD [EAX], AL
0x1000022 0000             ADD [EAX], AL
0x1000024 0000             ADD [EAX], AL
0x1000026 0000             ADD [EAX], AL
```

```
0x1000028 0000             ADD [EAX], AL
0x100002a 0000             ADD [EAX], AL
0x100002c 0000             ADD [EAX], AL
0x100002e 0000             ADD [EAX], AL
0x1000030 0000             ADD [EAX], AL
0x1000032 0000             ADD [EAX], AL
0x1000034 0000             ADD [EAX], AL
0x1000036 0000             ADD [EAX], AL
0x1000038 0000             ADD [EAX], AL
0x100003a 0000             ADD [EAX], AL
0x100003c d000             ROL BYTE [EAX], 0x1
0x100003e 0000             ADD [EAX], AL

Process: lsass.exe Pid: 1928 Address: 0x80000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x00080000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x00080010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x00080020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00080030  00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00   ................

0x80000 4d               DEC EBP
0x80001 5a               POP EDX
0x80002 90               NOP
0x80003 0003             ADD [EBX], AL
0x80005 0000             ADD [EAX], AL
0x80007 000400           ADD [EAX+EAX], AL
0x8000a 0000             ADD [EAX], AL
0x8000c ff               DB 0xff
0x8000d ff00             INC DWORD [EAX]
0x8000f 00b800000000     ADD [EAX+0x0], BH
0x80015 0000             ADD [EAX], AL
0x80017 004000           ADD [EAX+0x0], AL
0x8001a 0000             ADD [EAX], AL
0x8001c 0000             ADD [EAX], AL
0x8001e 0000             ADD [EAX], AL
0x80020 0000             ADD [EAX], AL
0x80022 0000             ADD [EAX], AL
0x80024 0000             ADD [EAX], AL
0x80026 0000             ADD [EAX], AL
0x80028 0000             ADD [EAX], AL
0x8002a 0000             ADD [EAX], AL
0x8002c 0000             ADD [EAX], AL
0x8002e 0000             ADD [EAX], AL
0x80030 0000             ADD [EAX], AL
0x80032 0000             ADD [EAX], AL
0x80034 0000             ADD [EAX], AL
0x80036 0000             ADD [EAX], AL
0x80038 0000             ADD [EAX], AL
0x8003a 0000             ADD [EAX], AL
0x8003c 0801             OR [ECX], AL
0x8003e 0000             ADD [EAX], AL

Process: lsass.exe Pid: 1928 Address: 0x1000000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 2, Protection: 6

0x01000000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x01000010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x01000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x01000030  00 00 00 00 00 00 00 00 00 00 00 00 d0 00 00 00   ................

0x1000000 4d               DEC EBP
0x1000001 5a               POP EDX
0x1000002 90               NOP
0x1000003 0003             ADD [EBX], AL
0x1000005 0000             ADD [EAX], AL
0x1000007 000400           ADD [EAX+EAX], AL
0x100000a 0000             ADD [EAX], AL
0x100000c ff               DB 0xff
0x100000d ff00             INC DWORD [EAX]
```

```
0x100000f 00b800000000     ADD [EAX+0x0], BH
0x1000015 0000             ADD [EAX], AL
0x1000017 004000           ADD [EAX+0x0], AL
0x100001a 0000             ADD [EAX], AL
0x100001c 0000             ADD [EAX], AL
0x100001e 0000             ADD [EAX], AL
0x1000020 0000             ADD [EAX], AL
0x1000022 0000             ADD [EAX], AL
0x1000024 0000             ADD [EAX], AL
0x1000026 0000             ADD [EAX], AL
0x1000028 0000             ADD [EAX], AL
0x100002a 0000             ADD [EAX], AL
0x100002c 0000             ADD [EAX], AL
0x100002e 0000             ADD [EAX], AL
0x1000030 0000             ADD [EAX], AL
0x1000032 0000             ADD [EAX], AL
0x1000034 0000             ADD [EAX], AL
0x1000036 0000             ADD [EAX], AL
0x1000038 0000             ADD [EAX], AL
0x100003a 0000             ADD [EAX], AL
0x100003c d000             ROL BYTE [EAX], 0x1
0x100003e 0000             ADD [EAX], AL

Process: lsass.exe Pid: 1928 Address: 0x6f0000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x006f0000  29 87 7f ae 00 00 00 00 ff ff ff ff 77 35 00 01   )...........w5..
0x006f0010  4b 00 45 00 52 00 4e 00 45 00 4c 00 33 00 32 00   K.E.R.N.E.L.3.2.
0x006f0020  2e 00 44 00 4c 00 4c 00 2e 00 41 00 53 00 4c 00   ..D.L.L...A.S.L.
0x006f0030  52 00 2e 00 30 00 33 00 36 00 30 00 62 00 37 00   R...0.3.6.0.b.7.

0x6f0000 29877fae0000     SUB [EDI+0xae7f], EAX
0x6f0006 0000             ADD [EAX], AL
0x6f0008 ff               DB 0xff
0x6f0009 ff               DB 0xff
0x6f000a ff               DB 0xff
0x6f000b ff7735           PUSH DWORD [EDI+0x35]
0x6f000e 0001             ADD [ECX], AL
0x6f0010 4b               DEC EBX
0x6f0011 004500           ADD [EBP+0x0], AL
0x6f0014 52               PUSH EDX
0x6f0015 004e00           ADD [ESI+0x0], CL
0x6f0018 45               INC EBP
0x6f0019 004c0033         ADD [EAX+EAX+0x33], CL
0x6f001d 0032             ADD [EDX], DH
0x6f001f 002e             ADD [ESI], CH
0x6f0021 0044004c         ADD [EAX+EAX+0x4c], AL
0x6f0025 004c002e         ADD [EAX+EAX+0x2e], CL
0x6f0029 004100           ADD [ECX+0x0], AL
0x6f002c 53               PUSH EBX
0x6f002d 004c0052         ADD [EAX+EAX+0x52], CL
0x6f0031 002e             ADD [ESI], CH
0x6f0033 0030             ADD [EAX], DH
0x6f0035 0033             ADD [EBX], DH
0x6f0037 0036             ADD [ESI], DH
0x6f0039 0030             ADD [EAX], DH
0x6f003b 006200           ADD [EDX+0x0], AH
0x6f003e 37               AAA
0x6f003f 00               DB 0x0

Process: lsass.exe Pid: 1928 Address: 0x680000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x00680000  90 06 68 00 c6 07 68 00 24 00 68 00 a5 04 00 00   ..h...h.$.h.....
0x00680010  f2 04 68 00 48 06 00 00 c9 04 68 00 29 00 00 00   ..h.H.....h.)...
0x00680020  00 00 6f 00 e8 13 00 00 00 5a 77 4d 61 70 56 69   ..o......ZwMapVi
0x00680030  65 77 4f 66 53 65 63 74 69 6f 6e 00 5a 51 81 c1   ewOfSection.ZQ..

0x680000 90               NOP
0x680001 06               PUSH ES
```

```
0x680002 6800c60768       PUSH DWORD 0x6807c600
0x680007 002400           ADD [EAX+EAX], AH
0x68000a 6800a50400       PUSH DWORD 0x4a500
0x68000f 00f2             ADD DL, DH
0x680011 0468             ADD AL, 0x68
0x680013 004806           ADD [EAX+0x6], CL
0x680016 0000             ADD [EAX], AL
0x680018 c9               LEAVE
0x680019 0468             ADD AL, 0x68
0x68001b 0029             ADD [ECX], CH
0x68001d 0000             ADD [EAX], AL
0x68001f 0000             ADD [EAX], AL
0x680021 006f00           ADD [EDI+0x0], CH
0x680024 e813000000       CALL 0x68003c
0x680029 5a               POP EDX
0x68002a 774d             JA 0x680079
0x68002c 61               POPA
0x68002d 7056             JO 0x680085
0x68002f 6965774f665365   IMUL ESP, [EBP+0x77], 0x6553664f
0x680036 6374696f         ARPL [ECX+EBP*2+0x6f], SI
0x68003a 6e               OUTS DX, BYTE [ESI]
0x68003b 005a51           ADD [EDX+0x51], BL
0x68003e 81               DB 0x81
0x68003f c1               DB 0xc1

Process: lsass.exe Pid: 1928 Address: 0x870000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x00870000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x00870010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x00870020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00870030  00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00   ................

0x870000 4d               DEC EBP
0x870001 5a               POP EDX
0x870002 90               NOP
0x870003 0003             ADD [EBX], AL
0x870005 0000             ADD [EAX], AL
0x870007 000400           ADD [EAX+EAX], AL
0x87000a 0000             ADD [EAX], AL
0x87000c ff               DB 0xff
0x87000d ff00             INC DWORD [EAX]
0x87000f 00b800000000     ADD [EAX+0x0], BH
0x870015 0000             ADD [EAX], AL
0x870017 004000           ADD [EAX+0x0], AL
0x87001a 0000             ADD [EAX], AL
0x87001c 0000             ADD [EAX], AL
0x87001e 0000             ADD [EAX], AL
0x870020 0000             ADD [EAX], AL
0x870022 0000             ADD [EAX], AL
0x870024 0000             ADD [EAX], AL
0x870026 0000             ADD [EAX], AL
0x870028 0000             ADD [EAX], AL
0x87002a 0000             ADD [EAX], AL
0x87002c 0000             ADD [EAX], AL
0x87002e 0000             ADD [EAX], AL
0x870030 0000             ADD [EAX], AL
0x870032 0000             ADD [EAX], AL
0x870034 0000             ADD [EAX], AL
0x870036 0000             ADD [EAX], AL
0x870038 0000             ADD [EAX], AL
0x87003a 0000             ADD [EAX], AL
0x87003c 0801             OR [ECX], AL
0x87003e 0000             ADD [EAX], AL
```

# Annex E Output of Dlldump plugin for PIDs 668, 868, 940 and 1928

The various DLLs dumped for PIDs 668, 868, 940 and 1928 using the *dlldump* plugin have been broken down by PID.

## E.1 DLLs dumped for services.exe (PID 668)

The following DLLs were dumped for *services.exe*:

```
0x82073020 services.exe         0x001000000 services.exe
OK: module.668.2273020.1000000.dll

0x82073020 services.exe         0x07c900000 ntdll.dll
OK: module.668.2273020.7c900000.dll

0x82073020 services.exe         0x077f60000 SHLWAPI.dll
OK: module.668.2273020.77f60000.dll

0x82073020 services.exe         0x077b70000 eventlog.dll
OK: module.668.2273020.77b70000.dll

0x82073020 services.exe         0x076360000 WINSTA.dll
OK: module.668.2273020.76360000.dll

0x82073020 services.exe         0x05ad70000 uxtheme.dll
OK: module.668.2273020.5ad70000.dll

0x82073020 services.exe         0x068000000 rsaenh.dll
OK: module.668.2273020.68000000.dll

0x82073020 services.exe         0x07dba0000 umpnpmgr.dll
OK: module.668.2273020.7dba0000.dll

0x82073020 services.exe         0x0771b0000 WININET.dll
OK: module.668.2273020.771b0000.dll

0x82073020 services.exe         0x077dd0000 ADVAPI32.dll
OK: module.668.2273020.77dd0000.dll

0x82073020 services.exe         0x077fe0000 Secur32.dll
OK: module.668.2273020.77fe0000.dll

0x82073020 services.exe         0x077c00000 VERSION.dll
OK: module.668.2273020.77c00000.dll

0x82073020 services.exe         0x076f20000 DNSAPI.dll
OK: module.668.2273020.76f20000.dll

0x82073020 services.exe         0x077b40000 Apphelp.dll
OK: module.668.2273020.77b40000.dll

0x82073020 services.exe         0x001020000 xpsp2res.dll
OK: module.668.2273020.1020000.dll

0x82073020 services.exe         0x0773d0000 comctl32.dll
OK: module.668.2273020.773d0000.dll

0x82073020 services.exe         0x05b860000 NETAPI32.dll
OK: module.668.2273020.5b860000.dll
```

```
0x82073020 services.exe         0x077e70000 RPCRT4.dll
OK: module.668.2273020.77e70000.dll

0x82073020 services.exe         0x076080000 MSVCP60.dll
OK: module.668.2273020.76080000.dll

0x82073020 services.exe         0x0013f0000 KERNEL32....0360c5e2
OK: module.668.2273020.13f0000.dll

0x82073020 services.exe         0x071ab0000 WS2_32.dll
OK: module.668.2273020.71ab0000.dll

0x82073020 services.exe         0x071ad0000 WSOCK32.dll
OK: module.668.2273020.71ad0000.dll

0x82073020 services.exe         0x0774e0000 ole32.dll
OK: module.668.2273020.774e0000.dll

0x82073020 services.exe         0x077120000 OLEAUT32.dll
OK: module.668.2273020.77120000.dll

0x82073020 services.exe         0x076f50000 wtsapi32.dll
OK: module.668.2273020.76f50000.dll

0x82073020 services.exe         0x076d60000 IPHLPAPI.DLL
OK: module.668.2273020.76d60000.dll

0x82073020 services.exe         0x05cb70000 ShimEng.dll
OK: module.668.2273020.5cb70000.dll

0x82073020 services.exe         0x077c10000 msvcrt.dll
OK: module.668.2273020.77c10000.dll

0x82073020 services.exe         0x0769c0000 USERENV.dll
OK: module.668.2273020.769c0000.dll

0x82073020 services.exe         0x07c800000 kernel32.dll
OK: module.668.2273020.7c800000.dll

0x82073020 services.exe         0x07dbd0000 SCESRV.dll
OK: module.668.2273020.7dbd0000.dll

0x82073020 services.exe         0x076bf0000 PSAPI.DLL
OK: module.668.2273020.76bf0000.dll

0x82073020 services.exe         0x07e410000 USER32.dll
OK: module.668.2273020.7e410000.dll

0x82073020 services.exe         0x077f10000 GDI32.dll
OK: module.668.2273020.77f10000.dll

0x82073020 services.exe         0x076c30000 WINTRUST.dll
OK: module.668.2273020.76c30000.dll

0x82073020 services.exe         0x07c9c0000 SHELL32.dll
OK: module.668.2273020.7c9c0000.dll

0x82073020 services.exe         0x047260000 AcAdProc.dll
OK: module.668.2273020.47260000.dll

0x82073020 services.exe         0x05f770000 NCObjAPI.DLL
OK: module.668.2273020.5f770000.dll

0x82073020 services.exe         0x075150000 Cabinet.dll
OK: module.668.2273020.75150000.dll

0x82073020 services.exe         0x077a80000 CRYPT32.dll
OK: module.668.2273020.77a80000.dll
```

```
0x82073020 services.exe        0x076c90000 IMAGEHLP.dll
OK: module.668.2273020.76c90000.dll

0x82073020 services.exe        0x071aa0000 WS2HELP.dll
OK: module.668.2273020.71aa0000.dll

0x82073020 services.exe        0x0776c0000 AUTHZ.dll
OK: module.668.2273020.776c0000.dll

0x82073020 services.exe        0x05d090000 comctl32.dll
OK: module.668.2273020.5d090000.dll

0x82073020 services.exe        0x077b20000 MSASN1.dll
OK: module.668.2273020.77b20000.dll
```

## E.2 DLLs dumped for lsass.exe (PID 868)

The following DLLs were dumped for *lsass.exe*:

```
0x81c498c8 lsass.exe           0x001000000 lsass.exe
OK: module.868.1e498c8.1000000.dll

0x81c498c8 lsass.exe           0x07c900000 ntdll.dll
OK: module.868.1e498c8.7c900000.dll

0x81c498c8 lsass.exe           0x077e70000 RPCRT4.dll
OK: module.868.1e498c8.77e70000.dll

0x81c498c8 lsass.exe           0x077f10000 GDI32.dll
OK: module.868.1e498c8.77f10000.dll

0x81c498c8 lsass.exe           0x077dd0000 ADVAPI32.dll
OK: module.868.1e498c8.77dd0000.dll

0x81c498c8 lsass.exe           0x07c800000 kernel32.dll
OK: module.868.1e498c8.7c800000.dll

0x81c498c8 lsass.exe           0x07e410000 USER32.dll
OK: module.868.1e498c8.7e410000.dll

0x81c498c8 lsass.exe           0x077fe0000 Secur32.dll
OK: module.868.1e498c8.77fe0000.dll
```

## E.3 DLLs dumped for svchost.exe (PID 940)

The following DLLs were dumped for *svchost.exe*:

```
0x81e61da0 svchost.exe         0x001000000 svchost.exe
OK: module.940.2061da0.1000000.dll

0x81e61da0 svchost.exe         0x07c900000 ntdll.dll
OK: module.940.2061da0.7c900000.dll

0x81e61da0 svchost.exe         0x077be0000 MSACM32.dll
OK: module.940.2061da0.77be0000.dll

0x81e61da0 svchost.exe         0x077f60000 SHLWAPI.dll
OK: module.940.2061da0.77f60000.dll

0x81e61da0 svchost.exe         0x05ad70000 UxTheme.dll
OK: module.940.2061da0.5ad70000.dll
```

```
0x81e61da0 svchost.exe          0x068000000 rsaenh.dll
OK: module.940.2061da0.68000000.dll
0x81e61da0 svchost.exe          0x0769c0000 USERENV.dll
OK: module.940.2061da0.769c0000.dll
0x81e61da0 svchost.exe          0x0771b0000 WININET.dll
OK: module.940.2061da0.771b0000.dll
0x81e61da0 svchost.exe          0x076fc0000 rasadhlp.dll
OK: module.940.2061da0.76fc0000.dll
0x81e61da0 svchost.exe          0x077dd0000 ADVAPI32.dll
OK: module.940.2061da0.77dd0000.dll
0x81e61da0 svchost.exe          0x077a80000 CRYPT32.dll
OK: module.940.2061da0.77a80000.dll
0x81e61da0 svchost.exe          0x077fe0000 Secur32.dll
OK: module.940.2061da0.77fe0000.dll
0x81e61da0 svchost.exe          0x077c00000 VERSION.dll
OK: module.940.2061da0.77c00000.dll
0x81e61da0 svchost.exe          0x076f20000 DNSAPI.dll
OK: module.940.2061da0.76f20000.dll
0x81e61da0 svchost.exe          0x076b40000 WINMM.dll
OK: module.940.2061da0.76b40000.dll
0x81e61da0 svchost.exe          0x071a50000 mswsock.dll
OK: module.940.2061da0.71a50000.dll
0x81e61da0 svchost.exe          0x05b860000 NETAPI32.dll
OK: module.940.2061da0.5b860000.dll
0x81e61da0 svchost.exe          0x000670000 xpsp2res.dll
OK: module.940.2061da0.670000.dll
0x81e61da0 svchost.exe          0x06f880000 AcGenral.DLL
OK: module.940.2061da0.6f880000.dll
0x81e61da0 svchost.exe          0x071a90000 wshtcpip.dll
OK: module.940.2061da0.71a90000.dll
0x81e61da0 svchost.exe          0x071ab0000 WS2_32.dll
OK: module.940.2061da0.71ab0000.dll
0x81e61da0 svchost.exe          0x076f60000 WLDAP32.dll
OK: module.940.2061da0.76f60000.dll
0x81e61da0 svchost.exe          0x071ad0000 WSOCK32.dll
OK: module.940.2061da0.71ad0000.dll
0x81e61da0 svchost.exe          0x0774e0000 ole32.dll
OK: module.940.2061da0.774e0000.dll
0x81e61da0 svchost.exe          0x07e410000 USER32.dll
OK: module.940.2061da0.7e410000.dll
0x81e61da0 svchost.exe          0x000d00000 KERNEL32....0360c8ee
OK: module.940.2061da0.d00000.dll
0x81e61da0 svchost.exe          0x077f10000 GDI32.dll
OK: module.940.2061da0.77f10000.dll
0x81e61da0 svchost.exe          0x077120000 OLEAUT32.dll
OK: module.940.2061da0.77120000.dll
```

```
0x81e61da0 svchost.exe          0x076fd0000 CLBCATQ.DLL
OK: module.940.2061da0.76fd0000.dll
0x81e61da0 svchost.exe          0x076d60000 iphlpapi.dll
OK: module.940.2061da0.76d60000.dll
0x81e61da0 svchost.exe          0x05cb70000 ShimEng.dll
OK: module.940.2061da0.5cb70000.dll
0x81e61da0 svchost.exe          0x076fb0000 winrnr.dll
OK: module.940.2061da0.76fb0000.dll
0x81e61da0 svchost.exe          0x07c9c0000 SHELL32.dll
OK: module.940.2061da0.7c9c0000.dll
0x81e61da0 svchost.exe          0x07c800000 kernel32.dll
OK: module.940.2061da0.7c800000.dll
0x81e61da0 svchost.exe          0x0773d0000 comctl32.dll
OK: module.940.2061da0.773d0000.dll
0x81e61da0 svchost.exe          0x076bf0000 PSAPI.DLL
OK: module.940.2061da0.76bf0000.dll
0x81e61da0 svchost.exe          0x0662b0000 hnetcfg.dll
OK: module.940.2061da0.662b0000.dll
0x81e61da0 svchost.exe          0x077c10000 msvcrt.dll
OK: module.940.2061da0.77c10000.dll
0x81e61da0 svchost.exe          0x077e70000 RPCRT4.dll
OK: module.940.2061da0.77e70000.dll
0x81e61da0 svchost.exe          0x077050000 COMRes.dll
OK: module.940.2061da0.77050000.dll
0x81e61da0 svchost.exe          0x076a80000 rpcss.dll
OK: module.940.2061da0.76a80000.dll
0x81e61da0 svchost.exe          0x05d090000 comctl32.dll
OK: module.940.2061da0.5d090000.dll
0x81e61da0 svchost.exe          0x071aa0000 WS2HELP.dll
OK: module.940.2061da0.71aa0000.dll
0x81e61da0 svchost.exe          0x077b20000 MSASN1.dll
OK: module.940.2061da0.77b20000.dll
```

## E.4 DLLs dumped for lsass.exe (PID 1928)

The following DLLs were dumped for *lsass.exe*:

```
0x81c47c00 lsass.exe            0x001000000 lsass.exe
OK: module.1928.1e47c00.1000000.dll
0x81c47c00 lsass.exe            0x07c900000 ntdll.dll
OK: module.1928.1e47c00.7c900000.dll
0x81c47c00 lsass.exe            0x077f60000 SHLWAPI.dll
OK: module.1928.1e47c00.77f60000.dll
0x81c47c00 lsass.exe            0x0771b0000 WININET.dll
OK: module.1928.1e47c00.771b0000.dll
0x81c47c00 lsass.exe            0x077dd0000 ADVAPI32.dll
OK: module.1928.1e47c00.77dd0000.dll
```

```
0x81c47c00 lsass.exe            0x077a80000 CRYPT32.dll
OK: module.1928.1e47c00.77a80000.dll

0x81c47c00 lsass.exe            0x077fe0000 Secur32.dll
OK: module.1928.1e47c00.77fe0000.dll

0x81c47c00 lsass.exe            0x077c00000 VERSION.dll
OK: module.1928.1e47c00.77c00000.dll

0x81c47c00 lsass.exe            0x076d60000 IPHLPAPI.DLL
OK: module.1928.1e47c00.76d60000.dll

0x81c47c00 lsass.exe            0x05b860000 NETAPI32.dll
OK: module.1928.1e47c00.5b860000.dll

0x81c47c00 lsass.exe            0x071ab0000 WS2_32.dll
OK: module.1928.1e47c00.71ab0000.dll

0x81c47c00 lsass.exe            0x071ad0000 WSOCK32.dll
OK: module.1928.1e47c00.71ad0000.dll

0x81c47c00 lsass.exe            0x0774e0000 ole32.dll
OK: module.1928.1e47c00.774e0000.dll

0x81c47c00 lsass.exe            0x07e410000 USER32.dll
OK: module.1928.1e47c00.7e410000.dll

0x81c47c00 lsass.exe            0x077f10000 GDI32.dll
OK: module.1928.1e47c00.77f10000.dll

0x81c47c00 lsass.exe            0x077120000 OLEAUT32.dll
OK: module.1928.1e47c00.77120000.dll

0x81c47c00 lsass.exe            0x0769c0000 USERENV.dll
OK: module.1928.1e47c00.769c0000.dll

0x81c47c00 lsass.exe            0x07c800000 kernel32.dll
OK: module.1928.1e47c00.7c800000.dll

0x81c47c00 lsass.exe            0x0773d0000 comctl32.dll
OK: module.1928.1e47c00.773d0000.dll

0x81c47c00 lsass.exe            0x076bf0000 PSAPI.DLL
OK: module.1928.1e47c00.76bf0000.dll

0x81c47c00 lsass.exe            0x077c10000 msvcrt.dll
OK: module.1928.1e47c00.77c10000.dll

0x81c47c00 lsass.exe            0x077e70000 RPCRT4.dll
OK: module.1928.1e47c00.77e70000.dll

0x81c47c00 lsass.exe            0x000870000 KERNEL32....0360b7ab
OK: module.1928.1e47c00.870000.dll

0x81c47c00 lsass.exe            0x076f20000 DNSAPI.dll
OK: module.1928.1e47c00.76f20000.dll

0x81c47c00 lsass.exe            0x07c9c0000 SHELL32.dll
OK: module.1928.1e47c00.7c9c0000.dll

0x81c47c00 lsass.exe            0x071aa0000 WS2HELP.dll
OK: module.1928.1e47c00.71aa0000.dll

0x81c47c00 lsass.exe            0x05d090000 comctl32.dll
OK: module.1928.1e47c00.5d090000.dll

0x81c47c00 lsass.exe            0x077b20000 MSASN1.dll
OK: module.1928.1e47c00.77b20000.dll
```

# Annex F    Fuzzy hash matches for Dlldump-based DLLs

## F.1    Fuzzy hash matches for DLL memory samples

This sub annex lists all fuzzy hash matches for the *dlldump*-based memory samples for PIDs 668, 868, 940 and 1928. The matches are as follows:

*Table 32: Fuzzy hash matches between Dlldump-based memory samples (sorted by %).*

| Matched Filename #1 | Matched Filename #2 | Match (in %) |
|---|---|---|
| module.668.2273020.773d0000.dll | module.1928.1e47c00.773d0000.dll | 100 |
| module.868.1e498c8.1000000.dll | module.1928.1e47c00.1000000.dll | 100 |
| module.868.1e498c8.77fe0000.dll | module.1928.1e47c00.77fe0000.dll | 100 |
| module.940.2061da0.5b860000.dll | module.1928.1e47c00.5b860000.dll | 100 |
| module.940.2061da0.773d0000.dll | module.1928.1e47c00.773d0000.dll | 100 |
| module.940.2061da0.773d0000.dll | module.668.2273020.773d0000.dll | 100 |
| module.668.2273020.5d090000.dll | module.1928.1e47c00.5d090000.dll | 99 |
| module.668.2273020.71ab0000.dll | module.1928.1e47c00.71ab0000.dll | 99 |
| module.668.2273020.771b0000.dll | module.1928.1e47c00.771b0000.dll | 99 |
| module.940.2061da0.5cb70000.dll | module.668.2273020.5cb70000.dll | 99 |
| module.940.2061da0.5d090000.dll | module.1928.1e47c00.5d090000.dll | 99 |
| module.940.2061da0.5d090000.dll | module.668.2273020.5d090000.dll | 99 |
| module.940.2061da0.71aa0000.dll | module.668.2273020.71aa0000.dll | 99 |
| module.940.2061da0.71ad0000.dll | module.668.2273020.71ad0000.dll | 99 |
| module.940.2061da0.771b0000.dll | module.1928.1e47c00.771b0000.dll | 99 |
| module.940.2061da0.771b0000.dll | module.668.2273020.771b0000.dll | 99 |
| module.940.2061da0.774e0000.dll | module.1928.1e47c00.774e0000.dll | 99 |
| module.940.2061da0.77b20000.dll | module.1928.1e47c00.77b20000.dll | 99 |
| module.940.2061da0.77c00000.dll | module.668.2273020.77c00000.dll | 99 |
| module.940.2061da0.77f10000.dll | module.668.2273020.77f10000.dll | 99 |
| module.940.2061da0.77fe0000.dll | module.668.2273020.77fe0000.dll | 99 |
| module.668.2273020.13f0000.dll | module.1928.1e47c00.870000.dll | 97 |

| Matched Filename #1 | Matched Filename #2 | Match (in %) |
|---|---|---|
| module.668.2273020.77120000.dll | module.1928.1e47c00.77120000.dll | 97 |
| module.668.2273020.77f10000.dll | module.1928.1e47c00.77f10000.dll | 97 |
| module.940.2061da0.77a80000.dll | module.1928.1e47c00.77a80000.dll | 97 |
| module.940.2061da0.77f10000.dll | module.1928.1e47c00.77f10000.dll | 97 |
| module.940.2061da0.d00000.dll | module.1928.1e47c00.870000.dll | 97 |
| module.940.2061da0.d00000.dll | module.668.2273020.13f0000.dll | 97 |
| module.668.2273020.76f20000.dll | module.1928.1e47c00.76f20000.dll | 96 |
| module.940.2061da0.7c9c0000.dll | module.1928.1e47c00.7c9c0000.dll | 96 |
| module.668.2273020.71ad0000.dll | module.1928.1e47c00.71ad0000.dll | 94 |
| module.668.2273020.76d60000.dll | module.1928.1e47c00.76d60000.dll | 94 |
| module.668.2273020.77f60000.dll | module.1928.1e47c00.77f60000.dll | 94 |
| module.868.1e498c8.77dd0000.dll | module.1928.1e47c00.77dd0000.dll | 94 |
| module.940.2061da0.71ad0000.dll | module.1928.1e47c00.71ad0000.dll | 94 |
| module.868.1e498c8.7e410000.dll | module.1928.1e47c00.7e410000.dll | 93 |
| module.940.2061da0.77f60000.dll | module.1928.1e47c00.77f60000.dll | 93 |
| module.940.2061da0.77f60000.dll | module.668.2273020.77f60000.dll | 93 |
| module.668.2273020.774e0000.dll | module.1928.1e47c00.774e0000.dll | 91 |
| module.940.2061da0.68000000.dll | module.668.2273020.68000000.dll | 91 |
| module.940.2061da0.774e0000.dll | module.668.2273020.774e0000.dll | 91 |
| module.940.2061da0.7c800000.dll | module.1928.1e47c00.7c800000.dll | 91 |
| module.868.1e498c8.77f10000.dll | module.1928.1e47c00.77f10000.dll | 90 |
| module.868.1e498c8.7c800000.dll | module.1928.1e47c00.7c800000.dll | 90 |
| module.940.2061da0.77c00000.dll | module.1928.1e47c00.77c00000.dll | 90 |
| module.668.2273020.77c00000.dll | module.1928.1e47c00.77c00000.dll | 88 |
| module.868.1e498c8.7c900000.dll | module.1928.1e47c00.7c900000.dll | 88 |
| module.940.2061da0.769c0000.dll | module.1928.1e47c00.769c0000.dll | 88 |
| module.940.2061da0.77120000.dll | module.1928.1e47c00.77120000.dll | 88 |
| module.940.2061da0.77120000.dll | module.668.2273020.77120000.dll | 88 |
| module.940.2061da0.77dd0000.dll | module.668.2273020.77dd0000.dll | 88 |

| Matched Filename #1 | Matched Filename #2 | Match (in %) |
|---|---|---|
| module.868.1e498c8.77f10000.dll | module.668.2273020.77f10000.dll | 85 |
| module.940.2061da0.77f10000.dll | module.868.1e498c8.77f10000.dll | 85 |
| module.940.2061da0.77c10000.dll | module.668.2273020.77c10000.dll | 83 |
| module.940.2061da0.7c900000.dll | module.668.2273020.7c900000.dll | 83 |
| module.940.2061da0.76bf0000.dll | module.1928.1e47c00.76bf0000.dll | 82 |
| module.940.2061da0.7c800000.dll | module.868.1e498c8.7c800000.dll | 82 |
| module.940.2061da0.7e410000.dll | module.1928.1e47c00.7e410000.dll | 82 |
| module.940.2061da0.76bf0000.dll | module.668.2273020.76bf0000.dll | 80 |
| module.668.2273020.7c9c0000.dll | module.1928.1e47c00.7c9c0000.dll | 79 |
| module.940.2061da0.77e70000.dll | module.668.2273020.77e70000.dll | 79 |
| module.940.2061da0.7c9c0000.dll | module.668.2273020.7c9c0000.dll | 79 |
| module.940.2061da0.7e410000.dll | module.868.1e498c8.7e410000.dll | 79 |
| module.668.2273020.71aa0000.dll | module.1928.1e47c00.71aa0000.dll | 77 |
| module.940.2061da0.71aa0000.dll | module.1928.1e47c00.71aa0000.dll | 77 |
| module.940.2061da0.76f20000.dll | module.1928.1e47c00.76f20000.dll | 77 |
| module.940.2061da0.76f20000.dll | module.668.2273020.76f20000.dll | 77 |
| module.940.2061da0.77c10000.dll | module.1928.1e47c00.77c10000.dll | 77 |
| module.940.2061da0.7e410000.dll | module.668.2273020.7e410000.dll | 77 |
| module.668.2273020.7c800000.dll | module.1928.1e47c00.7c800000.dll | 74 |
| module.868.1e498c8.77e70000.dll | module.1928.1e47c00.77e70000.dll | 72 |
| module.940.2061da0.670000.dll | module.668.2273020.1020000.dll | 72 |
| module.940.2061da0.769c0000.dll | module.668.2273020.769c0000.dll | 72 |
| module.940.2061da0.7c800000.dll | module.668.2273020.7c800000.dll | 72 |
| module.940.2061da0.76d60000.dll | module.668.2273020.76d60000.dll | 71 |
| module.668.2273020.77c10000.dll | module.1928.1e47c00.77c10000.dll | 69 |
| module.940.2061da0.76d60000.dll | module.1928.1e47c00.76d60000.dll | 69 |
| module.668.2273020.76bf0000.dll | module.1928.1e47c00.76bf0000.dll | 66 |
| module.668.2273020.7e410000.dll | module.1928.1e47c00.7e410000.dll | 66 |
| module.868.1e498c8.7c800000.dll | module.668.2273020.7c800000.dll | 66 |

| Matched Filename #1 | Matched Filename #2 | Match (in %) |
|---|---|---|
| module.868.1e498c8.7e410000.dll | module.668.2273020.7e410000.dll | 66 |
| module.940.2061da0.7c900000.dll | module.1928.1e47c00.7c900000.dll | 66 |
| module.940.2061da0.5ad70000.dll | module.668.2273020.5ad70000.dll | 65 |
| module.668.2273020.769c0000.dll | module.1928.1e47c00.769c0000.dll | 63 |
| module.940.2061da0.5b860000.dll | module.668.2273020.5b860000.dll | 63 |
| module.668.2273020.5b860000.dll | module.1928.1e47c00.5b860000.dll | 61 |
| module.940.2061da0.77dd0000.dll | module.868.1e498c8.77dd0000.dll | 61 |
| module.668.2273020.77dd0000.dll | module.1928.1e47c00.77dd0000.dll | 60 |
| module.668.2273020.77e70000.dll | module.1928.1e47c00.77e70000.dll | 60 |
| module.668.2273020.77fe0000.dll | module.1928.1e47c00.77fe0000.dll | 60 |
| module.868.1e498c8.77dd0000.dll | module.668.2273020.77dd0000.dll | 60 |
| module.868.1e498c8.77fe0000.dll | module.668.2273020.77fe0000.dll | 60 |
| module.940.2061da0.77dd0000.dll | module.1928.1e47c00.77dd0000.dll | 60 |
| module.940.2061da0.77fe0000.dll | module.1928.1e47c00.77fe0000.dll | 60 |
| module.940.2061da0.77fe0000.dll | module.868.1e498c8.77fe0000.dll | 60 |
| module.668.2273020.7c900000.dll | module.1928.1e47c00.7c900000.dll | 58 |
| module.940.2061da0.7c900000.dll | module.868.1e498c8.7c900000.dll | 55 |
| module.940.2061da0.71ab0000.dll | module.1928.1e47c00.71ab0000.dll | 49 |
| module.940.2061da0.71ab0000.dll | module.668.2273020.71ab0000.dll | 49 |
| module.940.2061da0.77e70000.dll | module.1928.1e47c00.77e70000.dll | 49 |
| module.868.1e498c8.7c900000.dll | module.668.2273020.7c900000.dll | 47 |
| module.668.2273020.77b20000.dll | module.1928.1e47c00.77b20000.dll | 46 |
| module.940.2061da0.77b20000.dll | module.668.2273020.77b20000.dll | 46 |

## F.2 Fuzzy hash similarities between DLL memory samples and carved memory data files

This sub annex lists all fuzzy hash matches between the *dlldump*-based memory samples for PIDs 668, 868, 940 and 1928 and the carved memory data files. The matches are as follows:

*Table 33: Fuzzy hash similarities between Dlldump-based memory samples and carved memory data files (sorted by %).*

| Matching Carved Filename | Matching Dlldump Memory Sample Filename | Match (in %) |
|---|---|---|
| f0263096.dll | module.940.2061da0.76fd0000.dll | 83 |
| f0264224.dll | module.868.1e498c8.7e410000.dll | 68 |
| f0263824.dll | module.1928.1e47c00.77c10000.dll | 66 |
| f0278312.dll | module.668.2273020.776c0000.dll | 66 |
| f0162672.dll | module.940.2061da0.68000000.dll | 65 |
| f0291256.dll | module.940.2061da0.76fc0000.dll | 61 |
| f0282544.exe | module.668.2273020.5cb70000.dll | 60 |
| f0282544.exe | module.940.2061da0.5cb70000.dll | 60 |
| f0163032.dll | module.668.2273020.68000000.dll | 58 |
| f0219248.dll | module.1928.1e47c00.1000000.dll | 58 |
| f0219248.dll | module.868.1e498c8.1000000.dll | 58 |
| f0264224.dll | module.1928.1e47c00.7e410000.dll | 58 |
| f0264224.dll | module.940.2061da0.7e410000.dll | 57 |
| f0264320.dll | module.668.2273020.77dd0000.dll | 57 |
| f0263824.dll | module.668.2273020.77c10000.dll | 54 |
| f0263824.dll | module.940.2061da0.77c10000.dll | 54 |
| f0264320.dll | module.940.2061da0.77dd0000.dll | 54 |
| f0270696.dll | module.1928.1e47c00.5d090000.dll | 52 |
| f0270696.dll | module.668.2273020.5d090000.dll | 52 |
| f0270696.dll | module.940.2061da0.5d090000.dll | 52 |
| f0263288.exe | module.668.2273020.7dba0000.dll | 50 |
| f0161872.dll | module.1928.1e47c00.76bf0000.dll | 47 |
| f0163032.dll | module.940.2061da0.68000000.dll | 47 |
| f0163816.dll | module.940.2061da0.71a90000.dll | 47 |
| f0263736.dll | module.668.2273020.5b860000.dll | 47 |
| f0283192.ttf | module.940.2061da0.76f20000.dll | 47 |

| Matching Carved Filename | Matching Dlldump Memory Sample Filename | Match (in %) |
|---|---|---|
| f0268288.dll | module.940.2061da0.77120000.dll | 46 |
| f0161872.dll | module.940.2061da0.76bf0000.dll | 44 |
| f0263736.dll | module.1928.1e47c00.5b860000.dll | 44 |
| f0263736.dll | module.940.2061da0.5b860000.dll | 44 |
| f0264280.dll | module.1928.1e47c00.7c800000.dll | 44 |
| f0264280.dll | module.868.1e498c8.7c800000.dll | 44 |
| f0264280.dll | module.940.2061da0.7c800000.dll | 44 |
| f0283624.exe | module.668.2273020.47260000.dll | 43 |
| f0264224.dll | module.668.2273020.7e410000.dll | 41 |
| f0263784.dll | module.668.2273020.77a80000.dll | 40 |
| f0267264.dll | module.940.2061da0.77e70000.dll | 40 |
| f0161872.dll | module.668.2273020.76bf0000.dll | 38 |
| f0580472.exe | module.940.2061da0.76fb0000.dll | 38 |
| f0264280.dll | module.668.2273020.7c800000.dll | 38 |
| f0267264.dll | module.668.2273020.77e70000.dll | 38 |
| f0270664.exe | module.668.2273020.7e410000.dll | 36 |
| f0268288.dll | module.1928.1e47c00.77120000.dll | 35 |
| f0162672.dll | module.668.2273020.68000000.dll | 33 |
| f0163960.dll | module.668.2273020.76f50000.dll | 33 |
| f0181384.dll | module.668.2273020.776c0000.dll | 33 |
| f0268288.dll | module.668.2273020.77120000.dll | 33 |
| f0161912.dll | module.940.2061da0.71aa0000.dll | 32 |
| f0840616.pyc | module.940.2061da0.77e70000.dll | 32 |
| f0702416.exe | module.940.2061da0.1000000.dll | 32 |
| f0282048.dll | module.1928.1e47c00.773d0000.dll | 32 |
| f0282048.dll | module.668.2273020.773d0000.dll | 32 |
| f0282048.dll | module.940.2061da0.773d0000.dll | 32 |
| f0283624.exe | module.668.2273020.5cb70000.dll | 32 |

| **Matching Carved Filename** | **Matching Dlldump Memory Sample Filename** | **Match (in %)** |
|---|---|---|
| f0283624.exe | module.940.2061da0.5cb70000.dll | 32 |
| f0161912.dll | module.668.2273020.71aa0000.dll | 30 |
| f0161912.dll | module.1928.1e47c00.71aa0000.dll | 29 |
| f0263776.dll | module.668.2273020.77a80000.dll | 29 |
| f0831408.exe | module.1928.1e47c00.77c10000.dll | 27 |
| f0840616.pyc | module.668.2273020.77e70000.dll | 25 |
| f0245488.dll | module.668.2273020.7dba0000.dll | 21 |

This page intentionally left blank.

# Annex G Commonly used registry keys in a typical malware infection

## G.1 Recommended registry keys for use with Volatility

Based on the author's own use and research of various Windows registry keys commonly used by malware, the following keys are recommended for evaluation. These keys are readily integrated into scripts using appropriate Volatility-based *printkey* plugin commands.

The reader's success in using these keys will undoubtedly vary based on the underlying Windows platform to be analysed and the malware's propensity for using the registry.

The proposed keys have been aggregated and their preceding *HKLM\Software*, *HKLM\System*, *HKCU\Software* and *HKCU* based information were stripped so that they can be readily used by Volatility.

The following keys have been used for evaluation in this work against Stuxnet. Two registry keys in the list below have been highlighted in red because they refer to likely locations for the two malicious device drivers, *MRxCls* and *MRxNet*:

- Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache
- Control Panel\Desktop
- Control Panel\Desktop\ScreenSaveActive
- ControlSet001\Enum\Root\LEGACY_malware\0000
- ControlSet001\services\MRxNet
- ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\AuthorizedApplications\List
- ControlSet001\services\MRxCls
- CurrentControlSet\Control\Session Manager\AppCertDlls
- CurrentControlSet\Control\Session Manager\AppCompatCache\AppCompatCache
- CurrentControlSet\Control\Session Manager\AppCompatibility\AppCompatCache
- CurrentControlSet\Control\SessionManager\Memory Management
- CurrentControlSet\Services
- Microsoft\Active Setup\Installed Components
- Microsoft\DirectPlugin
- Microsoft\Internet Explorer\CustomizeSearch
- Microsoft\Internet Explorer\Main
- Microsoft\Internet Explorer\Main\Default_Page_URL
- Microsoft\Internet Explorer\Main\Default_Search_URL
- Microsoft\Internet Explorer\Main\HomeOldSP
- Microsoft\Internet Explorer\Main\Local Page
- Microsoft\Internet Explorer\Main\Search Bar
- Microsoft\Internet Explorer\Main\Search Page

- Microsoft\Internet Explorer\Main\SearchAssistant
- Microsoft\Internet Explorer\Main\SearchURL
- Microsoft\Internet Explorer\Main\Start Page
- Microsoft\Internet Explorer\Main\Use Search Asst
- Microsoft\Internet Explorer\PhishingFilter
- Microsoft\Internet Explorer\Recovery
- Microsoft\Internet Explorer\Search
- Microsoft\Internet Explorer\Search Bar
- Microsoft\Internet Explorer\Search\CustomizeSearch
- Microsoft\Internet Explorer\Search\SearchAssistant
- Microsoft\Internet Explorer\SearchURL
- Microsoft\Internet Explorer\Toolbar
- Microsoft\Internet Explorer\TypedURLs
- Microsoft\Windows Defender\Real-Time Protection\EnableKnownGoodPrompts
- Microsoft\Windows Defender\Real-Time Protection\EnableUnknownPrompts
- Microsoft\Windows Defender\Real-Time Protection\ServicesAndDriversAgent
- Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Run
- Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Runonce
- Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\RunonceEx
- Microsoft\Windows NT\CurrentVersion\Windows
- Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs
- Microsoft\Windows NT\CurrentVersion\Windows\Load
- Microsoft\Windows NT\CurrentVersion\Winlogon
- Microsoft\Windows NT\CurrentVersion\Winlogon\Notify
- Microsoft\Windows NT\winlogon\userinit
- Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects
- Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedMRU
- Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU
- Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
- Microsoft\Windows\CurrentVersion\Explorer\RunMRU
- Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler
- Microsoft\Windows\CurrentVersion\Explorer\ShellExecuteHooks
- Microsoft\Windows\CurrentVersion\Explorer\UserAssist
- Microsoft\Windows\CurrentVersion\Internet Settings
- Microsoft\Windows\CurrentVersion\Internet Settings\EnableAutodial
- Microsoft\Windows\CurrentVersion\Internet Settings\EnableHttp1_1
- Microsoft\Windows\CurrentVersion\Internet Settings\MaxConnectionsPer1_0Server
- Microsoft\Windows\CurrentVersion\Internet Settings\MaxConnectionsPerServer
- Microsoft\Windows\CurrentVersion\Internet Settings\ProxyEnable
- Microsoft\Windows\CurrentVersion\Internet Settings\ProxyHttp1.1

- Microsoft\Windows\CurrentVersion\Internet Settings\ProxyOverride
- Microsoft\Windows\CurrentVersion\Internet Settings\ProxyServer
- Microsoft\Windows\CurrentVersion\Internet Settings\Zones\0
- Microsoft\Windows\CurrentVersion\Internet Settings\Zones\1
- Microsoft\Windows\CurrentVersion\Internet Settings\Zones\2
- Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
- Microsoft\Windows\CurrentVersion\Run
- Microsoft\Windows\CurrentVersion\RunOnce
- Microsoft\Windows\CurrentVersion\RunOnce\Setup
- Microsoft\Windows\CurrentVersion\RunOnceEx
- Microsoft\Windows\CurrentVersion\RunServices
- Microsoft\Windows\CurrentVersion\RunServicesOnce
- Microsoft\Windows\CurrentVersion\SharedDLLs
- Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad
- Microsoft\Windows\CurrentVersion\URL
- Microsoft\Windows\CurrentVersion\URL\DefaultPrefix
- Microsoft\Windows\CurrentVersion\URL\Prefixes
- Microsoft\Windows\ShellNoRoam\MUICache

These keys can be readily integrated into scripts. For example, consider the following Volatility *printkey* command:

```
$ volatility -f stuxnet.vmem printkey -o 0xe1991b60 -K
'Microsoft\Windows\CurrentVersion\RunServices'
```

A script built such commands requires only a few minutes to construct, based on the physical memory addresses listed in the above Table 29, used in conjunction with various command line tools including *cat*, *awk* and *sed*.

## G.2 Root Registry Keys

The author proposed registry keys are based on the following root registry keys:

HKEY_CURRENT_USER

HKEY_CURRENT_USER\Software

HKEY_LOCAL_MACHINE\Software

HKEY_LOCAL_MACHINE\System

This page intentionally left blank.

# Bibliography

Carbone, Richard. Malware memory analysis for non-specialists: Investigating a publicly available memory image of the Zeus Trojan horse. Technical Memorandum. Defence R&D Canada – Valcartier. TM 2013-018. April 2013.

Carbone, Richard. Malware memory analysis for non-specialists: Investigating publicly available memory images for Prolaco and SpyEye. Technical Memorandum. Defence R&D Canada – Valcartier. TM 2013-155. October 2013.

Carbone, Richard. Malware memory analysis for non-specialists: Investigating publicly available memory image 0zapftis (R2D2). Technical Memorandum. Defence R&D Canada – Valcartier. TM 2013-177. October 2013.

Volatility. CommandReference: Example usage cases and output for Volatility 2.0 commands. Online command reference. Volatility. February 2012. http://code.google.com/p/volatility/wiki/CommandReference.

## List of symbols/abbreviations/acronyms/initialisms

| | |
|---|---|
| AES | Advanced Encryption Standard |
| API | Application Programming Interface |
| ASCII | American Standard Code for Information Interchange |
| AV | Anti-Virus or Antivirus |
| C&C | Command & Control |
| CFNOC | Canadian Forces Network Operations Centre |
| CORFC | Centre d’opérations des réseaux des Forces canadiennes |
| COTS | Commercial Off The Shelf |
| CTPH | Context Triggered Piecewise Hash<br>Sometimes known as fuzzy hash or *ssdeep* hash |
| CVE | Common Vulnerabilities and Exposures |
| DLL | Dynamically Loaded Library |
| DND | Department of National Defence |
| DRDC | Defence Research & Development Canada |
| DRDKIM | Director Research and Development Knowledge and Information Management |
| EDT | Eastern Daylight Time |
| EXT4 | Fourth Extended Filesystem |
| FOSS | Free and Open Source Software |
| FTP | File Transfer Protocol |
| GICT | Groupe intégré de la criminalité technologique |
| GRC | Gendarmerie Royale du Canada |
| HKCU | HKEY_LOCAL_USER |
| HKLM | HKEY_LOCAL_MACHINE |
| ID | Identification |
| IP | Internet Protocol |
| ITCU | Integrated Technological Crime Unit |
| MAC | Mandatory Access Control |
| MD5 | Message Digest Algorithm 5 |
| MiB | Mebibyte |

| | |
|---|---|
| N/A | Not Available |
| NIST | National Institute of Standards and Technology |
| NSRL | National Software Reference Library |
| NTP | Network Time Protocol |
| PAE | Physical Address Extension |
| PE | Portable Executable |
| PID | Process ID |
| PPID | Parent Process ID |
| R&D | Research & Development |
| RAM | Random Access Memory |
| RCMP | Royal Canadian Mounted Police |
| RDDC | Recherche et Développement pour la Défense Canada |
| RDP | Remote Desktop Protocol |
| RSA | Ron Rivest, Adi Shamir and Leonard Adleman |
| SHA1 | Secure Hash Algorithm-1 |
| SSL | Secure Sockets Layer |
| TCP | Transmission Control Protocol |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| TID | Thread ID |
| UDP | User Datagram Protocol |
| UPX | Ultimate Packer for eXecutables |
| URL | Uniform Resource Locator |
| UTC | Coordinated Universal Time |
| VAD | Virtual Address Descriptor |
| VMEM | Virtual Memory |

# Glossary

**_Eprocess**

See Eprocess.

**_Ethread**

See Ethread.

**_Kthread**

See Kthread.

**Anti-Virus**

An Anti-virus, AV, or AV scanner is a software system or framework which is used to, at a minimum, scan a given system for signs of malware infection. This software may not just be a scanner but may also include system-protection and anti-malware detection and prevention capability.

**AV Scanner**

See Anti-Virus.

**Computer Memory Image**

See Memory Image.

**Context Triggered Piecewise Hash**

See Fuzzy Hash.

**Data Carving**

Commonly known as file carving, data carving is the process or act of recovering known data structures, generally based on recognized file patterns. Data carving only works on contiguous data structures as the recovery of fragmented data is not supported by most data recovery software and those that do only support a very limited number of file formats.

**DLL Injection**

DLL injection is a method for forcing programs to run in a manner their programmers did not design for or foresee. Under Windows, there are various methods for implementing this, some through the registry while others are carried out using APIs.

**Eprocess**

The Eprocess is a kernel-based process-specific data structure that encompasses a process' state-based information. This structure has a forward and backward pointer to active processes.

**Ethread**

An Ethread is used to identify threads to be worked on. Its structure describes the various aspects of the process or thread to be worked such as thread starting address and thread ID. It is also is a semi-opaque data structure. Unlike a Kthread structure, it is processor agnostic.

**Ext4**

Ext4 is the latest Ext-based filesystem of the Linux operating system that supersedes Ext2/3. It continues providing filesystem journaling. It also provides greater performance, reliability and allows for much larger file and filesystem sizes. This filesystem is natively supported by Linux.

**Fuzzy Hash**

This is a specific type of file hashing which has the ability to identify file similarities, usually represented as a percentage.

**Handle**

A handle is a pointer-like resource-based reference used to a specific system resource. Handles are abstract references to resources available within a given computer system. Under Windows, many types of handles exist but common examples pertain to files, directories, the registry and system based devices. It should not be confused with file handles.

**Hash**

A hash, commonly referred to as a file hash, is a reduced representation of some arbitrary data file by passing it through some cryptographic hashing algorithm. In so doing, a unique hash value should be emitted by the hashing program that can be used to identify and authenticate a given file's integrity and uniqueness against a set of hashes, commonly known as a hash-set. SHA1 and CTPH hashes are examples of hashing algorithms.

**HashKeeper**

HashKeeper is an MD5-based investigative signature file that was developed and maintained by the National Drug Intelligence Center. It contains known good and bad signatures for an array of files, including illicit images. Developed by the law enforcement community, various agencies, nationally and internationally contributed signatures. However, the source of many of the incorporated signatures are either not known or arrived at by non-forensic means; as such, these signatures are not accepted in a court of law.

**IRP Hook**

An IRP Hook is a kernel-based interception technique some rootkits, viruses and Trojan horses use in order to hide themselves from detection.

**Kthread**

A Kthread is a thread/process-based management kernel-specific data structure. It is similar to an Ethread but contains processor-specific data structures such as stack limit, lock and thread states. It also describes various aspects of the underlying processor-specific features and it is more opaque than an Ethread data structure.

**Memory Image**

A memory image or computer memory image is a bit-copy of a system's RAM. For physical computer systems, it is acquired through a memory-imaging program. In virtualized environments, memory can be acquired by an imaging program or by saving or dumping the virtual machine's memory state.

**Mutex**

A mutex is a Windows-based object used to provide exclusive access to a shared system resource. These resources can only be accessed one at a time, thus by issuing a mutex or mutual exclusion, a process or thread can be allocated said resource when it becomes available for use.

**Pagefile**

The pagefile is the operating system's swap file, swap device or swap space.

**Privilege Escalation Attack**

This type of attack takes advantage of bugs or errors in software and various operating system components that allow an attacker to run arbitrary code that runs at the system privilege of the exploited program. For example, if Windows program running with *Local System* is successfully exploited and the attacker is able to run or feed arbitrary code through that exploited program, then it will run at the system level of privilege.

**Process Injection**

See DLL Injection.

**SHA1 (Secure Hash Algorithm-1) Hash**

The SHA1 hash is a 160-bit cryptographic hash commonly used for forensic file identification and authentication.

**SSL (Secure Sockets Layer)**

SSL is a client-server TCP/IP Application Layer protocol. It is commonly used for the exchange of cryptographic keys that will be used to establish a "secure" communications channel between two systems.

**Strings Command**

The *strings* command is capable of extracting 7, 8, 16 and 32-bit text patterns from an arbitrary data file which can be text or binary based. 7-bit extraction represents the first 128 ASCII characters while 8-bit extraction represents the extended ASCII character set. 16 and 32-bit strings are typically reserved for Unicode-based text. Thus, the command line parameters required to instruct the *strings* command to perform 7, 8, 16 or 32-bit text extraction is *-s*, *-S*, *-l* and *-L*, respectively.

**Thread**

A thread is typically a subset process. A thread contains only the code necessary to perform a set of instructions. In single-threaded programs, a thread represents the program's executable code and stack while in multi-threaded applications a thread performs just one piece of the work that is distributed across multiple threads. These threads then typically communicate with each other through various inter-process mechanisms.

**Trojan horse**

A Trojan horse is a malicious non-replicating infectious computer program. It infects a computer when the delivery software is run at which time a payload is instantiated that does the actual infecting. However, Trojan's do not typically infect computers the way viruses do. As such, they do not generally infect computer files. The program delivering the payload is known as a dropper. The payload achieves its objective by gaining some form of administrative level privileges in the target's operating system, typically through subversion. A Trojan's typical objective is to provide backdoor access but it can also be used for other capabilities including data and information theft, arbitrary or specific data file encryption, inflict damage to the operating system or its data files, and in rare cases, even attempt to damage a system's hardware components.

**Unlinked DLL (or file)**

Unlinking a DLL or other file such as an executable or library is a common method malware and other malicious processes use to hide the fact that they may be using one of these resources covertly. Volatility's *ldrmodules* plugin supports several unlinked validation tests. It should be used to test for the existence of unlinked files associated to a process.

**UPX**

UPX is an open source data compression algorithm used to compress executable files. UPX executable file packers exist for Windows, Linux, Mac OS X and other platforms.

**UserAssist**

It is a series of user-based Windows registry keys containing information about various actions undertaken by a user (e.g., launching a specific program).

**Vmem**

A Vmem file is a VMware virtual machine-based paged memory file. It is generated when a virtual machine's state is saved containing the entire RAM allocated to that virtual machine.

**Worm**

Sometimes known as a computer or network worm, a worm is a malicious program designed to spread to as many computer systems as possible, usually by means of a network. Worms do not typically cause much, if any, damage to the underlying computer system. Instead, due to their need to replicate often consume not only a network's available bandwidth but crash underlying computer systems as they sometimes overwhelm the resources of those system as they attempt to propagate. Worms typically spread only to systems susceptible to the vulnerabilities necessary for their infection. Thus, unaffected systems do not become infected.

**Zero-day Exploit (sometimes referred to as Zero-day Attack)**

This is an attack or exploit carried out against a system or application that is currently unknown to others. Because the exploit is unknown, there would be no known patches or fixes for it, until information or news about it becomes known.

**DOCUMENT CONTROL DATA**

(Security markings for the title, abstract and indexing annotation must be entered when the document is Classified or Designated)

| | |
|---|---|
| 1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g., Centre sponsoring a contractor's report, or tasking agency, are entered in Section 8.)<br><br>Defence Research and Development Canada – Valcartier<br>2459 Pie-XI Blvd North<br>Quebec (Quebec)<br>G3J 1X5 Canada | 2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.)<br><br>UNCLASSIFIED<br><br>2b. CONTROLLED GOODS<br><br>(NON-CONTROLLED GOODS)<br>DMC A<br>REVIEW: GCEC APRIL 2011 |

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Malware memory analysis for non-specialists : Investigating publicly available memory image for the Stuxnet worm

4. AUTHORS (last name, followed by initials – ranks, titles, etc., not to be used)

Carbone, R.

| | | |
|---|---|---|
| 5. DATE OF PUBLICATION (Month and year of publication of document.)<br><br>November 2013 | 6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)<br><br>122 | 6b. NO. OF REFS (Total cited in document.)<br><br>26 |

7. DESCRIPTIVE NOTES (The category of the document, e.g., technical report, technical note or memorandum. If appropriate, enter the type of report, e.g., interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Scientific Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence Research and Development Canada – Valcartier
2459 Pie-XI Blvd North
Quebec (Quebec)
G3J 1X5 Canada

| | |
|---|---|
| 9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)<br><br>31XF20 « MOU RCMP "Live Forensics" » | 9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.) |
| 10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)<br><br>DRDC-RDDC-2013-R1 | 10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.) |

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

This report examines how an investigator can analyse an infected Windows® memory dump. The author investigates how to carry out such an analysis using Volatility and other investigative tools, including data carving utilities and anti-virus scanners. Volatility is a popular and evolving open source-based memory analysis framework upon which the author has proposed a memory-specific methodology for aiding fellow novice memory analysts. The author examines how Volatility can be used to find evidence and indicators of infection. This report is the fourth in this series concerning Windows malware-based memory analysis. This current work examines a memory image infected with the Stuxnet worm.

Dans ce rapport, on décrit comment un enquêteur procède pour analyser l’image mémoire d’un système Windows® infecté. L’auteur étudie les techniques d’analyse au moyen de Volatility et d’autres outils tels que les utilitaires de récupération de données et les scanneurs antivirus. Volatility est un cadre populaire et évolutif d’analyse de la mémoire de source ouverte sur lequel l’auteur s’appuie pour proposer une méthodologie propre à la mémoire dans le but d’aider ses collègues analystes novices. L’auteur examine comment Volatility peut être utilisé pour trouver des preuves ou des indices d’infection. Ce rapport est le quatrième d’une série consacrée à l’analyse de la mémoire dans un environnement Windows® infecté par un maliciel. Le présent ouvrage porte sur l’image mémoire infectée par le ver Stuxnet.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g., Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Antivirus; Anti-virus; Computer forensics; Digital forensics; Digital forensic investigations; Forensics; Infection; Malware; Memory analysis; Memory image; Rootkit; Scanners; Stuxnet; Virus scanner; Volatility; Windows; Worm