Malware memory analysis for non-specialists

Investigating publicly available memory image for the Tigger Trojan horse

R. Carbone

EC-Council Certified Forensic Investigator (CHFI)
SANS GIAC Certified GCIH and GREM
DRDC - Valcartier Research Centre

Defence Research and Development Canada

Scientific Report
DRDC-RDDC-2014-R28
June 2014

Abstract

This report examines how a computer forensic investigator can effectively analyse an infected Windows memory dump. The author investigates how to perform 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's previously proposed memory-specific methodology could be of aid to fellow novice memory analysts. The author examines how Volatility can be used to find evidence and indicators of infection. This report is the fifth in this series concerning Windows malware-based memory analysis. It examines a memory image infected with the Tigger/Syzor Trojan horse.

Significance to defence and security

Canadian Armed Forces (CAF) networks are a target of choice for malware and directed attacks. This series of reports provides junior incident handlers with the necessary knowledge to handle and mitigate complex malware-based attacks from a memory image, using non-expert level techniques. Junior incident handlers can be of immense help to current and ongoing investigations, as opposed to relying entirely on a few select expert software reverse engineers.

Résumé

Ce rapport examine comment un enquêteur en informatique judiciaire peut analyser de façon efficace une image mémoire Windows infectée. L'auteur étudie comment effectuer une telle analyse en utilisant Volatility ainsi que d'autres outils d'enquête, tels que des utilitaires de récupération de données et des logiciels antivirus. Volatility est un cadriciel populaire et évolutif à code source ouvert pour l'analyse de mémoire et pour lequel l'auteur avait précédemment proposé une méthodologie spécifique afin d'aider des collègues analystes novices. L'auteur examine comment Volatility peut être utilisé afin de trouver des preuves et des indicateurs d'infection. Ce rapport est le cinquième d'une série sur l'analyse de logiciels malveillants Windows basée sur la mémoire. Celui-ci porte sur une image mémoire infectée par le cheval de Troie Tigger/Syzor.

Importance pour la défense et la sécurité

Les réseaux des Forces Armés Canadiens (FAC) sont une cible de choix pour les logiciels malveillants et les attaques dirigées. Cette série de rapports fournit à des gestionnaires d'incidents de niveau junior les connaissances nécessaires pour gérer et atténuer des attaques complexes par des logiciels malveillants à partir d'une image mémoire, en utilisant des techniques applicables pour des non-experts. Des gestionnaires d'incidents de niveau junior peuvent être d'une aide immense pour des enquêtes actuelles et de longue durée, plutôt que de compter uniquement sur un petit nombre de rétro-ingénieurs logiciels.

Table of contents

List of tables

Acknowledgements

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

Disclaimer policy

It must be understood from the outset that this report examines computer malware and that handling virulent software is not without risk. As such, the reader should ensure that he has taken all the necessary precautions to avoid infecting his own computer system and those around him, whether on a corporate network or isolated system.

The reader must neither construe nor interpret the work described herein by the author as an endorsement of the aforementioned techniques and capacities as suitable for any specific purpose, construed, implied or otherwise. Moreover, the author does not endorse the specific use of any specific anti-virus product, the use of Volatility or any data carving technology. Many similar software tools, utilities and scanners exist beyond those used herein. They may be commercial or free and open source in nature and as such, the onus is on the reader to determine which software best suits his specific needs. While the author felt most comfortable working from within a Linux environment, the author does not specifically recommend the use of such a system for the reader. Instead, the reader should use the environment in which he is most comfortable.

Furthermore, the author of this report absolves himself in all ways conceivable with respect to how the reader may use, interpret or construe this report. The author assumes absolutely no liability or responsibility, implied or explicit. Moreover, the onus is on the reader to be appropriately equipped and knowledgeable in the application of digital forensics. Due to the offensive nature of computer malware, the author is no way responsible for the reader's use of any malware, whether examined herein or otherwise, in any offensive or defensive nature against any entity, even against the reader himself, for any purposes whatsoever.

Finally, the author and the Government of Canada are henceforth absolved of all wrongdoing, whether intentional, unintentional, construed or misunderstood on the part of the reader. If the reader does not agree to these terms, then his copy of this scientific report must be destroyed. Only if the reader agrees to these terms should he or she continue in reading it beyond this point. It is further assumed by all participants that if the reader has not read said Disclaimer upon reading this report and has acted upon its contents then the reader assumes all responsibility for any repercussions that may result from the information and data contained herein.

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 dumped using various Volatility plugins: 1. the use of the terms processes, memory sample files and memory dump files are used interchangeably; 2. the use of the terms fuzzy hash matching and similarity matching are used interchangeably and 3. the use of indicators of compromise and indicators of infection are used interchangeably.

Finally, the use of masculine is employed throughout this text for purposes of simplification.

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, 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.

Errata for previous memory analysis reports

With respect to the author's memory analysis of Stuxnet using Volatility, three errors were caught by the author after the report's publication. The errors and corrections are as follows:

Section 2.3.4.2:

Error 1: Using all proposed registry keys identified in Annex E

Correction 2: Using all proposed registry keys identified in Annex F

With respect to the author's memory analysis of R2D2 using Volatility, two errors were caught by the author after the report's publication. The errors and corrections are as follows:

Annex E.1:

Error 1: ControlSet001\services\malware

Correction 1: CurrentControlSet\services\malware

Error 2: ControlSet002\services\malware

Correction 2: CurrentControlSet\services\malware

However, only one instance of *CurrentControlSet\services\malware* is needed.

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 Tigger/Syzor. This document is the fifth in a series of reports. 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 without the need of additional guidance. The first report written by the author in this series examined the Zeus Trojan horse, found in DRDC Valcartier TM 2013-018 [1], while the second examined the Prolaco worm and SpyEye Trojan horse, found in DRDC Valcartier TM 2013-155 [2]. The third report examined the R2D2 Trojan horse, available in DRDC Valcartier TM 2013-177 [3]. The fourth report examined the Stuxnet worm; it can be found in DRDC Scientific Report DRDC-RDDC-2013-R1 [4]. Each analysis is progressively more difficult, thereby initiating the reader to more complex memory-based investigations.

This specific report examines the Tigger/Syzor Trojan horse, a highly difficult to detect piece of malware, in order to complement an ongoing assembly of quality tutorials necessary for building 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 [1][2][3][4] examines various Windows-based malware-infected memory images to 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 a learning guide. 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 a comprehensive investigative context. Thus, the author has strived to ensure that his investigative techniques are well documented, even if portions of an investigation are unsuccessful, in order to ensure that a coherent investigative context is used.

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 Tigger/Syzor was procured from the following location: http://code.google.com/p/volatility/wiki/PublicMemoryImages. The image's metadata, in uncompressed form, is as follows:

Table 1: Infected memory image metadata.

Memory image name	Size (in MiB)	SHA1 hash value
tigger.vmem	128 (exactly)	d9e470914354c87cdd21257abcf56a32c588e210

1.5 Data carving

An in-depth examination of data carving can be found in two previous Technical Memorandums written by the author, specifically [1][5].

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 mechanisms employed by the Tigger/Syzor Trojan horse, this investigation relies heavily on scanners, as was done for the reports [3] and [4].

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][4]. These six anti-virus 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 November 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]. Volatility 2.2 is no longer being used for memory analysis; it has been superseded by Volatility 2.3.

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

1.8 Investigative methodology

This overall investigative methodology, first proposed in [1], updated in [2], and further clarified in [4], has been partially revised to better reflect the complexities of memory-based malware investigations. This methodology provides a clear approach to conduct forensic memory investigations for non-reverse engineers and memory specialists. As additional memory infections are examined by the author in ensuing reports, this methodology will likely be updated to reflect other techniques required to adequately investigate memory images.

Beginning with the results obtained through the scanning, carving and various Volatility plugins, reasoning is applied to the information obtained from these sources in order to attempt determine the source or means of the infection. Additional processing is then applied to identify (through AV scanning, string or hexadecimal analysis) malware in the hopes of determining how the infection took hold.

This methodology can be summarised using the following steps:

1.8.1 Step 1 - Protect the disk image

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

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.

1.8.2 Step 2 - Preliminary memory image scanning

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

1. Some scanners (e.g., Avast) can perform in-depth analysis of memory images and in some 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 used by the investigator.

2. Save the output from the various scanners.

1.8.3 Step 3 - Data carving of memory image

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

1. It is suggested to use one highly capable data-carving tool (Photorec) instead of several mediocre tools.

2. Perform hashing against all carved memory data files:

    a. SHA1 hashing:

        i. Determine if any SHA1 matches can be identified against known hashsets (e.g., NSRL or HashKeeper hashsets). Save any identified hashset matches.

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

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

            3. The application of "good" and "bad" files is optional. If they are not applied, then all carved memory data files, upon hashing, are 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.

            1. Identify the fuzzy hashes of the variously carved memory data files and then compare them against all the other carved memory data files in order to identify similarities between them. This is done using the ssdeep tool.

3. Run the various 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 that a carved data memory file contains suspicious or malicious content, then it should be considered of interest. The more scanners corroborating a file's suspicious or malicious nature, the more it should be considered potentially 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 memory data files. Executable-like files (e.g., EXEs, DLLs, drivers) 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. Strings analysis should be conducting using 7, 8 and 16 and 32-bit strings as per the UNIX *strings* command.

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

4. Decide if pursuing the analysis is worthwhile:

    a. Based on scanning results obtained from the memory image itself (Step 2) and subsequent scanning of carved data files (Step 3), there may be no indications of infection.

    b. If a multi-scanner approach with up-to-date scanners yielding no tangible indications of infection, it is very likely that the underlying memory image is infection free. In such situation and based on departmental outlines, policies or recommendations, determine if the memory image is to be subjected to further analysis or the case should be closed.

1.8.4 Step 4 - Process-based Volatility plugin 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 establish how it occurred:

1. For an **unreadable** or **unprocessable** memory image:

    a. It is possible that investigative endeavours using Volatility will not yield tangible results. If a memory image cannot be processed using Volatility, it is likely that it is corrupt or unreadable. This could have been caused by errors during memory acquisition or the use of an unknown memory image file format.

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

    c. It may be worth the effort to attempt a *strings*-based analysis.

    d. Otherwise, **terminate** the investigation at this point.

2. For a **functional memory** image:

    a. Using non-reverse engineering Volatility-based 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*, etc.) before using more advanced plugins.

        ii. Plugins by themselves are not likely sufficient to put together the pieces, unless something is especially obvious. Instead, complex analyses may be required to identify potential indicators of infection, which may provide a picture of what occurred, enabling an analyst to piece together at least some of the events leading up to the infection.

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 to dump them from memory using appropriate Volatility plugins:

i. Plugins exist to dump DLLs, processes, arbitrary data files, processes and their memory space and drivers (currently applies to Volatility 2.3) from a memory image. Moreover, the *dump* Volatility plugin enables an investigator to dump additional data types from a given memory image.

ii. All dumped processes, process memory space, arbitrary files, DLLs, drivers or any other dumped file types 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 memory data files.

1. Using all available scanners, determine which of the dumped memory sample files (DLLs, process, etc.) are identified as malicious or suspicious. Cross-scanner correlation is important and samples detected by more than one scanner should be prioritized.

2. Dumped memory samples are to be hashed (SHA1 and fuzzy) and then compared against appropriate sources of known files hashes (i.e., NSRL, HashKeeper, etc.) and the various hashes derived from the carved memory data files.

a. In so doing, it may be possible to identify duplicates and similarities between some of the files obtained through the carving of the memory file and those dumped from the memory image using Volatility.

b. Consider that often files dumped and carved from memory may be similar, but not identical, due to fundamental differences in how they are extracted from an arbitrary memory image. Having access to both the carved and dumped versions of a file and comparing their similarities can help an investigator determine if carving or Volatility plugin-based processing and dumping yielded similar results.

3. Based on the previously obtained scanner results, work backwards to identify those Volatility-based dumped memory samples that correlate to against the carved memory data files.

iii. Any files dumped from the memory image that the investigator deems potentially suspicious as based on one or more indicators of infection (e.g., DLLs linked to a highly suspicious process or an oddly named device driver) that were not identified through scanning should be analysed using alternative means.

1. Alternative means includes, but is not limited to, string and hexadecimal analysis. Reverse engineering efforts are not examined in this methodology although for those with such skills, nothing precludes them from undertaking

such analyses. Specific file and hex editors provide the ability to parse and analyse various file formats. Very often, malicious software (DLLs, drivers and processes) will contain suspicious keywords, values, strings or specific Windows API functions.

a. Some whitelists and blacklists for Windows APIs are available, but they must be used with caution. The analyst must be aware that they may not be from legitimate sources and that some of the APIs may only apply to specific versions of Windows.

b. It is also important to understand that many of the meaningful strings extracted from a given program, process, DLL, device driver, etc. will largely depend on the underlying programming language that was used. While some strings tend to be universal to executables (including DLLs and drivers), others are entirely unique to specific programming languages.

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

a. It is possible that the underlying data file, executable (program, process, device driver, etc.) is encrypted, packed, or possibly damaged or corrupted (as may be expected).

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

1. However, even if the malware is no longer in the memory image, 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).

2. Possible reasons why the malware may no longer be memory-resident include:

a. Malware can force the system to page out most of its code so that it cannot be readily dumped or analysed from memory. Direct pagefile analysis, without memory contents is very difficult, making it a good location to hide.

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

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

a. If it is not possible to obtain a disk image, **cease and terminate** the analysis.

b. If a disk image is available, it can be analysed for evidence of infection. This type of analysis is not examined herein.

c. With the maximum amount of information thus having been obtained by the investigator, using applicable Volatility plugins from the memory image, in conjunction with various hash matches (SHA1 and fuzzy) and anti-virus scanner results, a picture should begin forming around the various fragments of evidence and information.

v. The use of multiple virus scanners helps to ensure that picking up malicious software and other memory resident objects increases. However, there is no guarantee that for a given malware, it or its components will be detected, regardless of the comprehensive scanning of Volatility-dumped objects or carved memory data files.

vi. Moreover, similarity hashes between objects detected as possibly malicious or suspicious and those that were not, but which share high levels of similarity, should be flagged.

vii. All these pieces of information, taken together, may form an obvious picture of the malware, how it remained resident (i.e., device drivers, rootkits, etc.) and possibly how it infected the system. The latter may be obvious, depending on the sophistication of the malware and its ability to hide itself from post-mortem analysis. This will largely depend on the complexity and techniques used by the malware to evade detection, both live and post-mortem.

viii. The malware may have left behind indications of its various components: odd or nonconforming filenames, systems with which it was possibly communicating and perhaps even indications of a malware dropper. In short, there are many potential indicators of compromise, as based on the wide assortment of applicable Volatility plugins.

ix. However, highly sophisticated malware that employ advanced hiding techniques may not be so obvious to coalesce into a cohesive picture. In some instances, the only indication that something is amiss is the fact that information about processes or other memory resident objects is not available, appears to be lacking or is in excess (e.g., process with too few or too many loaded DLLs, device driver which uses odd references, etc.).

1.8.5 Step 5 - Windows registry based Volatility memory analysis

Windows registry can be long and complex to analyse, but Volatility provides various Windows-specific plugins to aid with this process:

1. Extract UserAssist and ShellBag keys from the memory image using corresponding Volatility plugins.

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

3. Armed with a list of potential registry keys commonly used by malware, provided by the author at the end of this report, 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 investigator is free to choose whatever approach he is most comfortable using.

        ii. The author provides a list of commonly used registry keys targeted by malware as listed in Annex F.

4. Reading the script's output typically takes only a few minutes. Registry keys containing evidence or indicators of compromise are not generally difficult to identify. 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.

5. 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 suspected 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:

            1. Selecting or choosing appropriate keywords or wordlists is not always easy to develop. At a minimum, they should be context sensitive and reflect the evidence thus far obtained.

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

6. Other non-strings tools exist including Didier Stevens XORSearch and XORStrings. These tools are very useful for attempting to find text hidden using a XOR'ing-based technique against an arbitrary data file (e.g., memory image, process dump, etc.).

1.8.6 Step 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. Commonly used FOSS tools include *aeskeyfind*, *rsakeyfind* and *interrogate*.

        iii. COTS software typically finds encryption key(s) in the memory image that is then passed on to the tool's decryption component.

        iv. Unidentified encryption-like keys identified in a given memory image may be network-specific encryption keys:

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

            2. This delves into cryptography and reverse engineering techniques, which is outside the scope of the methodology.

2. If rootkits or malicious device drivers were found, attempt to establish their underlying capabilities:

    a. If a suspicious or malicious device driver or rootkit is dumped from the memory image, using Volatility's *devicetree* plugin, it may be possible for investigators to determine the underlying devices these drivers/rootkits use or create. This would provide valuable information about their underlying capabilities.

2 Memory investigation and analysis of Tigger

2.1 Background

2.1.1 Context

This analysis examines a memory image suspected of harbouring the Tigger/Syzor Trojan horse, as based on the methodology put forward in Section 1.8. Much information concerning this specific infection is available from web-based sources. References [6][7][8][9][10][11][12][13] and [14] provide a wealth of information.

However, in contrast to certain previous reports by the author that used malware reports for information regarding the investigative analysis of the underlying infection [1][2], this specific report does not. Instead, the cited reports and other sources of information are for the reader's use rather than for the current investigative analysis. Moreover, as with previous analyses [1][2][3][4], no use was made of existing Volatility analyses with respect to this specific infection.

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 aforementioned methodology, will also point out detected anomalies that may indicate potential indicators of compromise or other infection-based evidence.

It is important to note that while the methodology as outlined in Section 1.8 has six steps, the analysis as conducted herein has been broken down into three steps.

2.1.2 About Tigger

The Tigger/Syzor Trojan horse is a highly advanced malware that was spread within financial circles, specifically having targeted particular financial companies including stockbrokers. The story concerning this malware was first broken by security firm iDefense analyst Michael Hale Ligh. The malware has purportedly claimed more than a quarter million victims since first having been identified November 2008 although it is possible that the actual number of infected systems is higher [8][9][10][11][12][14].

Counted among the malware's victims are E-Trade, ING Direct ShareBuilder, Vanguard, Options Xpress, TD Ameritrade and Scottrade [9][10][11][12]. The malware's origins may have its origins in the Russian Federation, but this is speculative [6][13]. The exact propagation mechanism is not currently known, as it has not been publicly disclosed. Moreover, no major anti-virus vendor has to date published a meaningful report or analysis concerning this specific infection.

The malware has advanced capabilities including a keylogger that does not require the use of hooks, and unlike most other malware, this Trojan takes unprecedented steps to clean out other forms of malware that may already be infecting the victim system. It removes these malware as they may interfere with its attempts to "coast" below the radar. Moreover, it also has the ability to disable various anti-virus and malware protection mechanisms and software, both installed prior to and after Tigger infection. It infects systems by taking advantage of Windows privilege escalation attack MS08-066 using a very similar exploit found on the Milw0rm exploit code web site. Finally, the Trojan apparently has the ability to install a rootkit, conduct DLL injection, steal cookies and web certificates, sniff FTP and POP3 passwords from the network and even take screenshots [6][8][9][10][11][12][13][14].

2.2 Preliminary investigative steps

The steps examined in this subsection are considered necessary for examining a potentially infected memory image.

2.2.1 Safeguard the memory image

The memory image *tigger.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 tigger.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, none detected the memory image as infected.

Preliminary scanner examination indicated that this memory image was not infected with malware. Thus, a more in-depth analysis will be required. All anti-virus scan results were recorded and saved.

2.2.3 Data carving, file hashing and file type identification

Photorec succeeded in recovering 525 files carved from the memory image as per the recommended Photorec settings put forward in Section 1.7.2. Of these files, no duplicates were identified as based on their SHA1 hashes. Of the 525 recovered files, 270 were identified as PE-based files. Of these, 211 were identified as Windows 32-bit DLLs, while 59 were identified as standard Windows 32-bit PEs (no device drivers detected[1]).

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.

[1] Typically, the *file* command identifies PE device drivers as *PE32* (or *PE64* if they are 64-bit) *executables (native) Intel* ... Conversely, typical executables are generally identified as console or GUI-based.

All recovered files were SHA1-hashed of which no duplicates were identified through their hashes. These were then validated against NSRL hash-set 2.41 (June 2013). Results were stored for future use. Eight unique SHA1 hashes were confirmed as matching against the NSRL hash-set. However, in all 26 unique SHA1-based NSRL filename matches were identified, as listed in Annex B.

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

Finally, the variously carved files were examined using the *file* command to identify any suspicious files such as but limited to packed executables. However, none were found. This analysis was carried out from within the directory where the variously carved files resided using the following command:

```
$  find  -type  f  -print0  |  xargs  -0  file  |  awk
'{$1="";print}' | sort | uniq
```

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 matches were established. The only case detected by multiple scanners was for file *f0131152.exe*, identified by Avast, AVG, F-Prot and McAfee, as shown in the following table:

Table 3: Multi-scanner results of a potentially infected carved memory file.

Scanner	Carved Memory File	Infection Identification
Avast AVG F-Prot McAfee	f0131152.exe	Win32:Malware-gen Trojan horse Agent.APLY W32/Trojan3.QF BackDoor-DTN trojan

Scanner-specific logs can be found in Annex C. Each scanner succeeded in detecting one or more potentially infected files. In total, sixteen unique data carved memory files were identified as potentially malicious or infected.

2.3 Volatility analysis

In order to investigate this memory image the use and output of various Volatility plugins are examined.

Store all plugin-specific output into appropriately named text files for possible future use, for preserving the analysis workflow and maintaining adequate documentation.

2.3.1 Step 1: Background information and process listings

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

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 tigger.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/Tigger_Report/tigger.vmem)
                      PAE type : PAE
                           DTB : 0x319000L
                          KDBG : 0x80544ce0
          Number of Processors : 1
     Image Type (Service Pack) : 2
                KPCR for CPU 0 : 0xffdff000
             KUSER_SHARED_DATA : 0xffdf0000
           Image date and time : 2010-08-15 19:26:50 UTC+0000
     Image local date and time : 2010-08-15 15:26:50 -0400
```

This memory image appears to be running atop a 32-bit Windows XP computer system with Service Pack 2. It is equipped with one PAE-based processor (1 core) and the memory image is 128 MiB in size (based on the memory image's size determined using *ls -l*). The memory image was captured August 15, 2010 at 15:26:50 EDT.

2.3.1.2 Pslist plugin

The next step is to identify which processes are running within the memory image in order to establish 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. The plugin uses virtual memory addressing and scans for EPROCESS lists.

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

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

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start
0x810b1660	System	4	0	61	179	------	0	
0xff25a7e0	alg.exe	216	676	7	108	0	0	2010-08-11 06:06:39
0xff3667e8	VMwareTray.exe	432	1724	1	49	0	0	2010-08-11 06:09:31
0xff374980	VMwareUser.exe	452	1724	8	204	0	0	2010-08-11 06:09:32
0x80f94588	wuauclt.exe	468	1028	4	131	0	0	2010-08-11 06:09:37
0xff2ab020	smss.exe	544	4	3	21	------	0	2010-08-11 06:06:21
0xff1ecda0	csrss.exe	608	544	10	405	0	0	2010-08-11 06:06:23
0xff1ec978	winlogon.exe	632	544	22	519	0	0	2010-08-11 06:06:23
0xff247020	services.exe	676	632	16	269	0	0	2010-08-11 06:06:24
0xff255020	lsass.exe	688	632	21	348	0	0	2010-08-11 06:06:24
0xff27d4d0	wmiprvse.exe	828	856	9	214	0	0	2010-08-15 19:26:08
0xff218230	vmacthlp.exe	844	676	1	24	0	0	2010-08-11 06:06:24
0x80ff88d8	svchost.exe	856	676	18	202	0	0	2010-08-11 06:06:24
0xff364310	wscntfy.exe	888	1028	1	27	0	0	2010-08-11 06:06:49
0xff217560	svchost.exe	936	676	9	270	0	0	2010-08-11 06:06:24
0x80fbf910	svchost.exe	1028	676	88	1395	0	0	2010-08-11 06:06:24
0xff38b5f8	TPAutoConnect.e	1084	1968	1	61	0	0	2010-08-11 06:06:52
0xff22d558	svchost.exe	1088	676	7	80	0	0	2010-08-11 06:06:25
0xff203b80	svchost.exe	1148	676	15	210	0	0	2010-08-11 06:06:26
0xff1d7da0	spoolsv.exe	1432	676	14	137	0	0	2010-08-11 06:06:26
0xff1b8b28	vmtoolsd.exe	1668	676	5	218	0	0	2010-08-11 06:06:35
0xff3865d0	explorer.exe	1724	1708	13	314	0	0	2010-08-11 06:09:29
0x80f60da0	wuauclt.exe	1732	1028	7	178	0	0	2010-08-11 06:07:44
0xff1fdc88	VMUpgradeHelper	1788	676	5	100	0	0	2010-08-11 06:06:38
0xff143b28	TPAutoConnSvc.e	1968	676	5	100	0	0	2010-08-11 06:06:39

Looking at the above listing, nothing appears out of the ordinary except for the very large number of handles in use by PID 1028 (*svchost.exe*). However, this in of itself indicates very little. Thus, no suspicious process or manner of instantiation has been identified thus far.

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 stark contrast to the *pslist* plugin. The benefit of using this plugin is that sometimes it succeeds in listing processes that cannot be found using other process listing plugins (e.g., *pslist* and *pstree*).

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

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

Offset(P)	Name	PID	PPID	PDB	Time created	Time exited
0x01214660	System	4	0	0x00319000		
0x05f027e0	alg.exe	216	676	0x06cc0240	2010-08-11 06:06:39	
0x04be97e8	VMwareTray.exe	432	1724	0x06cc02e0	2010-08-11 06:09:31	
0x04b5a980	VMwareUser.exe	452	1724	0x06cc0300	2010-08-11 06:09:32	
0x010f7588	wuauclt.exe	468	1028	0x06cc0180	2010-08-11 06:09:37	
0x04a4d2c8	cmd.exe	532	1668	0x06cc0320	2010-08-15 19:26:50	2010-08-15 19:26:50
0x05471020	smss.exe	544	4	0x06cc0020	2010-08-11 06:06:21	
0x066f0da0	csrss.exe	608	544	0x06cc0040	2010-08-11 06:06:23	
0x066f0978	winlogon.exe	632	544	0x06cc0060	2010-08-11 06:06:23	
0x06015020	services.exe	676	632	0x06cc0080	2010-08-11 06:06:24	
0x05f47020	lsass.exe	688	632	0x06cc00a0	2010-08-11 06:06:24	
0x05c9f4d0	wmiprvse.exe	828	856	0x06cc0340	2010-08-15 19:26:08	
0x06384230	vmacthlp.exe	844	676	0x06cc00c0	2010-08-11 06:06:24	
0x0115b8d8	svchost.exe	856	676	0x06cc00e0	2010-08-11 06:06:24	
0x04c2b310	wscntfy.exe	888	1028	0x06cc0200	2010-08-11 06:06:49	
0x063c5560	svchost.exe	936	676	0x06cc0100	2010-08-11 06:06:24	
0x01122910	svchost.exe	1028	676	0x06cc0120	2010-08-11 06:06:24	
0x049c15f8	TPAutoConnect.e	1084	1968	0x06cc0220	2010-08-11 06:06:52	
0x061ef558	svchost.exe	1088	676	0x06cc0140	2010-08-11 06:06:25	
0x06499b80	svchost.exe	1148	676	0x06cc0160	2010-08-11 06:06:26	
0x066f1c08	logonui.exe	1168	632	0x06cc0180	2010-08-11 06:06:26	2010-08-11 06:09:35
0x06945da0	spoolsv.exe	1432	676	0x06cc01a0	2010-08-11 06:06:26	
0x069d5b28	vmtoolsd.exe	1668	676	0x06cc01c0	2010-08-11 06:06:35	
0x04a065d0	explorer.exe	1724	1708	0x06cc0280	2010-08-11 06:09:29	
0x010c3da0	wuauclt.exe	1732	1028	0x06cc02c0	2010-08-11 06:07:44	
0x0655fc88	VMUpgradeHelper	1788	676	0x06cc01e0	2010-08-11 06:06:38	
0x0211ab28	TPAutoConnSvc.e	1968	676	0x06cc0260	2010-08-11 06:06:39	

The information presented in this table appears the same as that identified by the *pslist* plugin. Differentiating their output may help to determine if there are any differences between them. This is carried out in the following 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 ----------------         ------
1 cmd.exe 532
1 logonui.exe    1168
```

Thus, it has been determined that two processes, PIDs 532 and 1168 (*cmd.exe* and *logonui.exe*, respectively), were identified using the *psscan* plugin. Again, nothing seems noticeably out of the ordinary.

Perhaps the next plugin, *psxview*, will be of more assistance.

2.3.1.5 Psxview plugin

Volatility offers an additional plugin 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 (shown below) 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 that do not.

Sometimes processes may be listed as hidden by the *csrss* mechanism but they generally are not hidden. Therefore, any process marked as hidden by this method requires that at least another mechanism has detected it as hidden too. Consider that for Windows 7 and Vista systems, their list of internal processes is not available and that for Windows XP sometimes required memory pages may have been swapped out, all of which may affect the outcome of the *csrss* mechanism [15].

Consider the plugin's output using command "*volatility -f tigger.vmem psxview*":

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

Offset(P)	Name	PID	pslist	psscan	thrdproc	pspcdid	csrss
0x01214660	System	4	True	True	True	True	False
0x05f027e0	alg.exe	216	True	True	True	True	True
0x04be97e8	VMwareTray.exe	432	True	True	True	True	True
0x04b5a980	VMwareUser.exe	452	True	True	True	True	True
0x010f7588	wuauclt.exe	468	True	True	True	True	True
0x04a4d2c8	cmd.exe	532	False	True	False	False	False
0x05471020	smss.exe	544	True	True	True	True	False
0x066f0da0	csrss.exe	608	True	True	True	True	False
0x066f0978	winlogon.exe	632	True	True	True	True	True
0x06015020	services.exe	676	True	True	True	True	True
0x05f47020	lsass.exe	688	True	True	True	True	True
0x05c9f4d0	wmiprvse.exe	828	True	True	True	True	True
0x06384230	vmacthlp.exe	844	True	True	True	True	True
0x0115b8d8	svchost.exe	856	True	True	True	True	True
0x04c2b310	wscntfy.exe	888	True	True	True	True	True
0x063c5560	svchost.exe	936	True	True	True	True	True
0x01122910	svchost.exe	1028	True	True	True	True	True
0x049c15f8	TPAutoConnect.e	1084	True	True	True	True	True
0x061ef558	svchost.exe	1088	True	True	True	True	True
0x06499b80	svchost.exe	1148	True	True	True	True	True
0x066f1c08	logonui.exe	1168	False	True	False	False	False
0x06945da0	spoolsv.exe	1432	True	True	True	True	True
0x069d5b28	vmtoolsd.exe	1668	True	True	True	True	True
0x04a065d0	explorer.exe	1724	True	True	True	True	True
0x010c3da0	wuauclt.exe	1732	True	True	True	True	True
0x0655fc88	VMUpgradeHelper	1788	True	True	True	True	True
0x0211ab28	TPAutoConnSvc.e	1968	True	True	True	True	True

Based on the plugin's output, process PIDs 532 and 1168 (*cmd.exe* and *logonui.exe*, respectively) are hidden as only the *psscan* plugin was able to identify their presence. While this is indicative of potentially suspicious activity, it need not be considered so, at least for the time being. Consider that PID 532 was spawned by PID 1668 (*vmtoolsd.exe*) and that PID 1168 was spawned by PID 632 (*winlogin.exe*). This appears to be normal system activity.

2.3.1.6 Summary and analysis

Thus far, no specific indicators of compromise have been identified using the various system information and process listing plugins.

Additional analyses will be conducted in the subsequent steps to determine if indicators of compromise can be found 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 indicators and possibly direct evidence of infection. These plugins often provide information that the 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 (i.e., *cmd.exe*; PID 532) 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.

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 (i.e., *cmd.exe*; PID 532) or through a backdoor and this data structure keeps both the input and output buffers for commands using this plugin.

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

```
$ volatility -f tigger.vmem cmdscan

$ volatility -f tigger.vmem consoles
```

Although the two commands resulted in some output, it was entirely irrelevant. The output from both plugins was with respect to *csrss.exe* which ultimately revealed nothing pertinent to the investigation.

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 and is only effective against Windows XP and 2003.

Running command "*volatility -f tigger.vmem connscan*" resulted in the following output:

Table 7: Volatility Connscan plugin output (sorted by Offset).

Offset (P)	Local Address	Remote Address	PID
0x02214988	172.16.176.143:1034	131.107.115.254:80	1260
0x06015ab0	172.16.176.143:1037	131.107.115.254:443	1260

This communication is suspicious, as PID 1260 is no longer running on this system. However, it may be a recently terminated connection. The fact that it is using ports 80 and 443 (HTTP and HTTPS, respectively), if in fact the communication channel proves to have been malicious, is indicative of a possible command and control or exfiltration stream.

The next section will attempt to determine more information about this remote system.

2.3.2.3 Identifying the remote system

This system has been identified as belonging to Microsoft, whose valid IP address range is, according to a Whois lookup, between 131.107.0.0 – 131.107.255.255.

Thus, because there are two currently running processes for *wuauclt.exe* it is likely that the system may have just completed a Windows update. In of itself, nothing should be construed from having two instances of *wuauclt.exe* running simultaneously.

2.3.2.4 Connections plugin

The *connections* plugin is used to find evidence of ongoing communications. Sometimes this plugin is able to identify ongoing connections that *connscan* cannot. Moreover, this plugin supports both physical and virtual memory addresses.

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

2.3.2.5 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 to receive a connection:

```
$ volatility -f tigger.vmem sockets > sockets.txt
$ volatility -f tigger.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 8: Combined Volatility Sockets and Sockscan plugins output (sorted by PID).

PID	Port	Proto	Protocol	Address	Create Time
4	0	47	GRE	0.0.0.0	2010-08-11 06:08:00
4	1033	6	TCP	0.0.0.0	2010-08-11 06:08:00
4	137	17	UDP	172.16.176.143	2010-08-15 19:25:18
4	138	17	UDP	172.16.176.143	2010-08-15 19:25:18
4	139	6	TCP	172.16.176.143	2010-08-15 19:25:18
4	445	17	UDP	0.0.0.0	2010-08-11 06:06:17
4	445	6	TCP	0.0.0.0	2010-08-11 06:06:17
216	1026	6	TCP	127.0.0.1	2010-08-11 06:06:39
688	0	255	Reserved	0.0.0.0	2010-08-11 06:06:35
688	4500	17	UDP	0.0.0.0	2010-08-11 06:06:35
688	500	17	UDP	0.0.0.0	2010-08-11 06:06:35
936	136	6	TCP	0.0.0.0	2010-08-11 06:06:24
1028	1053	17	UDP	127.0.0.1	2010-08-15 19:26:50
1028	123	17	UDP	127.0.0.1	2010-08-15 19:25:18
1028	123	17	UDP	127.0.0.1	2010-08-15 19:26:50
1088	1025	17	UDP	0.0.0.0	2010-08-11 06:06:38
1148	1900	17	UDP	127.0.0.1	2010-08-15 19:25:18
1148	1900	17	UDP	127.0.0.1	2010-08-15 19:26:50
1148	1900	17	UDP	172.16.176.143	2010-08-15 19:25:18

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 1028 (*svchost.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. NTP time services can be initiated in different ways from both *svchost.exe* and *services.exe*, the latter of which is not using NTP at this time [16][17] and [18].

Those ports found open for PID 4 (*System*) are entirely normal for ports 137 – 139 and 445 [18]. However, port 1033 is not a known port for the more common network services; in fact, it is sometimes associated to malware [20]. However, this does not indicate that this port is in use for malicious purposes, only that it cannot be readily identified.

Port 1026, in use by PID 216 (*alg.exe*) is sometimes used by Microsoft DCOM services [16][19]. Process *lsass.exe* (PID 688) has two open ports of interest, 4500 and 500. The former is used for NAT-based services while the latter is for key management [17][18] and [19].

PID 936 (*svchost.exe*) is using port 136 which represents the PROFILE Naming System, a defunct network service [25][24]. This port and process are suspicious. The same can be said for port 1053 in use by PID 1028 (*svchost.exe*). This port is well known to malware [21][22].

Port 1025, in use by PID 1088 (*svchost.exe*) is commonly used for NFS and IIS. There is no indication that these services are running on this system [19]. However, this port is also commonly used for network blackjack and various malware [23]. This port is suspect because it is uncommon for Windows XP to be providing either IIS or NFS services.

Finally, port 1900, in use by PID 1148 (*svchost.exe*) is a known Windows port used for the universal Plug'n'Play discovery [16][17][19].

Thus, at a minimum, ports 136 (PID 936), 1025 (PID 1088), 1053 (PID 1028) and 1033 (PID 4) are suspicious. The next subsection will examine the relationship between these processes and their ports to look for threads of commonality between them.

2.3.2.6 Examining the relationship between suspicious ports and processes

Based on the information obtained by examining and correlating the output from the *sockets* and *sockscan* plugins certain suspicious associations were identified. These associations have been compiled into the following table:

Table 9: Suspicious ports and their corresponding PIDs and PPIDs (sorted by PID).

PID	Port	Process Name	PPID	Parent Process Name
4	1033	System	0	N/A
936	136	svchost.exe	676	services.exe
1088	1025	svchost.exe	676	services.exe
1028	1053	svchost.exe	676	services.exe

While on the surface it appears that *services.exe* (PID 676) may have been infected, possibly through DLL injection from subsequently spawned processes such as *svchost.exe* (PIDs 936, 1028 and 1088), this may in fact not be the case. Without more information, any additional line of inquiry is merely conjecture.

As for the *System* process, it is unlikely that it is infected by *services.exe* for several reasons. The first is because it was not spawned by PID 676 (*services.exe*). The second is that once the system has booted up, the System process is no longer a prime target as its primary purpose is to spawn other system processes.

2.3.2.7 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., Tigger, Syzor, 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 has a very good working knowledge of the underlying operating system. Just looking at filenames[2] and locations will not produce meaningful results, unless something really sticks out.

In this investigation, emphasis is placed on identifying indicators of compromise without the use of external documentation (i.e., malware reports), thus the investigator must studiously examine this plugin's output.

Running command "*volatility -f tigger.vmem filescan*," after extensive verification against the NSRL and upon having ruled out various development, debugging programs and other miscellaneous files found in the plugin's output, two files stood out and are listed in the following table:

Table 10: Suspicious Volatility Filescan plugin output (sorted by Offset).

Offset (P)	#Ptr	#Hnd	Access	Name
0x007774a8	1	0	R--r-d	\Device\HarddiskVolume1\DOCUME~1\ADMINI~1\LOCALS~1\Temp\329000.exe
0x01094ea0	1	0	R--r-d	\Device\HarddiskVolume1\WINDOWS\system32\drivers\tmryqyrznr2.sys

These two files are particularly suspicious. The device driver has a very non-standard name and is of unknown origin. It may well be a rootkit. The executable found in the administrative user's

[2] 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.

home directory was saved to his temporary local working storage area, a very suspicious location for saving and running executables. This program may have been the malware dropper or contained the actual malicious payload which then instantiated the aforementioned device driver.

Further analyses using other various plugins may shed additional light on these two suspicious files.

2.3.2.8 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 tigger.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 for suspicious mutexes (sorted by name).

Offset (P)	#Ptr	#Hnd	Signal	Thread	CID	Name
0x05d281f0	2	1	1	0x00000000		746bbf3569adEncrypt
0x048b21e8	2	1	1	0x00000000		ContentFilter_Perf_Library_Lock_PID_33c
0x010719b0	2	1	1	0x00000000		ContentFilter_Perf_Library_Lock_PID_684
0x02bbe428	2	1	1	0x00000000		ContentIndex_Perf_Library_Lock_PID_33c
0x010dd398	2	1	1	0x00000000		ContentIndex_Perf_Library_Lock_PID_684
0x05f01c68	2	1	1	0x00000000		ExplorerIsShellMutex
0x04b5c668	2	1	1	0x00000000		HGFSMUTEX00000000000242b4
0x060a1468	2	1	1	0x00000000		ISAPISearch_Perf_Library_Lock_PID_33c
0x010d4030	2	1	1	0x00000000		ISAPISearch_Perf_Library_Lock_PID_684
0x05c18f08	3	2	1	0x00000000		MidiMapper_modLongMessage_RefCnt
0x06234b88	2	1	1	0x00000000		MSDTC_Perf_Library_Lock_PID_33c
0x010bc140	2	1	1	0x00000000		MSDTC_Perf_Library_Lock_PID_684
0x05d6bac0	2	1	1	0x00000000		PerfDisk_Perf_Library_Lock_PID_33c
0x010bc550	2	1	1	0x00000000		PerfDisk_Perf_Library_Lock_PID_684
0x05d6ba70	2	1	1	0x00000000		PerfNet_Perf_Library_Lock_PID_33c
0x010bc750	2	1	1	0x00000000		PerfNet_Perf_Library_Lock_PID_684
0x05d6ba20	2	1	1	0x00000000		PerfOS_Perf_Library_Lock_PID_33c
0x01066480	2	1	1	0x00000000		PerfOS_Perf_Library_Lock_PID_684
0x05def9f0	2	1	1	0x00000000		PerfProc_Perf_Library_Lock_PID_33c
0x01066bd0	2	1	1	0x00000000		PerfProc_Perf_Library_Lock_PID_684
0x05def9a0	2	1	1	0x00000000		PSched_Perf_Library_Lock_PID_33c
0x024c9ef8	2	1	1	0x00000000		PSched_Perf_Library_Lock_PID_684

Offset (P)	#Ptr	#Hnd	Signal	Thread	CID	Name
0x05def950	2	1	1	0x00000000		RemoteAccess_Perf_Library_Lock_PID_33c
0x010676d8	2	1	1	0x00000000		RemoteAccess_Perf_Library_Lock_PID_684
0x04a49d50	2	1	1	0x00000000		RSVP_Perf_Library_Lock_PID_33c
0x067358a8	2	1	1	0x00000000		RSVP_Perf_Library_Lock_PID_684
0x05f45148	7	6	1	0x00000000		SHIMLIB_LOG_MUTEX
0x04a49d00	2	1	1	0x00000000		Spooler_Perf_Library_Lock_PID_33c
0x01070380	2	1	1	0x00000000		Spooler_Perf_Library_Lock_PID_684
0x04a49cb0	2	1	1	0x00000000		TapiSrv_Perf_Library_Lock_PID_33c
0x010bcc38	2	1	1	0x00000000		TapiSrv_Perf_Library_Lock_PID_684
0x04861820	2	1	1	0x00000000		Tcpip_Perf_Library_Lock_PID_33c
0x010bbf10	2	1	1	0x00000000		Tcpip_Perf_Library_Lock_PID_684
0x048617d0	2	1	1	0x00000000		TermService_Perf_Library_Lock_PID_33c
0x010bed60	2	1	1	0x00000000		TermService_Perf_Library_Lock_PID_684
0x010d61e8	2	1	1	0x00000000		userenv: User Registry policy mutex
0x06016b88	2	1	1	0x00000000		VMwareGuestCopyPasteMutex
0x06b1a460	2	1	1	0x00000000		VMwareGuestDnDDataMutex
0x0644eeb0	5	4	1	0x00000000		WindowsUpdateTracingMutex
0x064951d0	2	1	1	0x00000000		WininetStartupMutex
0x04861780	2	1	1	0x00000000		WmiApRpl_Perf_Library_Lock_PID_33c
0x01069fa8	2	1	1	0x00000000		WmiApRpl_Perf_Library_Lock_PID_684
0x0105acf0	2	1	0	0xff3ba880	888:912	wscntfy_mtx
0x011211a8	3	2	1	0x00000000		ZonesCacheCounterMutex
0x01093c90	3	2	1	0x00000000		ZonesLockedCacheCounterMutex

It is very possible that most of the identified mutexes listed above are erroneous. However, without a very extensive knowledge of Windows-based reverse engineering or available whitelists and blacklists (for which none could be found), it is not easy to differentiate between false positives and genuinely suspicious mutexes. Instead, web searches were used to identify those that stood out in order to determine their underlying nature. Although the above output was thoroughly analysed and revised several times, it is possible that other suspicious mutexes went unrecognized.

Taking into account all six malware memory infections examined thus far in this series of reports [1][2][3][4] including this current report, this is the longest list of suspicious mutexes thus far generated.

Three items stand out when looking at the above list. The first is that all but one of the suspected mutexes has a Thread identification of *0x00000000*. Moreover, the majority of the mutexes end with a suffix of *PID_684* or *PID_33c*. The third item is that those mutexes ending in either suffix is preceded or followed by the other suffix-ending mutex, thereby suggesting a strong inference that some unknown process or thread is behind these mutexes.

In-depth web searches suggest that each of these mutexes is associated, directly or indirectly, to malware. Some mutexes can be linked to specific malware while others are simply known markers for various malicious software components. Nevertheless, when comparing these mutexes to all previous reports [1][2][3][4] and their respective mutexes, a picture emerges. It does not appear that coincidence is at play here – these mutexes exist for a reason and that reason appears to be to search for various system conditions – perhaps looking for explicit markers in memory. However, these author-proposed theories are only conjecture.

Finally, mutexes *VMwareGuestCopyPasteMutex* and *VMwareGuestDnDDataMutex* are indicative of known mutexes used by malware to determine if the operating system is VMware-based.

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

2.3.2.9 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 tigger.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
0xff158460	452	0x94	0x1f0001	Mutant	VMwareGuestDnDDataMutex
0xff246b88	452	0x9c	0x1f0001	Mutant	VMwareGuestCopyPasteMutex
0xff20eeb0	468	0xfc	0x1f0001	Mutant	WindowsUpdateTracingMutex
0x80f731e8	632	0x4c	0x1f0001	Mutant	userenv: User Registry policy mutex
0xff284f08	632	0x828	0x1f0001	Mutant	MidiMapper_modLongMessage_RefCnt
0xff257148	676	0x24	0x1f0001	Mutant	SHIMLIB_LOG_MUTEX
0xff257148	688	0x24	0x1f0001	Mutant	SHIMLIB_LOG_MUTEX
0xff13a428	828	0x2cc	0x1f0001	Mutant	ContentIndex_Perf_Library_Lock_PID_33c
0xff228b88	828	0x2dc	0x1f0001	Mutant	MSDTC_Perf_Library_Lock_PID_33c
0xff23b468	828	0x2d4	0x1f0001	Mutant	ISAPISearch_Perf_Library_Lock_PID_33c
0xff26d950	828	0x30c	0x1f0001	Mutant	RemoteAccess_Perf_Library_Lock_PID_33c
0xff26d9a0	828	0x304	0x1f0001	Mutant	PSched_Perf_Library_Lock_PID_33c

Offset (V)	PID	Handle	Access	Type	Details
0xff26d9f0	828	0x2fc	0x1f0001	Mutant	PerfProc_Perf_Library_Lock_PID_33c
0xff271a20	828	0x2f4	0x1f0001	Mutant	PerfOS_Perf_Library_Lock_PID_33c
0xff271a70	828	0x2ec	0x1f0001	Mutant	PerfNet_Perf_Library_Lock_PID_33c
0xff271ac0	828	0x2e4	0x1f0001	Mutant	PerfDisk_Perf_Library_Lock_PID_33c
0xff384cb0	828	0x324	0x1f0001	Mutant	TapiSrv_Perf_Library_Lock_PID_33c
0xff384d00	828	0x31c	0x1f0001	Mutant	Spooler_Perf_Library_Lock_PID_33c
0xff384d50	828	0x314	0x1f0001	Mutant	RSVP_Perf_Library_Lock_PID_33c
0xff3991e8	828	0x2c4	0x1f0001	Mutant	ContentFilter_Perf_Library_Lock_PID_33c
0xff3a9780	828	0x33c	0x1f0001	Mutant	WmiApRpl_Perf_Library_Lock_PID_33c
0xff3a97d0	828	0x334	0x1f0001	Mutant	TermService_Perf_Library_Lock_PID_33c
0xff3a9820	828	0x32c	0x1f0001	Mutant	Tcpip_Perf_Library_Lock_PID_33c
0xff257148	856	0x24	0x1f0001	Mutant	SHIMLIB_LOG_MUTEX
0xff2741f0	856	0x218	0x1f0001	Mutant	746bbf3569adEncrypt
0x80ef7cf0	888	0x38	0x1f0001	Mutant	wscntfy_mtx
0x80f30c90	1028	0xebc	0x1f0001	Mutant	ZonesLockedCacheCounterMutex
0x80fbe1a8	1028	0xe94	0x1f0001	Mutant	ZonesCacheCounterMutex
0xff20eeb0	1028	0x88c	0x1f0001	Mutant	WindowsUpdateTracingMutex
0xff257148	1028	0x24	0x1f0001	Mutant	SHIMLIB_LOG_MUTEX
0x80f30c90	1148	0x190	0x1f0001	Mutant	ZonesLockedCacheCounterMutex
0x80fbe1a8	1148	0x188	0x1f0001	Mutant	ZonesCacheCounterMutex
0xff2071d0	1148	0x1a8	0x1f0001	Mutant	WininetStartupMutex
0xff257148	1432	0x24	0x1f0001	Mutant	SHIMLIB_LOG_MUTEX
0x80f03480	1668	0x2f0	0x1f0001	Mutant	PerfOS_Perf_Library_Lock_PID_684
0x80f03bd0	1668	0x2f8	0x1f0001	Mutant	PerfProc_Perf_Library_Lock_PID_684
0x80f046d8	1668	0x308	0x1f0001	Mutant	RemoteAccess_Perf_Library_Lock_PID_684
0x80f06fa8	1668	0x338	0x1f0001	Mutant	WmiApRpl_Perf_Library_Lock_PID_684
0x80f0d380	1668	0x318	0x1f0001	Mutant	Spooler_Perf_Library_Lock_PID_684
0x80f0e9b0	1668	0x2c0	0x1f0001	Mutant	ContentFilter_Perf_Library_Lock_PID_684
0x80f58f10	1668	0x328	0x1f0001	Mutant	Tcpip_Perf_Library_Lock_PID_684
0x80f59140	1668	0x2d8	0x1f0001	Mutant	MSDTC_Perf_Library_Lock_PID_684
0x80f59550	1668	0x2e0	0x1f0001	Mutant	PerfDisk_Perf_Library_Lock_PID_684
0x80f59750	1668	0x2e8	0x1f0001	Mutant	PerfNet_Perf_Library_Lock_PID_684

Offset (V)	PID	Handle	Access	Type	Details
0x80f59c38	1668	0x320	0x1f0001	Mutant	TapiSrv_Perf_Library_Lock_PID_684
0x80f5bd60	1668	0x330	0x1f0001	Mutant	TermService_Perf_Library_Lock_PID_684
0x80f71030	1668	0x2d0	0x1f0001	Mutant	ISAPISearch_Perf_Library_Lock_PID_684
0x80f7a398	1668	0x2c8	0x1f0001	Mutant	ContentIndex_Perf_Library_Lock_PID_684
0xff13cef8	1668	0x300	0x1f0001	Mutant	PSched_Perf_Library_Lock_PID_684
0xff1e78a8	1668	0x310	0x1f0001	Mutant	RSVP_Perf_Library_Lock_PID_684
0xff25bc68	1724	0xa8	0x1f0001	Mutant	ExplorerIsShellMutex
0xff284f08	1724	0x3d0	0x1f0001	Mutant	MidiMapper_modLongMessage_RefCnt
0xff372668	1724	0x1d8	0x1f0001	Mutant	HGFSMUTEX00000000000242b4
0xff20eeb0	1732	0x104	0x1f0001	Mutant	WindowsUpdateTracingMutex
0xff20eeb0	1732	0x190	0x1f0001	Mutant	WindowsUpdateTracingMutex
0xff257148	1732	0x2c	0x1f0001	Mutant	SHIMLIB_LOG_MUTEX

As mentioned in the preceding section, in-depth web searches were conducted which suggested that the above-listed mutexes are associated to malware, some to known and others unknown malware. It is possible that other suspicious mutexes and handles were present but were not flagged due to the lack of appropriate context in which to evaluate them. Again, without in-depth knowledge of Windows reverse engineering it is not possible to readily determine which handles are suspicious and which are not, especially since no reliable whitelist or blacklist was readily available.

Examining the table, many of the various mutexes found using the *mutantscan* plugin were found with *handles* plugin. This is not to suggest that all these mutexes are necessarily suspicious but the fact that this is the only investigation conducted by the author against the variously infected memory images [1][2][3][4] thus far where this has occurred is unlikely to be coincidental. Interestingly, when comparing this analysis' *mutantscan* and *handles* plugins' results against those from the previously conducted reports [1][2][3][4], it becomes clear that this memory image is unique in this sense. Moreover, there are many uncommon mutexes at work in this memory image. This information should push the investigator to further scrutinize this memory image using additional Volatility plugins.

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. However, neither plugin has the ability to detect unloaded or unlinked drivers. The results obtained in this section will be corroborated using the *modscan* plugin in the following subsection.

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

```
$ volatility -f tigger.vmem driverscan

$ volatility -f tigger.vmem driverirp
```

The output from these commands was pruned for pertinence.

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

Table 13: Volatility Driverscan plugin output for suspicious driver.

Offset (P)	#Ptr	#Hnd	Start	Size (in hex)	Service Key	Name	Driver Name
0x0108ef38	2	0	0x00000000	0x0	tmryqyrznr2.sys	tmryq....sys	N/A

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

```
DriverName: tmryqyrznr2.sys
DriverStart: 0x0
DriverSize: 0x0
DriverStartIo: 0x0
   0 IRP_MJ_CREATE                        0x00000000 Unknown
   1 IRP_MJ_CREATE_NAMED_PIPE             0x00000000 Unknown
   2 IRP_MJ_CLOSE                         0x00000000 Unknown
   3 IRP_MJ_READ                          0x00000000 Unknown
   4 IRP_MJ_WRITE                         0x00000000 Unknown
   5 IRP_MJ_QUERY_INFORMATION             0x00000000 Unknown
   6 IRP_MJ_SET_INFORMATION               0x00000000 Unknown
   7 IRP_MJ_QUERY_EA                      0x00000000 Unknown
   8 IRP_MJ_SET_EA                        0x00000000 Unknown
   9 IRP_MJ_FLUSH_BUFFERS                 0x00000000 Unknown
  10 IRP_MJ_QUERY_VOLUME_INFORMATION      0x00000000 Unknown
  11 IRP_MJ_SET_VOLUME_INFORMATION        0x00000000 Unknown
  12 IRP_MJ_DIRECTORY_CONTROL             0x00000000 Unknown
  13 IRP_MJ_FILE_SYSTEM_CONTROL           0x00000000 Unknown
  14 IRP_MJ_DEVICE_CONTROL                0x00000000 Unknown
  15 IRP_MJ_INTERNAL_DEVICE_CONTROL       0x00000000 Unknown
  16 IRP_MJ_SHUTDOWN                      0x00000000 Unknown
  17 IRP_MJ_LOCK_CONTROL                  0x00000000 Unknown
  18 IRP_MJ_CLEANUP                       0x00000000 Unknown
  19 IRP_MJ_CREATE_MAILSLOT               0x00000000 Unknown
  20 IRP_MJ_QUERY_SECURITY                0x00000000 Unknown
  21 IRP_MJ_SET_SECURITY                  0x00000000 Unknown
  22 IRP_MJ_POWER                         0x00000000 Unknown
  23 IRP_MJ_SYSTEM_CONTROL                0x00000000 Unknown
  24 IRP_MJ_DEVICE_CHANGE                 0x00000000 Unknown
  25 IRP_MJ_QUERY_QUOTA                   0x00000000 Unknown
  26 IRP_MJ_SET_QUOTA                     0x00000000 Unknown
  27 IRP_MJ_PNP                           0x00000000 Unknown
```

Much is suspicious about this device driver. Its size is not available, as per the *driverscan* plugin, an anomaly at best and often the distinctive hallmark of a rootkit device driver. As based on the information obtained from the *driverirp* plugin, this device driver, unlike all the other device drivers encountered in the various infected memory analyses conducted thus far by the author [1][2][3][4], provides absolutely no information about its capabilities via IRP_MJ. Therefore, it can be suggested that this device driver has likely manipulated its state in memory. Little doubt remains that this driver is not a rootkit.

2.3.2.11 Modscan plugin

The *modscan* plugin scans a memory image for kernel structure LDR_DATA_TABLE_ENTRY which enables the plugin to not only list all device drivers that the *driverscan* plugin detects but it can also identify drivers that have been unloaded or unlinked by rootkits. The plugin uses physical memory address offsets.

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

Table 14: Volatility Modscan plugin output for suspicious driver.

Offset (P)	Name	Base Address	Size	File
0x04b59b08	N/A	0x00000000	N/A	N/A

While examining the plugin's output, it was determined that it succeeded in identifying 122 unique device drivers in memory while the *driverscan* and *driverirp* plugins each could only identify 99 drivers. However, while going through the output generated from the *modscan* plugin, a careful analysis revealed that all device drivers were readily attributable except for the above nameless driver.

The above driver is very likely a part of the infection sought after and is probably a rootkit.

2.3.2.12 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 numerous exceptions exist. For instance, filter drivers are not commonly registered as services that 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 tigger.vmem svcscan*" did not produce any information concerning the previously identified suspicious driver, *tmryqyrznr2.sys* or any other unexpected system service.

2.3.2.13 Ldrmodules plugin

The *ldrmodules* plugin scans a memory image for signs of unlinked files (such as but not limited to DLLs) in memory. These may be indicative of suspicious or malicious files lurking in memory. Although no suspicious DLLs were found, 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 tigger.vmem ldrmodules | grep False*" was issued which generated the following output:

Table 15: Volatility ldrmodules plugin output for hidden modules (sorted by PID).

PID	Process	Base	InLoad	InInit	InMem	MappedPath
216	alg.exe	0x01000000	True	False	True	\WINDOWS\system32\alg.exe
432	VMwareTray.exe	0x00400000	True	False	True	\Program Files\VMware\VMware Tools\VMwareTray.exe
452	VMwareUser.exe	0x00400000	True	False	True	\Program Files\VMware\VMware Tools\VMwareUser.exe
468	wuauclt.exe	0x00400000	True	False	True	\WINDOWS\system32\wuauclt.exe
632	winlogon.exe	0x01000000	True	False	True	\WINDOWS\system32\winlogon.exe
676	services.exe	0x01000000	True	False	True	\WINDOWS\system32\services.exe
688	lsass.exe	0x01000000	True	False	True	\WINDOWS\system32\lsass.exe
828	wmiprvse.exe	0x01000000	True	False	True	\WINDOWS\system32\wbem\wmiprvse.exe
856	svchost.exe	0x01000000	True	False	True	\WINDOWS\system32\svchost.exe
936	svchost.exe	0x01000000	True	False	True	\WINDOWS\system32\svchost.exe
1028	svchost.exe	0x01000000	True	False	True	\WINDOWS\system32\svchost.exe
1084	TPAutoConnect.e	0x00400000	True	False	True	\Program Files\VMware\VMware Tools\TPAutoConnect.exe
1088	svchost.exe	0x01000000	True	False	True	\WINDOWS\system32\svchost.exe
1148	svchost.exe	0x01000000	True	False	True	\WINDOWS\system32\svchost.exe
1432	spoolsv.exe	0x01000000	True	False	True	\WINDOWS\system32\spoolsv.exe
1668	vmtoolsd.exe	0x00400000	True	False	True	\Program Files\VMware\VMware Tools\vmtoolsd.exe
1724	explorer.exe	0x01000000	True	False	True	\WINDOWS\explorer.exe
1732	wuauclt.exe	0x00400000	True	False	True	\WINDOWS\system32\wuauclt.exe

Looking at the above data, none of the detected unlinked modules, all Windows executables, are suspicious. Since no specific process has yet been identified as suspicious, there is insufficient context in which to judge the results of the *ldrmodules* plugin.

2.3.2.14 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 given memory image. Running command "*volatility -f tigger.vmem dlllist*" identified, in total, 1072 DLLs and other modules loaded into memory.

Based on this plugin's output, nothing out of the ordinary was found.

2.3.2.15 Threads and thrdscan plugins

Two Volatility plugins are used in this subsection, specifically the *threads* and *thrdscan* plugins. These plugins are used to provide additional process and thread based information in order to shed light on possibly hidden processes/threads.

There is little doubt that a rootkit is involved, as the evidence has already shown. However, since no process thus far has been implicated, nor have any associated DLLs, hidden or otherwise, (or other files such as executables) been identified, it is plausible that a hidden thread may currently be at work.

The *threads* plugin searches for _ETHREADS and _KTHREADS data structures while the *thrdscan* plugin searches only for _ETHREADS data structures. Importantly, the output from each plugin differs significantly. Moreover, the former plugin uses virtual memory addressing whereas the latter uses physical memory addressing.

The *threads* and *thrdscan* plugins were run against the memory image and the output pruned for relevance to the investigation. The following two commands were issued:

```
$ volatility -f tigger.vmem threads
$ volatility -f tigger.vmem thrdscan | grep OrphanThread
```

Output from the *threads* plugin revealed no information of immediate pertinence. However, the *thrdscan* plugin revealed that four threads were identified as orphans as seen in the following table:

Table 16: Volatility Thrdscan plugin for orphaned threads (sorted by TID).

PID	TID	Created	Exited	State	Start Address
4	600	2010-08-11 06:09:35	2010-08-11 06:09:35	Terminated	0xf2fe2150 UNKNOWN
4	1648	2010-08-15 19:26:13	1970-01-01 00:00:00	Waiting: DelayExecution	0xf2edd150 UNKNOWN
4	1720	2010-08-15 19:26:13	1970-01-01 00:00:00	Waiting: DelayExecution	0xf2edc54e UNKNOWN
4	1992	2010-08-15 19:26:13	1970-01-01 00:00:00	Waiting: DelayExecution	0xf2edba46 UNKNOWN

An orphaned thread is a thread whose parent process has since terminated. Sometimes the sole purpose of a process is to spawn one or more threads and then terminate, whether by intentional design or from known/unknown program errors/bugs. While this type of behaviour is not in of itself illegitimate, it is sometimes the hallmark of malicious software trying to remain covert.

In this analysis, four orphaned threads were identified, even though TID 600 has since terminated, leaving only three active orphaned threads, TIDs 1648, 1720 and 1992. Anyone of these threads could be responsible for the infection. There simply is insufficient information to determine which specific thread to focus on. Thus, all three active threads are highly suspect because there should not be three actively running orphaned threads, all executed at the same time, 2010-08-15 19:26:13.

These are kernel-mode threads, as per the fact their PID is 4, indicating the System process, which executes and schedules threads for the kernel and other kernel related objects including devices. Moreover, the three running threads are waiting for some event or signal to bring them back from their waiting state. Finally, because these three orphaned threads are system threads they were likely initiated by a rootkit (or device driver).

2.3.2.16 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 associated to, specifically ports 136 (PID 936), 1025 (PID 1088), 1053 (PID 1028) and 1033 (PID 4), three of which (PIDs 936, 1028 and 1088, respectively) were spawned by PID 676 (*services.exe*).

Furthermore, highly suspicious files were detected in this memory image, specifically *WINDOWS\system32\drivers\tmryqyrznr2.sys* and *DOCUME~1\ADMINI~1\LOCALS~1\Temp\329000.exe*. Moreover, many suspicious mutexes and handles were identified which have not been seen in the previous five analyses [1][2][3][4].

The *driverscan*, *driverirp* and *modscan* plugins have confirmed that file *tmryqyrznr2.sys* is very likely a rootkit, displaying highly abnormal and suspicious characteristics and behaviour.

Finally, three very suspicious still active threads were identified on this system all with the same creation time. Bringing the pieces of evidence together, it appears that not only is a rootkit at work but that several covert threads are active and lurking in memory, but to what purpose? This is not yet known.

2.3.3 Step 3: Detection and analysis of dumped suspicious driver, threads and other items of interest

Sufficient evidence has been established thus far indicating that a highly suspicious driver has been identified. Moreover, suspicious network ports were found listening for non-standard connections, some of which are used commonly by malware. Finally, three orphaned system-level threads were found running in memory.

This phase of the analysis will attempt to dump what is attainable from the memory image, analyse it and if possible corroborate it to the evidence thus far obtained.

2.3.3.1 Create data directories

In order to proceed with the dumping of various items from the memory image, certain data storage directories must first be created. The following directories must be created: *malfind*, *dlldump*, *moddump*, *memdump*, *procexedump* and *dump*. This is done using the following commands:

```
$ mkdir malfind

$ mkdir dlldump

$ mkdir moddump

$ mkdir memdump

$ mkdir procexedump

$ mkdir dumpfiles
```

2.3.3.2 Malfind plugin

The evidence and potential indicators of compromise thus far identified indicate that potentially malicious code may be at work in the form of a rootkit. However, it is likely that DLL injection may also be in use and therefore the *malfind* plugin may be helpful in identifying which process(es) has been compromised.

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

2.3.3.2.1 Running the plugin

Because important evidence has been uncovered indicating infection, it makes sense to conduct an at large analysis of this memory image using this plugin. Thus, the following command was run against the entire memory image:

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

This command succeeded in dumping seven sample files from the memory image. However, looking at only the textual output generated by the plugin, as listed in Annex D, none of the files was indicative of code injection. Nevertheless, subsequent analyses will confirm or rule them out from direct involvement concerning this infection.

2.3.3.2.2 AV scanning and file type determination

Of these seven samples, analysed using the Linux *file* command to determine their file type, they were all identified as arbitrary data files.

All seven samples were scanned using the six aforementioned scanners. Of these samples, none was identified as infected or possibly malicious.

Strings-based identification did not reveal any fundamental features of the dumped *malfind*-based files.

2.3.3.2.3 SHA1 and fuzzy hashes

All seven dumped files were hashed using the *sha1sum* command to identify duplicates; none were identified. These hashes were then compared against the NSRL 2.41 hashset; again, no matches were established. The hashes from the *malfind*-dumped files were then compared against the SHA1 hashes obtained for the data carved memory files revealing no matches.

The memory samples were then fuzzy hashed against one another to determine if there were any similarities between them; however, no similarities were identified. The dumped files were then compared against the data carved memory files; again, no matches were established.

2.3.3.2.4 Summary

The *malfind* plugin did not succeed in identifying or dumping any DLL injected process. This does not preclude DLL injection from having occurred or from such code residing within this memory image, only that it was not possible to detect it using this plugin.

2.3.3.3 Dlldump 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.

2.3.3.3.1 Running the plugin

Using the *dlldump* plugin, it will be possible to dump all detectable memory-resident DLLs to disk for subsequent analysis. However, DLL dumping was carried out against the entire memory image so address offsets are not needed.

The following command was issued to dump detectable DLLs from the memory image:

```
$ volatility -f tigger.vmem --dump-dir=dlldump
```

In all, 427 files were dumped from the memory image. However, while examining the textual output of this plugin, it became apparent that many DLLs had been paged out. In all 645 DLLs were paged out, either in part or in full, while 395 DLLs were fully identified as residing within the memory image. In addition, 32 DLLs were identified as not containing the correct DOS signature, not a problem in of itself.

2.3.3.3.2 AV scanning and file type determination

File analysis has revealed that of the 427 DLLs dumped by the *dlldump* plugin, 32 were identified as empty files. File size analysis has confirmed that these files in fact occupy zero bytes of disk space. In addition, 18 other files were identified as executables while the remaining 377 dumped files were recognized as DLLs.

All 427 dumped samples were scanned using the aforementioned scanners. Of all the dumped memory samples, only four were identified as potentially malicious or infected by two specific scanners. Specifics are shown in the following table:

Table 17: Multi-scanner results for Dlldump samples (sorted by scanner).

Scanner	Dlldump	Infection Identification
Avast	module.1724.4a065d0.1000000.dll	Win32:Malware-gen
Comodo	module.828.5c9f4d0.20000000.dll module.452.4b5a980.20000000.dll module.468.10f7588.20000000.dll	Malware Malware Malware

Of the above-listed DLLs identified by the various scanners, no matches were established. Thus, due to the nature of these results, it was necessary to manually inspect, using *strings* and hexadecimal analysis to determine whether these DLLs were in fact suspicious or malicious.

However, further analysis revealed that none of the identified DLLs were either infected or malicious. Therefore, they no longer warranted consideration.

2.3.3.3.3 SHA1 and fuzzy hashes

SHA1 hashing of the dumped files identified 387 unique files obtained using the *dlldump* plugin. The remaining 40 DLLs were various duplicates, for a total of 427 dumped samples. These duplicates can be found in Annex E.1.

No SHA1-based matches were identified between these dumped memory samples and those obtained via data carving. When comparing the SHA1 hashes of the DLL samples against the NSRL, only one hash match was identified, as shown below:

DA39A3EE5E6B4B0D3255BFEF95601890AFD80709

However, this particular hash matched 339,184 NSRL 2.41 entries; far too many to ever incorporate into this report. Thus, with respect to this hash, the vast majority are likely false positives.

Fuzzy hash matching was then carried out between the dumped DLLs and the data carved memory files to identify file similarities. Although no 100% matches were obtained, 114 partial matches were found. A full listing of these matches is available in Annex E.2.

Fuzzy hash matching carried out between the dumped DLLs to identify similarities between them found 500 partial and full matches. However, applying a similarity-matching threshold of 75% reduced this number to 222. However, the reader is free to choose a different threshold. Of these 222 matches, 27 were identified as 100% matches. The matches found above this threshold are listed in Annex E.3.

Similarity matching conducted between the *malfind*-dumped and *dlldump*-obtained memory samples did not identify any matches.

2.3.3.3.4 Summary

This plugin, while powerful and often of immense use to investigations, was not able to provide any tangible results for this particular examination. Moreover, those DLL modules identified as possibly infected or malicious by the aforementioned scanners all turned out to be false positives.

2.3.3.4 Moddump plugin

Volatility's *moddump* plugin was specifically designed to dump drivers from memory to disk. The plugin can be used to dump individual or all available drivers which have not been paged out or which have a valid base address to disk.

2.3.3.4.1 Running the plugin

It is very likely that a rootkit is at work in this memory image, as the evidence has already demonstrated. Although it is very unlikely that rootkit *tmryqyrznr2.sys* (see Sections 2.3.2.7, 2.3.2.10 and 2.3.2.11) can be dumped from the memory image because no base address was identified for it, other drivers might be present in the memory image that may warrant further inspection. Since it is not known if other drivers in this memory image are potentially malicious, all drivers will be dumped using this plugin, with the following command:

```
$ volatility -f tigger.vmem moddump --dump-dir=moddump
```

Upon running the plugin and examining its textual output, it was determined that four specific drivers could not be dumped as they had been paged out. Moreover, rootkit *tmryqyrznr2.sys* was not found in the list of dumped drivers, therefore it has not been acquired using this plugin. Nevertheless, subsequent analysis of the dumped drivers may reveal additional details about this infection.

2.3.3.4.2 AV scanning and file type determination

In all, 115 device drivers were dumped to disk. Using the *file* command, 13 of these dumped files were detected as Windows DLLs while 102 were identified as device drivers.

When using the aforementioned AV scanners against the dumped driver, only F-Prot identified one specific driver as possibly infected or malicious, specifically:

driver.804d7000.sys

However, this turned out to be a false positive. Moreover, this driver was identified as *NTOSKRNL.EXE*.

Malware scanning failed to reveal any additional details about the dumped drivers, indicating that if any malicious driver (i.e., *tmryqyrznr2.sys*) was lurking within this memory image then it was not dumped to disk using this plugin.

2.3.3.4.3 SHA1 and fuzzy hashes

The device drivers were SHA1 hashed to determine if there were any duplicates among them, none were identified. These hashes were then compared against the NSRL for which no matches were found. Finally, the hashes were compared against those obtained for the data carving memory files that identified 12 unique matches, as shown in the following table:

Table 18: SHA1 hash matches established between moddump samples and carved memory files.

SHA1 Hash	Moddump	Carved Memory File
002e9989ff8971fa63058e3610f4905b65c2dd4d	driver.fbf3a000.sys	f0220920_diskdump.sys
5d0c10a8b9925d7172758d26a7ce320351fe780f	driver.fc5cb000.sys	f0132272.exe
cd0817e5033b034b8b0e5adef83a1f0e33b4fd9b	driver.fc5fb000.sys	f0241960_dxgthk.sys
5a1234a9e93f23a0530d65a186981852d270c926	driver.fc793000.sys	f0260112_bootvid.dll
61699bbe25fc0ae20e9cbf0b112a2f264ee70e1d	driver.fc8ab000.sys	f0260784_intelide.sys
96d18a44a848dd16a2f1a559cdb75499045f8dd7	driver.fc99f000.sys	f0134816.exe
12182d877cb8af59027f7835b30574bf1884a8b9	driver.fc9a7000.sys	f0163072_videosim.sys
d75ae8f1d0504a3ed6e60c727966df37b081d0e4	driver.fc9b1000.sys	f0159576_usbd.sys
28f14019b53517123bd7edbae0ad8c5382562d14	driver.fc9b3000.sys	f0049680_null.sys
9187b7920f0dd036840d8782ef622af57e30cbef	driver.fcac9000.sys	f0131624_audstub.sys
bf7a5844be252e0bb86deab55ae47087824722c4	driver.fcb25000.sys	f0134352_tdi.sys
6180f339c1a8b0bcf41a590e7fc9a7d53be855d0	driver.fcbef000.sys	f0163000_beep.sys

Fuzzy hash matching was then carried out between the *moddump*-dumped memory samples and the data carved memory files to identify file similarities. In total, 101 matches were established with 22 identified as 100% matches. However, the SHA1 hashes listed in the above table tell a different story concerning these 100% matches. Thus, some of these matches are byte identical while others are not. Nevertheless, due to the manner in which similarity-matching works, even slight variations could still allow for 100% matching to be established. All non-duplicate fuzzy hash matches are listed in Annex E.4.

Additional similarity matching, conducted between the various *moddump*-dumped samples and the dumped DLLs has revealed two 35% matches between device driver *driver.fc67b000.sys* and dumped DLLs *module.1724.4a065d0.754d0000.dll* and *module.452.4b5a980.754d0000.dll*. Finally, similarity matching against the *malfind*-dumped samples has established no identifiable matches.

2.3.3.4.4 Summary

The *moddump* plugin is a highly capable plugin that can be used to dump device drivers from a Windows memory image for further analysis in the event a possible rootkit is suspected. Although it often works without issue, it requires that all drivers have a valid base address in order to be dumped. However, since the suspected rootkit in question has no valid base address, it cannot be dumped using this plugin.

2.3.3.5 Memdump plugin

The *memdump* plugin is used to dump the addressable memory space of a given process. If memory address offsets are to be specified then they must be physical memory address offsets. The plugin dumps all data segments associated with a specified process to a single destination file and may include process-associated malware including process-injected code and malicious DLLs and other files.

2.3.3.5.1 Running the plugin

There is little doubt that malware is contained within this memory image. However, to date none of the plugins used have succeeded in dumping any associated malware.

Since it is not known which process is infected, all processes will be dumped using the following command:

```
$ volatility -f tigger.vmem memdump --dump-dir=memdump
```

Upon running the plugin and examining its textual output, no errors or warnings were identified while running the completion. In all, 25 processes were dumped to disk.

2.3.3.5.2 AV scanning and file type determination

Scanning of the dumped processes and their respective memory spaces using the aforementioned scanners revealed no indications of infection.

Using the *file* command, the 25 memory dump samples were identified as arbitrary data files.

2.3.3.5.3 SHA1 and fuzzy hashes

SHA1 hashing has revealed that, as expected, each process memory dump is unique. Moreover, no matches with the NSRL dataset were established nor were identical files identified from the carved memory data files.

Fuzzy hash matching has identified in all 298 partial matches, however, no 100% matches were established. Similarities ranged from 94% to 60%, thereby indicating that the majority of dumped processes shared similarities between one another, specifically shared DLLs and their common code base. These comparisons can be found in Annex E.5.

Additional similarity matching, conducted between the process memory dumps and those samples dumped thus far using the *moddump*, *dlldump* and *malfind* plugins found no similarities.

2.3.3.5.4 Summary

Although the *memdump* plugin worked as expected, analysis of its dumped files has not revealed any indication of infection.

2.3.3.6 Procexedump plugin

The *procexedump* plugin is used to dump only a process' executable code. It does not dump process slack space nor will it dump process dependencies including DLLs (if this is required, the *memdump* plugin should be used). If memory address offsets are to be specified then they must be physical memory address offsets.

2.3.3.6.1 Running the plugin

The objective in using this plugin is to dump all running processes' executable code and then scan them in the hopes of identifying malicious code that may have been patched into its code space through various forms of injection.

The command issued to run this plugin was:

```
$ volatility -f tigger.vmem procexedump --dump-
dir=procexedump
```

The plugin succeeded in dumping 19 processes to disk, conducted without any warnings of errors.

2.3.3.6.2 AV scanning and file type determination

While scanning the various dumped processes, only Avast identified one specific process as possibly infected, process *executable.1724.exe* which correlates to *explorer.exe*. However, a *strings*-based analysis of the executable's code indicates that it may potentially contain malicious code. However, at this point, this remains inconclusive without direct reverse engineering of the code.

A *file*-based analysis of the dumped processes has revealed that they were all identified as Windows 32-bit executables with the exception of dumped process file *executable.844.exe*, identified and confirmed as an empty file.

2.3.3.6.3 SHA1 and fuzzy hashes

SHA1 hashing has revealed that of the 19 dumped processes, four of them are identical. Specifically, *executable.856.exe*, *executable.936.exe*, *executable.1088.exe* and *executable.1148.exe* share the same SHA1 hash, all of which have been identified as various instances of *svchost.exe*.

Interestingly, while validating the SHA1 hashes of the dumped processes against the NSRL, only one specific hash match was identified, DA39A3EE5E6B4B0D3255BFEF95601890AFD80709. However, this particular hash matched 339,184 NSRL 2.41 entries; far too many to ever incorporate into this report. Thus, with respect to this hash, the vast majority are likely false positives.

Fuzzy hash matching conducted between the dumped processes identified 11 matches in all, as shown below in Table 20. Of these, six were identified as 100% matches. The SHA1 hashes of these 11 matches are found below in Table 19.

Table 19: SHA1 hash of 100% similarity matched files (sorted by hash).

SHA1 Hash	Procexedump
49d2e77a2ea666a3abccfa7beda276edf14d753b	executable.856.exe
49d2e77a2ea666a3abccfa7beda276edf14d753b	executable.936.exe
49d2e77a2ea666a3abccfa7beda276edf14d753b	executable.1088.exe
49d2e77a2ea666a3abccfa7beda276edf14d753b	executable.1148.exe
6f5a05ca36d4dded800a936dccb8fc7319e4dca3	executable.468.exe
a708f135e8ed7df8fca3eacbe8a80e710f96d270	executable.1732.exe
ba7630d931df003751faae76d4c2413af68ea154	executable.1028.exe

Table 19 clearly indicates four unique hashes, which when taken in conjunction with the information displayed in Table 20, show that these matches pair up in specific ways.

Table 20: Fuzzy hash similarities established between Procexedump samples (sorted by %).

Procexedump	Procexedump	Match (in %)
executable.1148.exe	executable.1088.exe	(100)
executable.856.exe	executable.1088.exe	(100)
executable.856.exe	executable.1148.exe	(100)
executable.936.exe	executable.1088.exe	(100)
executable.936.exe	executable.1148.exe	(100)

Procexedump	Procexedump	Match (in %)
executable.936.exe	executable.856.exe	(100)
executable.1088.exe	executable.1028.exe	(66)
executable.1148.exe	executable.1028.exe	(66)
executable.856.exe	executable.1028.exe	(66)
executable.936.exe	executable.1028.exe	(66)
executable.468.exe	executable.1732.exe	(58)

Similarity matching between these dumped processes and those samples obtained using the *malfind*, *dlldump*, *moddump* and *memdump* plugins have established that matches were identified with some of the dumped DLLs. Specifically, in all 52 matches were identified between them dumped DLLs and dumped process and 30 of these matches were found to be 100% matches. These similarity matches are listed in Annex E.6. What these similarities indicate, is that upon having examined the full file type analysis of both sets of dumped files (processes and DLLs; not included herein), some *dlldump* dumped samples were not expected. Specifically, the *dlldump* plugin sometimes dumped processes instead of DLLs, or at the very least, they appeared to the *file* command as actual executables.

Finally, possibly infected file *executable.1724.exe*, identified by Avast, is the exact same file as acquired by *dlldump* and confirmed by Avast, *module.1724.4a065d0.1000000.dll*.

2.3.3.6.4 Summary

The *procmemdump* plugin worked as expected and indications of the very same infection were identified in dumped process *executable.1724.exe* as found in dumped DLL *module.1724.4a065d0.1000000.dll*.

2.3.3.7 Dumpfiles plugin

Similar to other memory dumping plugins, this specific plugin is used to dump all arbitrary objects in memory that have been cached. Files, including executables, DLLs, drivers, fonts, configuration files, portions of the registry, etc. are cached in system memory. This plugin relies on the manner in which files are mapped into memory and then attempts to reconstruct it to disk. The plugin accepts physical address offsets for objects and for memory, represented by command line parameter “-Q” and “-o”, respectively.

This plugin was not part of the Volatility 2.2 distribution as used in previous reports [1][2][3][4]. It is instead entirely new to version 2.3.

2.3.3.7.1 Running the plugin

The objective in running this plugin was to attempt to obtain data files for analysis that were not possible using the other memory dumping plugins. Although the author had interest in passing all dumped files through multiple scanners, it would be preferable if the suspected malicious driver could be obtained. Thus, two commands were issued. The first to dump all mapped-cached objects to disk and the second to dump the physical object memory-based offset of the suspicious driver. The commands were:

```
$ volatility -f tigger.vmem dumpfiles --dump-dir=dumpfiles

$ volatility -f tigger.vmem dumpfiles -Q 0x01094ea0 --dump-
dir=dumpfiles
```

The memory address offset *0x01094ea0* was identified in Section 2.3.2.7 using the *filescan* plugin.

In all, the first command succeeded in dumping 317 files whereas the second command dumped two files, both of which were mapped back to the malicious driver as per this plugin's output:

```
ImageSectionObject 0x01094ea0   None
\Device\HarddiskVolume1\WINDOWS\system32\drivers\tmryqyrznr
2.sys

DataSectionObject 0x01094ea0   None
\Device\HarddiskVolume1\WINDOWS\system32\drivers\tmryqyrznr
2.sys
```

The malicious driver has finally been obtained and was remapped to disk files *file.None.0x80f2f8d0.dat* and *file.None.0xff3772c0.img*.

2.3.3.7.2 AV scanning and file type determination

File analysis has identified 63 of the dumped files as arbitrary data files. Three were identified as Internet Explorer cache files and four as Windows Registry files. The remaining 249 files were recognized as various PE-based files (e.g., DLLs, console and GUI applications and programs). Three of these were identified as device drivers; two of the files were the successfully obtained rootkits alluded to by previous evidence in this report whereas the remaining identified driver, *file.608.0x80efb928.img*, contains too little code beyond its PE header to be certain of its file type.

While scanning the various files dumped using the two aforementioned Volatility commands, the following files were identified as infected by the different scanners:

Table 21: Multi-scanner results for dumpfiles samples.

Scanner	Dumpfiles	Infection Identification
Avast AVG BitDefender McAfee	file.None.0xff3772c0.img	Win32:Trojan-gen Trojan horse Generic12.BOSL Rootkit.Agent.AITB BackDoor-DTN!sys trojan

Scanner	Dumpfiles	Infection Identification
Avast AVG BitDefender McAfee	file.None.0x80f2f8d0.dat	Win32:Trojan-gen Trojan horse Generic12.BOSL Rootkit.Agent.AITB BackDoor-DTN!sys Trojan
AVG F-Prot	file.1724.0xff255878.dat	Trojan horse Injector.CU W32/Damaged_File.B.gen!Eldorado
AVG	file.1724.0xff25aae0.img	Trojan horse Injector.CU
Comodo	file.688.0xff36e0e8.img	Gen:Variant.Symmi.25418

Based on the information presented in the above table, little doubt should remain that device drivers *file.None.0xff3772c0.img* and *file.None.0x80f2f8d0.dat* are in fact the rootkit sought after.

Interestingly, Comodo detected 20 of the 319 dumped files (317 using the first Volatility command, two using the second) as Heur.Corrupt.PE, a generic and possibly false positive assessment by the scanner. The following files were detected as such by the scanner:

file.1028.0x80ef7840.img
file.1028.0x80f0b428.img
file.1028.0x80ffb9b0.img
file.1028.0xff149920.img
file.1028.0xff1fbc78.img
file.1028.0xff210610.dat
file.1028.0xff287f08.dat
file.1028.0xff3832b8.img
file.1028.0xff38d008.img
file.1028.0xff3b4560.img
file.632.0x80f63868.dat
file.632.0xff26ee88.dat
file.632.0xff277ef8.dat
file.676.0x80fcb6c8.dat
file.676.0xff2316b0.dat
file.688.0x80f555e0.img
file.688.0xff3696b0.dat
file.844.0x80f0dd00.dat
file.844.0xff272ad0.dat
file.856.0xff380eb8.img

Strings analysis of *file.1724.0xff25aae0.img* indicates that it may potentially contain malicious code. However, at this point, this remains inconclusive without direct reverse engineering of the code. The same can be said for *file.688.0xff36e0e8.img*. The former appears to contain Windows APIs commonly used by malware whereare the latter contains suspicious function names.

2.3.3.7.3 SHA1 and fuzzy hashes

SHA1 analysis has identified no matches between any of the files dumped using the *dumpfiles* plugin. Fuzzy hash comparison between these files, using a threshold of 80% has found 34 file matches, as identified in the following table:

Table 22: Fuzzy hash similarities established between dumpfiles samples (sorted by %).

Dumpfiles	Dumpfiles	Match (in %)
file.1028.0xff134388.vacb	file.1028.0x80fb0360.dat	(100)
file.1028.0xff141ed0.vacb	file.1028.0x80fb0710.dat	(100)
file.1028.0xff14fab8.vacb	file.1028.0x80f00008.dat	(100)
file.1028.0xff154008.vacb	file.1028.0x80ff9b50.dat	(100)
file.1028.0xff229ed0.vacb	file.1028.0x80f00550.dat	(100)
file.1028.0xff244e78.dat	file.1028.0xff2025d0.vacb	(100)
file.1028.0xff265d50.vacb	file.1028.0x80fb0778.dat	(100)
file.1028.0xff2805d8.dat	file.1028.0x80f74460.vacb	(100)
file.1724.0xff239fa0.dat	file.1724.0x80f67e68.img	(100)
file.4.0x80fb0160.dat	file.4.0x80f2f0f8.vacb	(100)
file.4.0x80ff8368.dat	file.4.0x80fb2940.vacb	(100)
file.4.0xff22a278.vacb	file.4.0xff14f248.dat	(100)
file.4.0xff23aed0.vacb	file.4.0xff14bf80.dat	(100)
file.4.0xff272a58.dat	file.4.0xff24fed0.vacb	(100)
file.4.0xff2904a0.vacb	file.4.0x80f72cc0.dat	(100)
file.4.0xff29fed0.vacb	file.4.0x80f72a58.dat	(100)
file.4.0xff2a4168.vacb	file.4.0x80fb71b0.dat	(100)
file.4.0xff2a44d0.vacb	file.4.0x80fb7218.dat	(100)
file.4.0xff387aa8.vacb	file.4.0x80fb5f38.dat	(100)
file.None.0xff3772c0.img	file.None.0x80f2f8d0.dat	(100)
file.1724.0xff27acc8.dat	file.1724.0xff128b70.img	(99)
file.1028.0x80fb0778.dat	file.1028.0x80fb0710.dat	(91)
file.1028.0xff141ed0.vacb	file.1028.0x80fb0778.dat	(91)
file.1028.0xff265d50.vacb	file.1028.0x80fb0710.dat	(91)
file.1028.0xff265d50.vacb	file.1028.0xff141ed0.vacb	(91)
file.632.0xff37d388.img	file.632.0xff2673f8.dat	(91)
file.1028.0x80fb0778.dat	file.1028.0x80f00008.dat	(86)
file.1028.0xff14fab8.vacb	file.1028.0x80fb0778.dat	(86)
file.1028.0xff265d50.vacb	file.1028.0x80f00008.dat	(86)

Dumpfiles	Dumpfiles	Match (in %)
file.1028.0xff265d50.vacb	file.1028.0xff14fab8.vacb	(86)
file.1028.0x80fb0710.dat	file.1028.0x80f00008.dat	(82)
file.1028.0xff141ed0.vacb	file.1028.0x80f00008.dat	(82)
file.1028.0xff14fab8.vacb	file.1028.0x80fb0710.dat	(82)
file.1028.0xff14fab8.vacb	file.1028.0xff141ed0.vacb	(82)

SHA1 analysis of files *file.None.0xff3772c0.img* and *file.None.0x80f2f8d0.dat* reveals that they are not the same. However, a fuzzy hash comparison between them reveals that they are 100% similar. Thus, they must differ only by a few bytes. Fuzzy hash comparison between these two files and those data carved from the memory image show no similarity.

SHA1 comparison between the data carved files and those from the *dumpfiles* plugin reveal no matches. Fuzzy hash matching between these files using a threshold of 70% has revealed four matches, as shown in the following table:

Table 23: Fuzzy hash similarities established between carved memory files and dumpfiles samples (sorted by %).

Carved Memory File	Dumpfiles	Match (in %)
f0125224_gdiext	file.1732.0xff38f220.img	(100)
f0139344.evt	file.676.0x80f72008.dat	(100)
f0139320.evt	file.676.0x80ffd6c8.dat	(93)
f0181368.dll	file.1028.0xff398408.img	(75)

SHA1 comparison between those files dumped using the *dumpfiles* plugin and those obtained using previous dumping plugins including *dlldump*, *malfind*, *memdump*, *procexedump*, *moddump* as well as those obtained via the data carving of the memory, image have identified no matches whatsoever.

However, fuzzy hash matching between these samples dumped using the *dlldump*, *malfind*, *memdump*, *procexedump*, *moddump* plugins and the *dumpfiles* plugin has resulted in various matches. None matches were established between samples dumped from *dumpfiles* versus those from *malfind* or *memdump*. One partial match was established between *moddump* and *dumpfiles* (see Table 24). Fifteen were identified between *procexedump* and *dumpfiles* using a threshold of 70% (see). Finally, 153 matches were established between *dlldump* and *dumpfiles*, as listed in Annex E.7.

Table 24: Fuzzy hash similarity established between moddump and dumpfiles sample.

Moddump	Dumpfiles	Match (in %)
driver.fc67b000.sys	file.1028.0x80fad3a8.img	(35)

Table 25: Fuzzy hash similarities established between procexedump and dumpfiles samples (sorted by %).

Procexedump	Dumpfiles	Match (in %)
executable.1088.exe	file.856.0xff380eb8.img	(100)
executable.1148.exe	file.856.0xff380eb8.img	(100)
executable.688.exe	file.688.0x80f2a300.img	(100)
executable.856.exe	file.856.0xff380eb8.img	(100)
executable.936.exe	file.856.0xff380eb8.img	(100)
executable.1432.exe	file.1432.0x80fb3240.img	(94)
executable.452.exe	file.452.0xff281b10.img	(88)
executable.632.exe	file.632.0x80ff40c0.img	(88)
executable.432.exe	file.432.0x80fb4f30.img	(83)
executable.1668.exe	file.1668.0xff3bcbc8.img	(82)
executable.828.exe	file.828.0xff245990.img	(82)
executable.1724.exe	file.1724.0xff25aae0.img	(80)
executable.468.exe	file.1732.0x80f73380.img	(80)
executable.1084.exe	file.1084.0xff3af0d8.img	(72)
executable.676.exe	file.676.0x81023c50.img	(72)

Ultimately, no cross-plugin matches identified any similarities between dumped rootkit samples *file.None.0xff3772c0.img* and *file.None.0x80f2f8d0.dat*.

2.3.3.7.4 Summary

The *dumpfile* plugin is very powerful, capable of dumping a wide assortment of object types from a given memory image. Moreover, it enables the investigator to dump specific objects from memory in so long as a memory address is provided. Without this plugin, it would not have been otherwise possible to obtain the well-hidden rootkit long suspected of residing within this memory image. Furthermore, as data carving and running the *dumpfile* plugin without addresses has revealed that without a precise memory address, some objects hidden in memory remain that way. Thus, in order to have obtained this rootkit the previous steps conducted thus far in this investigation were necessary in order to have obtained the required memory address for the suspected rootkit.

2.3.3.8 Summary and analysis

This step has clearly indicated that attempting to dump various elements from a suspected memory image can yield tangible results. Although none of the other plugins run in this step were as concrete as the dumpfile plugin, without them other possibly infected objects would have been missed. The variously dumped DLL and process samples will have to be further reverse engineered in order to determine which are truly infected and how they tie in to this specific infection and investigation.

Finally, although the various fuzzy hash analyses did not reveal that any of the files suspected infection were related to other non-suspected files, this type of analysis is always of use as it has the potential to reveal new associations of infection.

2.3.4 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 F provides a list of registry keys commonly used by malware. It is nearly the same list as found in report [4] where the names of the Stuxnet device drivers have replaced with the Tigger rootkit device driver.

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 tigger.vmem hivelist*":

Table 26: Volatility Hivelist plugin output.

Virtual Address	Physical Address	Filename and Location
0xe1c49008	0x036dc008	\Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
0xe1c41b60	0x04010b60	\Device\HarddiskVolume1\Documents and Settings\LocalService\NTUSER.DAT
0xe1a39638	0x021eb638	\Device\HarddiskVolume1\Documents and Settings\NetworkService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
0xe1a33008	0x01f98008	\Device\HarddiskVolume1\Documents and Settings\NetworkService\NTUSER.DAT
0xe153ab60	0x06b7db60	\Device\HarddiskVolume1\WINDOWS\system32\config\software
0xe1542008	0x06c48008	\Device\HarddiskVolume1\WINDOWS\system32\config\default
0xe1537b60	0x06ae4b60	\SystemRoot\System32\Config\SECURITY

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

Virtual Address	Physical Address	Filename and Location
0xe1544008	0x06c4b008	\Device\HarddiskVolume1\WINDOWS\system32\config\SAM
0xe13ae580	0x01bbd580	[no name]
0xe101b008	0x01867008	\Device\HarddiskVolume1\WINDOWS\system32\config\system
0xe1008978	0x01824978	[no name]
0xe1e158c0	0x009728c0	\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
0xe1da4008	0x00f6e008	\Device\HarddiskVolume1\Documents and Settings\Administrator\NTUSER.DAT

2.3.4.2 Printkey plugin

Using all proposed registry keys identified in Annex F, 1053 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. This plugin uses virtual memory address offsets.

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 from said text file:

```
Registry: User Specified
Key name: 0000 (S)
Last updated: 2010-08-15 19:26:13 UTC+0000

Subkeys:
 (V) Control

Values:
REG_SZ        Service         : (S) tmryqyrznr2.sys^@
REG_DWORD     Legacy          : (S) 1
REG_DWORD     ConfigFlags     : (S) 0
REG_SZ        Class           : (S) LegacyDriver^@
REG_SZ        ClassGUID       : (S) {8ECC055D-047F-11D1-A537-
0000F8753ED1}^@
REG_SZ        DeviceDesc      : (S) tmryqyrznr2.sys^@
Legend: (S) = Stable   (V) = Volatile

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

Registry: User Specified
Key name: RunOnce (S)
```

```
Last updated: 2010-08-15 19:26:07 UTC+0000

Subkeys:

Values:
REG_SZ        tdss            : (S)
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\329000.exe
Legend: (S) = Stable   (V) = Volatile
```

This registry information pertains to both the identified rootkit and what was likely a malware dropper initiated by the user.

Thus, the persistence of this infection was made possible through the Windows registry which loaded the rootkit and perpetuated 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 relevant to the infection:

```
REG_BINARY   UEME_RUNPATH:C:\Documents and
Settings\Administrator\Desktop\tigger.exe :
ID:             2
Count:          1
Last updated:   2010-08-15 19:25:52 UTC+0000
0x00000000  02 00 00 00 06 00 00 00 a0 97 71 ac af 3c cb 01
..........q..<..
```

This UserAssist key indicates that the user initiated the infection instantiating an executable named *tigger.exe*.

2.3.5 Step 5: Miscellaneous

This final step examines two additional lines of inquiry. More specifically, it may be possible to determine, at least partially, the capabilities of the malicious device driver and perhaps any encryption keys used by rootkit or other potentially malicious software component in memory.

2.3.5.1 Devicetree

The Volatility *devicetree* plugin is used to determine the relationship between device drivers and their required Windows devices. In so doing, it may be possible to determine what Windows device is used by a given driver and hence the underlying purpose of a malicious driver. Upon running command "*volatility -f tigger.vmem devicetree*," without pruning, the following output was generated:

```
DRV 0x01058388 \Driver\Fdc
---| DEV 0xff35d150 FloppyPDO0 FILE_DEVICE_DISK
------| ATT 0xff3567f8 FloppyPDO0 - \Driver\Flpydisk FILE_DEVICE_DISK
---| DEV 0x80fa6a98  FILE_DEVICE_CONTROLLER
DRV 0x01058e28 \Driver\serenum
---| DEV 0x80fa6e88  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80ef5480  FILE_DEVICE_BUS_EXTENDER
DRV 0x01059258 \Driver\Serial
---| DEV 0x80fa6040 Serial1 FILE_DEVICE_SERIAL_PORT
------| ATT 0x80fa6e88 Serial1 - \Driver\serenum FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80ef5638 Serial0 FILE_DEVICE_SERIAL_PORT
------| ATT 0x80ef5480 Serial0 - \Driver\serenum FILE_DEVICE_BUS_EXTENDER
DRV 0x010593e8 \Driver\Parport
---| DEV 0x80efba38 Parallel0 FILE_DEVICE_PARALLEL_PORT
---| DEV 0x80ef5030 ParallelPort0 FILE_DEVICE_PARALLEL_PORT
DRV 0x0106fca0 \Driver\IpNat
---| DEV 0x80fbe570 IPNAT FILE_DEVICE_NETWORK
DRV 0x0108ef38
DRV 0x010ac3b0 \Driver\agp440
---| DEV 0x80fe72f8  FILE_DEVICE_BUS_EXTENDER
DRV 0x010adf38 \Driver\rdpdr
---| DEV 0x80f4cf10 RdpDrDvMgr FILE_DEVICE_UNKNOWN
---| DEV 0xff3ca6e8 RdpDr FILE_DEVICE_NETWORK_FILE_SYSTEM
---| DEV 0x80feeca0 RdpDrPort FILE_DEVICE_NETWORK_REDIRECTOR
DRV 0x010aee28 \Driver\Update
---| DEV 0x80febe10 Processor FILE_DEVICE_UNKNOWN
DRV 0x010b2158 \Driver\Tcpip
---| DEV 0x80febb98 RawIp FILE_DEVICE_NETWORK
---| DEV 0xff3c51a0 Udp FILE_DEVICE_NETWORK
---| DEV 0xff3824e0 Tcp FILE_DEVICE_NETWORK
---| DEV 0x80efb1f0 IPMULTICAST FILE_DEVICE_NETWORK
---| DEV 0xff379aa0 Ip FILE_DEVICE_NETWORK
DRV 0x010b6ac8 \Driver\NDProxy
---| DEV 0x80f53c90 NDProxy FILE_DEVICE_NETWORK
DRV 0x010cd5f8 \FileSystem\Ntfs
---| DEV 0x80f69770  FILE_DEVICE_DISK_FILE_SYSTEM
------| ATT 0x8102d360  - \FileSystem\sr FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x80f6a4e0 Ntfs FILE_DEVICE_DISK_FILE_SYSTEM
------| ATT 0x8102d020 Ntfs - \FileSystem\sr FILE_DEVICE_DISK_FILE_SYSTEM
DRV 0x010cdd28 \Driver\KSecDD
---| DEV 0x80f6ac10 KsecDD FILE_DEVICE_KSEC
DRV 0x010cdf38 \FileSystem\sr
---| DEV 0xff130b28  FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x8102d360  FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x8102d020  FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0x80f6ae20 SystemRestore FILE_DEVICE_UNKNOWN
DRV 0x01105718 \Driver\intelppm
---| DEV 0x80fa2330  FILE_DEVICE_UNKNOWN
DRV 0x01105ae0 \Driver\CmBatt
---| DEV 0x80fa2810 AcAdapter1 FILE_DEVICE_BATTERY
DRV 0x01105df0 \Driver\usbehci
---| DEV 0xff351030 USBPDO-0 FILE_DEVICE_BUS_EXTENDER
------| ATT 0xff354488 USBPDO-0 - \Driver\usbhub UNKNOWN
---| DEV 0xff3d4028 USBFDO-1 FILE_DEVICE_CONTROLLER
DRV 0x01106318 \Driver\es1371
---| DEV 0x80efa428 00000077 FILE_DEVICE_BUS_EXTENDER
------| ATT 0x80fae658 00000077 - \Driver\gameenum
        FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fa2030 00000074 FILE_DEVICE_KS
DRV 0x01106d58 \Driver\vmxnet
---| DEV 0x80fa3518 {D50E22C4-A428-4EF8-A24C-45BFC93B64B7}
     FILE_DEVICE_PHYSICAL_NETCARD
DRV 0x011076c0 \Driver\usbuhci
---| DEV 0xff3a1618 USBPDO-1 FILE_DEVICE_BUS_EXTENDER
------| ATT 0x80efc168 USBPDO-1 - \Driver\usbhub UNKNOWN
---| DEV 0x80ee1028 USBFDO-0 FILE_DEVICE_CONTROLLER
DRV 0x01107918 \Driver\vmx_svga
---| DEV 0xff342038 Video1 FILE_DEVICE_VIDEO
```

```
---| DEV 0xff3e3038 Video0 FILE_DEVICE_VIDEO
DRV 0x01107b78 \Driver\vmci
---| DEV 0x80fa4a10 vmci FILE_DEVICE_CONTROLLER
DRV 0x01108030 \Driver\Cdrom
---| DEV 0x80fa4030 CdRom0 FILE_DEVICE_CD_ROM
------| ATT 0x80fa4c60 CdRom0 - \Driver\redbook FILE_DEVICE_CD_ROM
DRV 0x01108db8 \Driver\redbook
---| DEV 0x80fa4c60  FILE_DEVICE_CD_ROM
DRV 0x011099a0 \Driver\Imapi
---| DEV 0x80fa5420  FILE_DEVICE_MASS_STORAGE
------| ATT 0x80fa4030  - \Driver\Cdrom FILE_DEVICE_CD_ROM
---------| ATT 0x80fa4c60  - \Driver\redbook FILE_DEVICE_CD_ROM
DRV 0x0110a790 \Driver\Mouclass
---| DEV 0xff20f030 PointerClass3 FILE_DEVICE_MOUSE
---| DEV 0xff1eb030 PointerClass2 FILE_DEVICE_MOUSE
---| DEV 0x80feea18 PointerClass1 FILE_DEVICE_MOUSE
---| DEV 0x80ef6e48 PointerClass0 FILE_DEVICE_MOUSE
DRV 0x0110a888 \Driver\vmmouse
---| DEV 0x80ef6020  FILE_DEVICE_MOUSE
------| ATT 0x80ef6e48  - \Driver\Mouclass FILE_DEVICE_MOUSE
DRV 0x0110b2a8 \Driver\Kbdclass
---| DEV 0x80fedc50 KeyboardClass1 FILE_DEVICE_KEYBOARD
---| DEV 0x80fa7e38 KeyboardClass0 FILE_DEVICE_KEYBOARD
DRV 0x0110b940 \Driver\i8042prt
---| DEV 0x80fa73e8  FILE_DEVICE_8042_PORT
------| ATT 0x80ef6020  - \Driver\vmmouse FILE_DEVICE_MOUSE
---------| ATT 0x80ef6e48  - \Driver\Mouclass FILE_DEVICE_MOUSE
---| DEV 0x80fa7020  FILE_DEVICE_8042_PORT
------| ATT 0x80fa7e38  - \Driver\Kbdclass FILE_DEVICE_KEYBOARD
DRV 0x0110c5f0 \Driver\swenum
---| DEV 0xff14a768 KSENUM#00000002 FILE_DEVICE_UNKNOWN
------| ATT 0x80fcd738 KSENUM#00000002 - \Driver\sysaudio FILE_DEVICE_KS
---| DEV 0xff2812b0 KSENUM#00000001 FILE_DEVICE_UNKNOWN
------| ATT 0x80fca768 KSENUM#00000001 - \Driver\wdmaud FILE_DEVICE_KS
---| DEV 0x80f4bf10  FILE_DEVICE_BUS_EXTENDER
DRV 0x01110b40 \Driver\Fips
---| DEV 0xff3b8030 Fips FILE_DEVICE_FIPS
DRV 0x01114f38 \Driver\mouhid
---| DEV 0x80faee20  FILE_DEVICE_MOUSE
------| ATT 0xff20f030  - \Driver\Mouclass FILE_DEVICE_MOUSE
---| DEV 0xff1f1020  FILE_DEVICE_MOUSE
------| ATT 0xff1eb030  - \Driver\Mouclass FILE_DEVICE_MOUSE
DRV 0x01129238 \Driver\atapi
---| DEV 0x8101e6a8 IdeDeviceP1T0L0-5 FILE_DEVICE_CD_ROM
------| ATT 0x80fa5420 IdeDeviceP1T0L0-5 - \Driver\Imapi
        FILE_DEVICE_MASS_STORAGE

---------| ATT 0x80fa4030 IdeDeviceP1T0L0-5 - \Driver\Cdrom
           FILE_DEVICE_CD_ROM
------------| ATT 0x80fa4c60 IdeDeviceP1T0L0-5 - \Driver\redbook
              FILE_DEVICE_CD_ROM
---| DEV 0x80f6b030 IdePort1 FILE_DEVICE_CONTROLLER
---| DEV 0x80f0f030 IdePort0 FILE_DEVICE_CONTROLLER
DRV 0x01129548 \Driver\VolSnap
---| DEV 0x81018c60  FILE_DEVICE_DISK
DRV 0x01129640 \Driver\PartMgr
---| DEV 0x8101b298  FILE_DEVICE_DISK
DRV 0x0112a5c0 \Driver\dmio
---| DEV 0x80fc6738 DmPnP FILE_DEVICE_NETWORK
---| DEV 0x80fc6030 DmIoDaemon FILE_DEVICE_NETWORK
---| DEV 0x80fc72f0 DmInfo FILE_DEVICE_NETWORK
---| DEV 0x80fc7458 DmConfig FILE_DEVICE_NETWORK
DRV 0x0112a7d8 \Driver\dmload
---| DEV 0x80fc76b8 DmLoader FILE_DEVICE_UNKNOWN
DRV 0x0112aaa0 \Driver\Ftdisk
---| DEV 0x81018030 HarddiskVolume1 FILE_DEVICE_DISK
------| ATT 0x81018c60 HarddiskVolume1 - \Driver\VolSnap FILE_DEVICE_DISK
---| DEV 0x80fc78d0 FtControl FILE_DEVICE_NETWORK
```

```
DRV 0x0112c1a8 \Driver\PCI
---| DEV 0xff8cf320  FILE_DEVICE_UNKNOWN
------| ATT 0xff91d030  - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0xff3d4028  - \Driver\usbehci FILE_DEVICE_CONTROLLER
---| DEV 0x80fc9328 NTPNP_PCI0042 FILE_DEVICE_UNKNOWN
------| ATT 0x81019030 NTPNP_PCI0042 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80fa2030 NTPNP_PCI0042 - \Driver\es1371 FILE_DEVICE_KS
---| DEV 0xff99ad30 NTPNP_PCI0041 FILE_DEVICE_UNKNOWN
------| ATT 0x80f202a0 NTPNP_PCI0041 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80fa3518 NTPNP_PCI0041 - \Driver\vmxnet
           FILE_DEVICE_PHYSICAL_NETCARD
---| DEV 0xff8cfcb0 NTPNP_PCI0040 FILE_DEVICE_UNKNOWN
------| ATT 0x810cb1f8 NTPNP_PCI0040 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80ee1028 NTPNP_PCI0040 - \Driver\usbuhci
           FILE_DEVICE_CONTROLLER
---| DEV 0x80f9c020  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f203a8  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f20570  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f20738  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f20900  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f20ac8  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f20c90  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f20e58  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f20020  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe63a8  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe6570  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe6738  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe6900  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe6ac8  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe6c90  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe6e58  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe6020  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9d3a8  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9d570  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9d738  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9d900  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9dac8  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9dc90  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9de58  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9d020  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f213a8  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f21570  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f21738  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f21900  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f21ac8  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f21c90  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f21e58  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f21020  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe7558  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x8101a350 NTPNP_PCI0039 FILE_DEVICE_UNKNOWN
------| ATT 0x80fe8658 NTPNP_PCI0039 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f9c020 NTPNP_PCI0039 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x8101a688 NTPNP_PCI0038 FILE_DEVICE_UNKNOWN
------| ATT 0x80fe8770 NTPNP_PCI0038 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f203a8 NTPNP_PCI0038 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x8101a9c0 NTPNP_PCI0037 FILE_DEVICE_UNKNOWN
------| ATT 0x80fe8888 NTPNP_PCI0037 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f20570 NTPNP_PCI0037 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x8101acf8 NTPNP_PCI0036 FILE_DEVICE_UNKNOWN
------| ATT 0x80fe89a0 NTPNP_PCI0036 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f20738 NTPNP_PCI0036 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x8101a030 NTPNP_PCI0035 FILE_DEVICE_UNKNOWN
------| ATT 0x80fe8ab8 NTPNP_PCI0035 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f20900 NTPNP_PCI0035 - \Driver\PCI
```

```
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x81019350 NTPNP_PCI0034 FILE_DEVICE_UNKNOWN
------| ATT 0x80fe8bd0 NTPNP_PCI0034 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f20ac8 NTPNP_PCI0034 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x81019688 NTPNP_PCI0033 FILE_DEVICE_UNKNOWN
------| ATT 0x80fe8ce8 NTPNP_PCI0033 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f20c90 NTPNP_PCI0033 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x810199c0 NTPNP_PCI0032 FILE_DEVICE_UNKNOWN
------| ATT 0x810226f0 NTPNP_PCI0032 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f20e58 NTPNP_PCI0032 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x81019cf8 NTPNP_PCI0031 FILE_DEVICE_UNKNOWN
------| ATT 0x80fe8e00 NTPNP_PCI0031 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f20020 NTPNP_PCI0031 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fc8120 NTPNP_PCI0030 FILE_DEVICE_UNKNOWN
------| ATT 0x80fe8f18 NTPNP_PCI0030 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80fe63a8 NTPNP_PCI0030 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fc8458 NTPNP_PCI0029 FILE_DEVICE_UNKNOWN
------| ATT 0x80fe8030 NTPNP_PCI0029 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80fe6570 NTPNP_PCI0029 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fc8790 NTPNP_PCI0028 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9f1f8 NTPNP_PCI0028 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80fe6738 NTPNP_PCI0028 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fc8ac8 NTPNP_PCI0027 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9f310 NTPNP_PCI0027 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80fe6900 NTPNP_PCI0027 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f494a8 NTPNP_PCI0026 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9f428 NTPNP_PCI0026 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80fe6ac8 NTPNP_PCI0026 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f497e0 NTPNP_PCI0025 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9f540 NTPNP_PCI0025 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80fe6c90 NTPNP_PCI0025 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f49b18 NTPNP_PCI0024 FILE_DEVICE_UNKNOWN
------| ATT 0x81022808 NTPNP_PCI0024 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80fe6e58 NTPNP_PCI0024 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f49e50 NTPNP_PCI0023 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9f658 NTPNP_PCI0023 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80fe6020 NTPNP_PCI0023 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xff99a358 NTPNP_PCI0022 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9f770 NTPNP_PCI0022 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f9d3a8 NTPNP_PCI0022 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xff99a690 NTPNP_PCI0021 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9f888 NTPNP_PCI0021 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f9d570 NTPNP_PCI0021 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xff99a9c8 NTPNP_PCI0020 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9f9a0 NTPNP_PCI0020 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f9d738 NTPNP_PCI0020 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xff91d688 NTPNP_PCI0019 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9fab8 NTPNP_PCI0019 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f9d900 NTPNP_PCI0019 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xff91d9c0 NTPNP_PCI0018 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9fbd0 NTPNP_PCI0018 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f9dac8 NTPNP_PCI0018 - \Driver\PCI
```

```
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xff91dcf8 NTPNP_PCI0017 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9fce8 NTPNP_PCI0017 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f9dc90 NTPNP_PCI0017 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xffaac238 NTPNP_PCI0016 FILE_DEVICE_UNKNOWN
------| ATT 0x81022920 NTPNP_PCI0016 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f9de58 NTPNP_PCI0016 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xffaac570 NTPNP_PCI0015 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9fe00 NTPNP_PCI0015 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f9d020 NTPNP_PCI0015 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xffaac8a8 NTPNP_PCI0014 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9ff18 NTPNP_PCI0014 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f213a8 NTPNP_PCI0014 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x810215b8 NTPNP_PCI0013 FILE_DEVICE_UNKNOWN
------| ATT 0x80f9f030 NTPNP_PCI0013 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f21570 NTPNP_PCI0013 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x810218f0 NTPNP_PCI0012 FILE_DEVICE_UNKNOWN
------| ATT 0x81022290 NTPNP_PCI0012 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f21738 NTPNP_PCI0012 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x81021c28 NTPNP_PCI0011 FILE_DEVICE_UNKNOWN
------| ATT 0x810223a8 NTPNP_PCI0011 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f21900 NTPNP_PCI0011 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xff8a88f8 NTPNP_PCI0010 FILE_DEVICE_UNKNOWN
------| ATT 0x810224c0 NTPNP_PCI0010 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f21ac8 NTPNP_PCI0010 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xff8a8c30 NTPNP_PCI0009 FILE_DEVICE_UNKNOWN
------| ATT 0x810225d8 NTPNP_PCI0009 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f21c90 NTPNP_PCI0009 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xff8f6348 NTPNP_PCI0008 FILE_DEVICE_UNKNOWN
------| ATT 0x81022a38 NTPNP_PCI0008 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f21e58 NTPNP_PCI0008 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xff8f6528 NTPNP_PCI0007 FILE_DEVICE_UNKNOWN
------| ATT 0x81022b50 NTPNP_PCI0007 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f21020 NTPNP_PCI0007 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xff8a8300 NTPNP_PCI0006 FILE_DEVICE_CONTROLLER

------| ATT 0x8101da30 NTPNP_PCI0006 - \Driver\vmscsi
        FILE_DEVICE_CONTROLLER
---| DEV 0xff8a8030 NTPNP_PCI0005 FILE_DEVICE_VIDEO
------| ATT 0xff3e3038 NTPNP_PCI0005 - \Driver\vmx_svga FILE_DEVICE_VIDEO
---| DEV 0xff8cf770 NTPNP_PCI0004 FILE_DEVICE_UNKNOWN
------| ATT 0x80fa4a10 NTPNP_PCI0004 - \Driver\vmci
        FILE_DEVICE_CONTROLLER
---| DEV 0xff8cf950 NTPNP_PCI0003 FILE_DEVICE_CONTROLLER
------| ATT 0x81022c68 NTPNP_PCI0003 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x810cd2d0 NTPNP_PCI0003 -  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fc9e50 NTPNP_PCI0002 FILE_DEVICE_UNKNOWN
------| ATT 0x81022d80 NTPNP_PCI0002 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80f9cdb0 NTPNP_PCI0002 - \Driver\isapnp
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xff91d368 NTPNP_PCI0001 FILE_DEVICE_UNKNOWN
------| ATT 0x8101a238 NTPNP_PCI0001 - \Driver\ACPI FILE_DEVICE_ACPI
---------| ATT 0x80fe7558 NTPNP_PCI0001 - \Driver\PCI
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0xff8cf030 NTPNP_PCI0000 FILE_DEVICE_UNKNOWN
---| DEV 0x80fc94f8  FILE_DEVICE_BUS_EXTENDER
DRV 0x01135670 \Driver\Raspti
```

```
---| DEV 0xff3c79d0 {AC004F37-7092-43C9-A7FC-EE28E95879FD}
     FILE_DEVICE_PHYSICAL_NETCARD
DRV 0x0114e3b0 \Driver\WS2IFSL
---| DEV 0xff363030 WS2IFSL FILE_DEVICE_NAMED_PIPE
DRV 0x0114ef38 \Driver\TermDD
---| DEV 0x80fee030 RDP_CONSOLE1 FILE_DEVICE_8042_PORT
------| ATT 0x80feea18 RDP_CONSOLE1 - \Driver\Mouclass FILE_DEVICE_MOUSE
---| DEV 0x80f4d770 RDP_CONSOLE0 FILE_DEVICE_8042_PORT
------| ATT 0x80fedc50 RDP_CONSOLE0 - \Driver\Kbdclass
        FILE_DEVICE_KEYBOARD
---| DEV 0x80f4d290 Termdd FILE_DEVICE_TERMSRV
DRV 0x011501c0 \Driver\Ptilink
---| DEV 0x80f4c1c8 ParTechInc2 FILE_DEVICE_PARALLEL_PORT
---| DEV 0x80f4a388 ParTechInc1 FILE_DEVICE_PARALLEL_PORT
---| DEV 0x80f4a6d0 ParTechInc0 FILE_DEVICE_PARALLEL_PORT
DRV 0x0117b670 \FileSystem\FltMgr
---| DEV 0x81018270 FltMgrMsg UNKNOWN
---| DEV 0x81018558 FltMgr FILE_DEVICE_DISK_FILE_SYSTEM
DRV 0x0117fcc8 \Driver\Disk
---| DEV 0x8102e7e8 DP(1)0x7000-0xff5f3000+1 FILE_DEVICE_DISK
---| DEV 0x8101b4c0 DR0 FILE_DEVICE_DISK
------| ATT 0x8101b298 DR0 - \Driver\PartMgr FILE_DEVICE_DISK
DRV 0x01181490 \Driver\vmscsi
---| DEV 0x8101ba38 vmscsi1Port2Path0Target0Lun0 FILE_DEVICE_DISK
------| ATT 0x8101b4c0 vmscsi1Port2Path0Target0Lun0 - \Driver\Disk
        FILE_DEVICE_DISK
---------| ATT 0x8101b298 vmscsi1Port2Path0Target0Lun0 - \Driver\PartMgr
           FILE_DEVICE_DISK
---| DEV 0x8101da30 vmscsi1 FILE_DEVICE_CONTROLLER
DRV 0x01185198
---| DEV 0x810cda08 CompositeBattery FILE_DEVICE_BATTERY
DRV 0x01190ab0 \FileSystem\Mup
---| DEV 0x8102d5c0 Mup FILE_DEVICE_MULTI_UNC_PROVIDER
---| DEV 0x8102d708 Root FILE_DEVICE_DFS
---| DEV 0x8102d998 Dfs FILE_DEVICE_DFS_FILE_SYSTEM
DRV 0x01190f38 \Driver\NDIS
---| DEV 0x8102de20 Ndis FILE_DEVICE_NETWORK
DRV 0x01233af8 \Driver\ACPI
---| DEV 0x810ce1b0 00000071 FILE_DEVICE_CONTROLLER
------| ATT 0x80fa6a98 00000071 - \Driver\Fdc FILE_DEVICE_CONTROLLER
---| DEV 0x810ce2c8 00000070 FILE_DEVICE_ACPI
------| ATT 0x80fa6040 00000070 - \Driver\Serial FILE_DEVICE_SERIAL_PORT
---------| ATT 0x80fa6e88 00000070 - \Driver\serenum
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x810ce3e0 0000006f FILE_DEVICE_ACPI
------| ATT 0x80ef5638 0000006f - \Driver\Serial FILE_DEVICE_SERIAL_PORT

---------| ATT 0x80ef5480 0000006f - \Driver\serenum
           FILE_DEVICE_BUS_EXTENDER
---| DEV 0x810ce4f8 0000006e FILE_DEVICE_ACPI
------| ATT 0x80ef5030 0000006e - \Driver\Parport
        FILE_DEVICE_PARALLEL_PORT
---| DEV 0x80f9c2c8 0000006d FILE_DEVICE_ACPI
---| DEV 0x80f9c3e0 0000006c FILE_DEVICE_ACPI
------| ATT 0x80fa73e8 0000006c - \Driver\i8042prt FILE_DEVICE_8042_PORT
---------| ATT 0x80ef6020 0000006c - \Driver\vmmouse FILE_DEVICE_MOUSE
------------| ATT 0x80ef6e48 0000006c - \Driver\Mouclass
              FILE_DEVICE_MOUSE
---| DEV 0x80f9c4f8  FILE_DEVICE_ACPI
------| ATT 0x80fa7020  - \Driver\i8042prt FILE_DEVICE_8042_PORT
---------| ATT 0x80fa7e38  - \Driver\Kbdclass FILE_DEVICE_KEYBOARD
---| DEV 0x80f9c610 0000006a FILE_DEVICE_ACPI
---| DEV 0x80f9c728 00000069 FILE_DEVICE_ACPI
---| DEV 0x80f9c840 00000068 FILE_DEVICE_ACPI
---| DEV 0x80f9c958  FILE_DEVICE_ACPI
---| DEV 0x80f9ca70  FILE_DEVICE_ACPI
---| DEV 0x80f9cb88 00000065 FILE_DEVICE_ACPI
---| DEV 0xff91d030  FILE_DEVICE_ACPI
```

```
------| ATT 0xff3d4028  - \Driver\usbehci FILE_DEVICE_CONTROLLER
---| DEV 0x81019030  FILE_DEVICE_ACPI
------| ATT 0x80fa2030  - \Driver\es1371 FILE_DEVICE_KS
---| DEV 0x80f202a0  FILE_DEVICE_ACPI
------| ATT 0x80fa3518  - \Driver\vmxnet FILE_DEVICE_PHYSICAL_NETCARD
---| DEV 0x810cb1f8 00000060 FILE_DEVICE_ACPI
------| ATT 0x80ee1028 00000060 - \Driver\usbuhci FILE_DEVICE_CONTROLLER
---| DEV 0x80fe8658 0000005f FILE_DEVICE_ACPI
------| ATT 0x80f9c020 0000005f - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe8770 0000005e FILE_DEVICE_ACPI
------| ATT 0x80f203a8 0000005e - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe8888 0000005d FILE_DEVICE_ACPI
------| ATT 0x80f20570 0000005d - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe89a0 0000005c FILE_DEVICE_ACPI
------| ATT 0x80f20738 0000005c - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe8ab8 0000005b FILE_DEVICE_ACPI
------| ATT 0x80f20900 0000005b - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe8bd0 0000005a FILE_DEVICE_ACPI
------| ATT 0x80f20ac8 0000005a - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe8ce8 00000059 FILE_DEVICE_ACPI
------| ATT 0x80f20c90 00000059 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe8e00 00000058 FILE_DEVICE_ACPI
------| ATT 0x80f20020 00000058 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe8f18 00000057 FILE_DEVICE_ACPI
------| ATT 0x80fe63a8 00000057 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fe8030 00000056 FILE_DEVICE_ACPI
------| ATT 0x80fe6570 00000056 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9f1f8 00000055 FILE_DEVICE_ACPI
------| ATT 0x80fe6738 00000055 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9f310 00000054 FILE_DEVICE_ACPI
------| ATT 0x80fe6900 00000054 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9f428 00000053 FILE_DEVICE_ACPI
------| ATT 0x80fe6ac8 00000053 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9f540 00000052 FILE_DEVICE_ACPI
------| ATT 0x80fe6c90 00000052 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9f658 00000051 FILE_DEVICE_ACPI
------| ATT 0x80fe6020 00000051 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9f770 00000050 FILE_DEVICE_ACPI
------| ATT 0x80f9d3a8 00000050 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9f888 0000004f FILE_DEVICE_ACPI
------| ATT 0x80f9d570 0000004f - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9f9a0 0000004e FILE_DEVICE_ACPI
------| ATT 0x80f9d738 0000004e - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9fab8 0000004d FILE_DEVICE_ACPI
------| ATT 0x80f9d900 0000004d - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9fbd0 0000004c FILE_DEVICE_ACPI
------| ATT 0x80f9dac8 0000004c - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9fce8 0000004b FILE_DEVICE_ACPI
------| ATT 0x80f9dc90 0000004b - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9fe00 0000004a FILE_DEVICE_ACPI
------| ATT 0x80f9d020 0000004a - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9ff18 00000049 FILE_DEVICE_ACPI
------| ATT 0x80f213a8 00000049 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9f030 00000048 FILE_DEVICE_ACPI
------| ATT 0x80f21570 00000048 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x81022290 00000047 FILE_DEVICE_ACPI
------| ATT 0x80f21738 00000047 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x810223a8 00000046 FILE_DEVICE_ACPI
------| ATT 0x80f21900 00000046 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x810224c0 00000045 FILE_DEVICE_ACPI
------| ATT 0x80f21ac8 00000045 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x810225d8 00000044 FILE_DEVICE_ACPI
------| ATT 0x80f21c90 00000044 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x810226f0 00000043 FILE_DEVICE_ACPI
------| ATT 0x80f20e58 00000043 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x81022808 00000042 FILE_DEVICE_ACPI
------| ATT 0x80fe6e58 00000042 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x81022920 00000041 FILE_DEVICE_ACPI
```

```
------| ATT 0x80f9de58 00000041 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x81022a38 00000040 FILE_DEVICE_ACPI
------| ATT 0x80f21e58 00000040 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x81022b50 0000003f FILE_DEVICE_ACPI
------| ATT 0x80f21020 0000003f - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x81022c68 0000003e FILE_DEVICE_ACPI
------| ATT 0x810cd2d0 0000003e -  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x81022d80 0000003d FILE_DEVICE_ACPI
------| ATT 0x80f9cdb0 0000003d - \Driver\isapnp FILE_DEVICE_BUS_EXTENDER
---| DEV 0x8101a238 0000003c FILE_DEVICE_ACPI
------| ATT 0x80fe7558 0000003c - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80fc9090 0000003b FILE_DEVICE_ACPI
---| DEV 0x81021238 0000003a FILE_DEVICE_ACPI
---| DEV 0xffaacbd0 00000039 FILE_DEVICE_ACPI
---| DEV 0x810421f8 00000038 FILE_DEVICE_ACPI
---| DEV 0x8105c1f8 00000037 FILE_DEVICE_ACPI
------| ATT 0x80fa2330 00000037 - \Driver\intelppm FILE_DEVICE_UNKNOWN
---| DEV 0x810c81f8 00000036 FILE_DEVICE_ACPI
------| ATT 0x80fa2810 00000036 - \Driver\CmBatt FILE_DEVICE_BATTERY
---| DEV 0x810b61f8 00000035 FILE_DEVICE_ACPI
------| ATT 0x80fc94f8 00000035 - \Driver\PCI FILE_DEVICE_BUS_EXTENDER
---| DEV 0x810cf020  FILE_DEVICE_ACPI
DRV 0x01233f38 \FileSystem\RAW
---| DEV 0x810d0bf0 RawTape FILE_DEVICE_TAPE_FILE_SYSTEM
---| DEV 0x810d0d08 RawCdRom FILE_DEVICE_CD_ROM_FILE_SYSTEM
---| DEV 0x810d0e20 RawDisk FILE_DEVICE_DISK_FILE_SYSTEM
DRV 0x01234770 \Driver\WMIxWDM
---| DEV 0x810d1488 WMIDataDevice FILE_DEVICE_UNKNOWN
DRV 0x01234ab8 \Driver\ACPI_HAL
---| DEV 0x810d1868 00000034 FILE_DEVICE_BUS_EXTENDER
------| ATT 0x810cf020 00000034 - \Driver\ACPI FILE_DEVICE_ACPI
---| DEV 0x810d1988  FILE_DEVICE_BUS_EXTENDER
DRV 0x01238338 \Driver\PnpManager
---| DEV 0x810d1cd0 00000032 FILE_DEVICE_CONTROLLER
------| ATT 0x80fa9e60 00000032 - \Driver\mssmbios FILE_DEVICE_UNKNOWN
---| DEV 0x810d1f10 00000031 FILE_DEVICE_CONTROLLER
------| ATT 0x80febe10 00000031 - \Driver\Update FILE_DEVICE_UNKNOWN
---| DEV 0x81099190 00000030 FILE_DEVICE_CONTROLLER
------| ATT 0x80f4bf10 00000030 - \Driver\swenum FILE_DEVICE_BUS_EXTENDER
---| DEV 0x810993d0 0000002f FILE_DEVICE_CONTROLLER
------| ATT 0x80fee030 0000002f - \Driver\TermDD FILE_DEVICE_8042_PORT
---------| ATT 0x80feea18 0000002f - \Driver\Mouclass FILE_DEVICE_MOUSE
---| DEV 0x81099610 0000002e FILE_DEVICE_CONTROLLER
------| ATT 0x80f4d770 0000002e - \Driver\TermDD FILE_DEVICE_8042_PORT
---------| ATT 0x80fedc50 0000002e - \Driver\Kbdclass
           FILE_DEVICE_KEYBOARD
---| DEV 0x81099850 0000002d FILE_DEVICE_CONTROLLER
------| ATT 0x80f4cf10 0000002d - \Driver\rdpdr FILE_DEVICE_UNKNOWN
---| DEV 0x81099a90 0000002c FILE_DEVICE_CONTROLLER
------| ATT 0xff3c79d0 0000002c - \Driver\Raspti
        FILE_DEVICE_PHYSICAL_NETCARD
---| DEV 0x81099cd0 0000002b FILE_DEVICE_CONTROLLER
------| ATT 0xff3c7030 0000002b - \Driver\PSched
        FILE_DEVICE_PHYSICAL_NETCARD
---| DEV 0x81099f10 0000002a FILE_DEVICE_CONTROLLER
------| ATT 0x80fed2b8 0000002a - \Driver\PSched
        FILE_DEVICE_PHYSICAL_NETCARD
---| DEV 0x810d2190 00000029 FILE_DEVICE_CONTROLLER
------| ATT 0xff3ca030 00000029 - \Driver\PptpMiniport
        FILE_DEVICE_PHYSICAL_NETCARD
---| DEV 0x810d23d0 00000028 FILE_DEVICE_CONTROLLER
------| ATT 0xff3c89d0 00000028 - \Driver\RasPppoe
        FILE_DEVICE_PHYSICAL_NETCARD
---| DEV 0x810d2610 00000027 FILE_DEVICE_CONTROLLER
------| ATT 0xff3c9a08 00000027 - \Driver\NdisWan
        FILE_DEVICE_PHYSICAL_NETCARD
---| DEV 0x810d2850 00000026 FILE_DEVICE_CONTROLLER
------| ATT 0xff3d81a0 00000026 - \Driver\Rasl2tp
```

```
        FILE_DEVICE_PHYSICAL_NETCARD
---| DEV 0x810d2a90 00000025 FILE_DEVICE_CONTROLLER
------| ATT 0xff3d8a48 00000025 - \Driver\audstub FILE_DEVICE_UNKNOWN
---| DEV 0x810d2cd0 00000024 FILE_DEVICE_CONTROLLER
------| ATT 0xff3d8b58 00000024 - \Driver\audstub FILE_DEVICE_UNKNOWN
---| DEV 0x810d2f10 00000023 FILE_DEVICE_CONTROLLER
------| ATT 0xff3d8c68 00000023 - \Driver\audstub FILE_DEVICE_UNKNOWN
---| DEV 0x8109a190 00000022 FILE_DEVICE_CONTROLLER
------| ATT 0xff3d8d78 00000022 - \Driver\audstub FILE_DEVICE_UNKNOWN
---| DEV 0x8109a3d0 00000021 FILE_DEVICE_CONTROLLER
------| ATT 0xff3d8e88 00000021 - \Driver\audstub FILE_DEVICE_UNKNOWN
---| DEV 0x8109a610 00000020 FILE_DEVICE_CONTROLLER
---| DEV 0x8109a850 0000001f FILE_DEVICE_CONTROLLER
---| DEV 0x8109aa90 0000001e FILE_DEVICE_CONTROLLER
---| DEV 0x8109acd0 0000001d FILE_DEVICE_CONTROLLER
---| DEV 0x8109af10 0000001c FILE_DEVICE_CONTROLLER
---| DEV 0x810d3190 0000001b FILE_DEVICE_CONTROLLER
---| DEV 0x810d33d0 0000001a FILE_DEVICE_CONTROLLER
---| DEV 0x810d3610 00000019 FILE_DEVICE_CONTROLLER
---| DEV 0x810d3850 00000018 FILE_DEVICE_CONTROLLER
---| DEV 0x810d3a90 00000017 FILE_DEVICE_CONTROLLER
---| DEV 0x810d3cd0 00000016 FILE_DEVICE_CONTROLLER
---| DEV 0x810d3f10 00000015 FILE_DEVICE_CONTROLLER
---| DEV 0x8109b190 00000014 FILE_DEVICE_CONTROLLER
---| DEV 0x8109b3d0 00000013 FILE_DEVICE_CONTROLLER
---| DEV 0x8109b610 00000012 FILE_DEVICE_CONTROLLER
---| DEV 0x8109b850 00000011 FILE_DEVICE_CONTROLLER
---| DEV 0x8109ba90 00000010 FILE_DEVICE_CONTROLLER
---| DEV 0x8109bcd0 0000000f FILE_DEVICE_CONTROLLER
---| DEV 0x8109bf10 0000000e FILE_DEVICE_CONTROLLER
---| DEV 0x810d4190 0000000d FILE_DEVICE_CONTROLLER
---| DEV 0x810d43d0 0000000c FILE_DEVICE_CONTROLLER
---| DEV 0x810d4610 0000000b FILE_DEVICE_CONTROLLER
---| DEV 0x810d4850 0000000a FILE_DEVICE_CONTROLLER
---| DEV 0x810d4a90 00000009 FILE_DEVICE_CONTROLLER
---| DEV 0x810d4cd0 00000008 FILE_DEVICE_CONTROLLER
---| DEV 0x810d4f10 00000007 FILE_DEVICE_CONTROLLER
---| DEV 0x8109c1f8 00000006 FILE_DEVICE_CONTROLLER
---| DEV 0x8109c438 00000005 FILE_DEVICE_CONTROLLER
---| DEV 0x8109c678 00000004 FILE_DEVICE_CONTROLLER
------| ATT 0x80fc78d0 00000004 - \Driver\Ftdisk FILE_DEVICE_NETWORK
---| DEV 0x8109c8b8 00000003 FILE_DEVICE_CONTROLLER
------| ATT 0x80fc6738 00000003 - \Driver\dmio FILE_DEVICE_NETWORK
---| DEV 0x8109caf8 00000002 FILE_DEVICE_CONTROLLER
------| ATT 0x810cda08 00000002 -  FILE_DEVICE_BATTERY
---| DEV 0x8109cdd8 00000001 FILE_DEVICE_CONTROLLER

------| ATT 0x810d1988 00000001 - \Driver\ACPI_HAL
        FILE_DEVICE_BUS_EXTENDER
---| DEV 0x8109d430  FILE_DEVICE_CONTROLLER
DRV 0x020ac150 \Driver\isapnp
---| DEV 0x80f9cca0  FILE_DEVICE_BUS_EXTENDER
---| DEV 0x80f9cdb0  FILE_DEVICE_BUS_EXTENDER
DRV 0x020ba228 \Driver\MountMgr
---| DEV 0x80fc7b98 MountPointManager FILE_DEVICE_NETWORK
DRV 0x021216a8
---| DEV 0x80fc7e60  FILE_DEVICE_CONTROLLER
------| ATT 0x80f6b030  - \Driver\atapi FILE_DEVICE_CONTROLLER
---| DEV 0x80fc7030 PciIde0Channel0-0 FILE_DEVICE_CONTROLLER
------| ATT 0x80f0f030 PciIde0Channel0-0 - \Driver\atapi
        FILE_DEVICE_CONTROLLER
---| DEV 0x810cd2d0 PciIde0 FILE_DEVICE_BUS_EXTENDER
DRV 0x02e2c318 \Driver\HTTP
---| DEV 0xff12ef18 AppPool FILE_DEVICE_NETWORK
---| DEV 0xff12e030 Filter FILE_DEVICE_NETWORK
---| DEV 0xff136168 Control FILE_DEVICE_NETWORK
DRV 0x03fc7030 \Driver\audstub
---| DEV 0xff3d8a48  FILE_DEVICE_UNKNOWN
```

```
---| DEV 0xff3d8b58  FILE_DEVICE_UNKNOWN
---| DEV 0xff3d8c68  FILE_DEVICE_UNKNOWN
---| DEV 0xff3d8d78  FILE_DEVICE_UNKNOWN
---| DEV 0xff3d8e88  FILE_DEVICE_UNKNOWN
DRV 0x03fc7960 \Driver\Rasl2tp
---| DEV 0xff3d81a0 {0381115A-3E2D-4FEA-BF6D-996255383226}
     FILE_DEVICE_PHYSICAL_NETCARD
DRV 0x04032448 \Driver\RasPppoe
---| DEV 0xff3c89d0 {2286112A-6F52-4590-BA30-0F5638DC58F6}
     FILE_DEVICE_PHYSICAL_NETCARD
DRV 0x040327b0 \Driver\mssmbios
---| DEV 0x80fa9e60  FILE_DEVICE_UNKNOWN
DRV 0x04032cb8 \Driver\NdisWan
---| DEV 0xff35f618 NdisWan FILE_DEVICE_NETWORK
---| DEV 0xff3c9a08 NdisWanIp FILE_DEVICE_PHYSICAL_NETCARD
DRV 0x04032ec8 \Driver\NdisTapi
---| DEV 0xff3cb030 NdisTapi UNKNOWN
DRV 0x041643e0 \Driver\Gpc
---| DEV 0xff3cad90 Gpc FILE_DEVICE_NETWORK
DRV 0x04164558 \Driver\PSched
---| DEV 0xff35c938 PSched FILE_DEVICE_NETWORK
---| DEV 0xff3c7030 {E97B3699-3BFB-4C56-9092-ADFAEAF3E50C}
     FILE_DEVICE_PHYSICAL_NETCARD
---| DEV 0x80fed2b8 {98F15284-32B4-4BEA-AF7D-185608FBA9B8}
     FILE_DEVICE_PHYSICAL_NETCARD
DRV 0x041647d0 \Driver\PptpMiniport
---| DEV 0xff3ca030 {AA4ABE7E-2C3C-4227-8C16-FA7C332F4194}
     FILE_DEVICE_PHYSICAL_NETCARD
DRV 0x041ddf38 \FileSystem\Msfs
---| DEV 0x80fad1f8 Mailslot FILE_DEVICE_MAILSLOT
DRV 0x041ed408 \Driver\NetBT
---| DEV 0xff1bb6b0 NetBT_Tcpip_{D50E22C4-A428-4EF8-A24C-45BFC93B64B7}
     FILE_DEVICE_NETWORK
---| DEV 0x80feb4a8 NetBt_Wins_Export FILE_DEVICE_NETWORK
---| DEV 0xff3735a8 NetbiosSmb FILE_DEVICE_NETWORK
DRV 0x0438bcc0 \FileSystem\Fs_Rec
---| DEV 0xff36af08 FatCdRomRecognizer FILE_DEVICE_CD_ROM_FILE_SYSTEM
---| DEV 0xff36a6b8 FatDiskRecognizer FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0xff362ae0 UdfsDiskRecognizer FILE_DEVICE_DISK_FILE_SYSTEM
---| DEV 0xff367428 UdfsCdRomRecognizer FILE_DEVICE_CD_ROM_FILE_SYSTEM
---| DEV 0xff363428 CdfsRecognizer FILE_DEVICE_CD_ROM_FILE_SYSTEM
DRV 0x0438e030 \FileSystem\MRxSmb
---| DEV 0xff3c5870 LanmanDatagramReceiver FILE_DEVICE_NETWORK_BROWSER
---| DEV 0xff3c5988 LanmanRedirector FILE_DEVICE_NETWORK_FILE_SYSTEM
DRV 0x043d1030 \FileSystem\Rdbss
---| DEV 0xff355c00 FsWrap FILE_DEVICE_NETWORK_FILE_SYSTEM
DRV 0x044180e8 \Driver\gameenum
---| DEV 0x80fae658  FILE_DEVICE_BUS_EXTENDER
DRV 0x0486eca8 \Driver\wdmaud
---| DEV 0x80fca768  FILE_DEVICE_KS
DRV 0x048f8da0 \Driver\IPSec
---| DEV 0xff393958 IPSEC FILE_DEVICE_NETWORK
DRV 0x048f9030 \FileSystem\vmhgfs
---| DEV 0x80f4bb20 hgfsInternal UNKNOWN
---| DEV 0x80ff01f0 HGFS FILE_DEVICE_NETWORK_FILE_SYSTEM
DRV 0x0493f880 \Driver\sysaudio
---| DEV 0x80fcd738 sysaudio FILE_DEVICE_KS
DRV 0x04a49880 \Driver\RasAcd
---| DEV 0xff382d80 RasAcd UNKNOWN
DRV 0x04a96ca8 \Driver\mnmdd
---| DEV 0x80fec2a8 Video3 FILE_DEVICE_VIDEO
DRV 0x04b60a18 \Driver\RDPCDD
---| DEV 0xff36f260 Video4 FILE_DEVICE_VIDEO
DRV 0x04be4b10 \Driver\Null
---| DEV 0xff367860 Null FILE_DEVICE_NULL
DRV 0x04be5550 \Driver\VgaSave
---| DEV 0xff374040 Video2 FILE_DEVICE_VIDEO
DRV 0x04be6030 \Driver\AFD
```

```
---| DEV 0xff37e250 Afd FILE_DEVICE_NAMED_PIPE
DRV 0x04be65f0 \Driver\Beep
---| DEV 0xff366a98 Beep FILE_DEVICE_BEEP
DRV 0x04be6a18 \FileSystem\Npfs
---| DEV 0xff365ce0 NamedPipe FILE_DEVICE_NAMED_PIPE
DRV 0x04bf9520 \Driver\Flpydisk
---| DEV 0xff3567f8 Floppy0 FILE_DEVICE_DISK
DRV 0x04c2b030 \FileSystem\NetBIOS
---| DEV 0xff38b030 Netbios FILE_DEVICE_TRANSPORT
DRV 0x04c2e4b0 \FileSystem\Cdfs
---| DEV 0xff3be030 Cdfs FILE_DEVICE_CD_ROM_FILE_SYSTEM
DRV 0x04d699c0 \Driver\usbhub
---| DEV 0x80ffe030 00000079 UNKNOWN
---| DEV 0x80f2e030 USBPDO-3 FILE_DEVICE_UNKNOWN
------| ATT 0x80ffe030 USBPDO-3 - \Driver\usbhub UNKNOWN
---| DEV 0x80ff4700 USBPDO-2 FILE_DEVICE_UNKNOWN
------| ATT 0x80f01bb8 USBPDO-2 - \Driver\usbccgp FILE_DEVICE_UNKNOWN
---| DEV 0x80efc168 00000076 UNKNOWN
---| DEV 0xff354488 00000075 UNKNOWN
DRV 0x05e36740 \FileSystem\Fastfat
---| DEV 0xff3a7d80 FatCdrom FILE_DEVICE_CD_ROM_FILE_SYSTEM
---| DEV 0xff3a5f18 Fat FILE_DEVICE_DISK_FILE_SYSTEM
------| ATT 0xff130b28 Fat - \FileSystem\sr FILE_DEVICE_DISK_FILE_SYSTEM
DRV 0x05f45030 \FileSystem\MRxDAV
---| DEV 0xff1dcae8 WebDavRedirector FILE_DEVICE_NETWORK_FILE_SYSTEM
DRV 0x05f46030 \Driver\VMMEMCTL
---| DEV 0xff24f4b8 vmmemctl FILE_DEVICE_UNKNOWN
DRV 0x05f8b030 \Driver\ParVdm
---| DEV 0x80f0e680 ParallelVdm0 FILE_DEVICE_PARALLEL_PORT
DRV 0x05fd22d8 \FileSystem\Srv
---| DEV 0xff21b6d0 LanmanServer FILE_DEVICE_NETWORK
DRV 0x06384d18 \Driver\Win32k
DRV 0x065a24c8 \Driver\Ndisuio
---| DEV 0xff255f18 Ndisuio FILE_DEVICE_NETWORK
DRV 0x06901030 \Driver\hidusb
---| DEV 0xff1fb9a0 0000007d FILE_DEVICE_UNKNOWN
------| ATT 0x80faee20 0000007d - \Driver\mouhid FILE_DEVICE_MOUSE
---------| ATT 0xff20f030 0000007d - \Driver\Mouclass FILE_DEVICE_MOUSE
---| DEV 0xff1fb578 0000007c FILE_DEVICE_UNKNOWN
------| ATT 0xff1f1020 0000007c - \Driver\mouhid FILE_DEVICE_MOUSE
---------| ATT 0xff1eb030 0000007c - \Driver\Mouclass FILE_DEVICE_MOUSE
---| DEV 0xff287800 _HID00000001 FILE_DEVICE_UNKNOWN
---| DEV 0x80ff8030 _HID00000000 FILE_DEVICE_UNKNOWN
DRV 0x06944030 \Driver\usbccgp
---| DEV 0xff283030 0000007b FILE_DEVICE_UNKNOWN
------| ATT 0xff287800 0000007b - \Driver\hidusb FILE_DEVICE_UNKNOWN
---| DEV 0xff266030 0000007a FILE_DEVICE_UNKNOWN
------| ATT 0x80ff8030 0000007a - \Driver\hidusb FILE_DEVICE_UNKNOWN
---| DEV 0x80f01bb8 00000078 FILE_DEVICE_UNKNOWN
DRV 0x069d4f38 \Driver\Wanarp
---| DEV 0xff1b9e20 WANARP FILE_DEVICE_NETWORK
```

The above Windows device list is long and arduous to both read and comprehend to non-Windows device driver/kernel experts, of which the author is neither. Thus, either attempting to read this list and determine which of the various devices is the keylogger (see Section 2.1.2) or rootkit is not readily possible for non-experts.

However, an in-depth look at this list does present various anomalous looking entries, yet nothing pertinent about them can be readily identified in the available literature. As such, how does one go about identifying the telltale marks of such a keylogger or rootkit? The author does not have an answer to this question nor has he been able to find one.

Finally, this rootkit, unlike R2D2 or Stuxnet's keylogger or rootkit components, does not readily show itself in such a list [3][4]. As such, the use of such plugins work only when the rootkit or keylogger do not go to extreme lengths to hide themselves from detection, as this rootkit and keylogger do.

2.3.5.2 Extract encryption keys

Although the documentation detailing the Tigger/Syzor infection found in references [6][7][8][9][10][11][12][13][14] do not specify the use of encryption keys. Nevertheless, multiple RSA keys were found in this memory image using the *rsakeyfind*[4] tool. However, the *interrogate*[5] program could not corroborate this. Thus, it is possible that some or all of these keys could be false positives. Follow-up using commercial software should be carried out by the investigator.

It is not known if the Tigger/Syzor infection makes use of RSA-based encryption as no information concerning this could be found. As each RSA key is comprised of two prime numbers, multiple exponents and a coefficient, listing the various RSA-based keys will consume too many pages for the small information it will bring, since these results could not be corroborated.

2.3.5.3 Summary and analysis

As this step has clearly shown, little additional information could be ascertained about the Tigger/Syzor malware. While this step was of use against R2D2 [3] and Stuxnet [4], it was of limited use here.

[4] Aeskeyfind can be found at https://citp.princeton.edu/research/memory/code/.

[5] Interrogate can be found at https://github.com/carmaa/interrogate.

3 Conclusion

What can be concluded from this work is 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.

While the Tigger/Syzor Trojan horse/rootkit had a limited attack vector, targeting primarily various financial institutions, with respect to its ability, it far exceeds the capabilities of all other malware investigated by the author with respect to this series of reports. Considering how readily detectable Stuxnet was, which is allegedly state-sponsored, considering the anti-detection mechanisms inherent in Tigger/Syzor, the question of its origins must be pondered. Was it state-sponsored? There is no credible information one way or the other.

It is interesting that very little media/web coverage was given to this particular piece of malware, as compared to previous malware including Zeus [1] and Stuxnet [4]. It likely received little coverage because of the limited number of systems infected and given the nature of the victims (financial institutions), they likely made efforts to keep the news under wraps as best they could.

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. The techniques and methodology presented herein will be of use, to varying extents, against newer and more difficult to analyse malware.

Unlike the Stuxnet infection, little technical information was available concerning the Tigger/Syzor. Nevertheless, this investigation did not make direct use of them during the analysis of this memory image.

This document is the fifth 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. Malware memory analysis for non-specialists: Investigating publicly available memory image for the Stuxnet worm. Scientific Report. Defence Research and Development Canada DRDC-RDDC-2013-R1. November 2013.

[5] 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. Last accessed May 2014.

[6] ThreatExpert. ThreatExpert Report: Infostealer.Banker.C, Backdoor:WinNT/Syzor.A, Hacktool.Rootkit!sd6. Informational web site. ThreatExpert. February 2009. http://www.threatexpert.com/report.aspx?md5=f945a45cf20722418a3036d557240b5d. Last accessed May 2014.

[7] Microsoft. Backdoor:WinNT/Syzor.A. Informational web site. Microsoft Malware Protection Center. 2013. http://www.microsoft.com/security/portal/threat/encyclopedia/Entry.aspx?Name=Backdoor%3AWinNT%2FSyzor.A#tab=2. Last accessed May 2014.

[8] Ligh, Michael Hale. Why I Enjoyed Tigger/Syzor. Information web site. MNIN Security Blog: Coding, Reversing, Exploiting. February 2009. http://mnin.blogspot.ca/2009/02/why-i-enjoyed-tiggersyzor.html#!/2009/02/why-i-enjoyed-tiggersyzor.html. Last accessed May 2014.

[9] SpamFighter. Security Experts Wand Computer users Against “Tigger” Trojan. Informational web site. SpamFighter. March 2009. http://www.spamfighter.com/News-11974-Security-Experts-Warn-Computer-Users-Against-%E2%80%9CTigger%E2%80%9D-Trojan.htm. Last accessed May 2014.

[10] Kassner, Michael. Tigger.A: Sophisticated trojan that likes stockbrokers. Online article. TechRepublic IT Security. TechRepublic. March 2009. http://www.techrepublic.com/blog/it-security/tiggera-sophisticated-trojan-that-likes-stockbrokers/. Last accessed May 2014.

[11] Wilson, Tim. 'Tigger' Trojan Keeps Security Researchers Hopping. Online article. DarkReading.com. March 2009. http://www.darkreading.com/attacks-breaches/tigger-trojan-keeps-security-researchers/215800583. Last accessed May 2014.

[12] Krebs, Brian. The Tigger Trojan: Icky, Sticky Stuff. Online article. The Washington Post. February 2009. http://voices.washingtonpost.com/securityfix/2009/02/the_t-i-double-guh-r_trojan_ic.html. Last accessed May 2014.

[13] ThreatExpert. ThreatExpert Report: New Malware.fa, Mal/HckPk-A, Mal/Basine-C, Backdoor:WinNT/Syzor.A. Informational web site. ThreatExpert. August 2008. http://www.threatexpert.com/report.aspx?md5=f945a45cf20722418a3036d557240b5d. Last accessed May 2014.

[14] Department of Homeland Security. Department of Homeland Security Daily Open Source Infrastructure Report for 6 March 2009. Security bulletin. Department of Homeland Security. March 2009.

[15] 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. Last accessed May 2014.

[16] 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. Last accessed May 2014.

[17] Microsoft TechNet. Network Ports Used by Key Microsoft Server Products. Support article. Microsoft. 2013. http://technet.microsoft.com/en-us/library/cc875824.aspx. Last accessed May 2014.

[18] Microsoft TechNet. Port Assignment for Commonly-Used Services. Support article. Microsoft. 2013. http://technet.microsoft.com/en-us/library/cc959833.aspx. Last accessed May 2014.

[19] 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. Last accessed May 2014.

[20] Speedguide.net. Port 1033 (tcp/udp). Informational web site. Speedguide.net. 2013. http://www.speedguide.net/port.php?port=1033. Last accessed May 2014.

[21] Speedguide.net. Port 1053 (tcp/udp). Informational web site. Speedguide.net. 2013. http://www.speedguide.net/port.php?port=1053. Last accessed May 2014.

[22] SANS.org. Port Details: Port 1053. SANS Internet Storm Center informational web site. 2013. http://isc.sans.org/port.html?port=1053. Last accessed May 2014.

[23] Speedguide.net. Port 1053 (tcp/udp). Informational web site. Speedguide.net. 2013. http://www.speedguide.net/port.php?port=1025. Last accessed May 2014.

[24] Speedguide.net. Port 1053 (tcp/udp). Informational web site. Speedguide.net. 2013. http://www.speedguide.net/port.php?port=136. Last accessed May 2014.

[25] About.com. Glossary of TCP Port and UDP Port Numbers – TCP/IP Port Number Glossary. Informational web site. About.com. 2013. http://compnetworking.about.com/od/tcpip/l/blports_gl100.htm. Last accessed May 2014.

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 2.3.1:

Table A.1: List of Volatility 2.3.1 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
dumpcerts	Dump RSA private and public SSL keys
dumpfiles	Extract memory mapped and cached files
envars	Display process environment variables
eventhooks	Print details on windows event hooks
evtlogs	Extract Windows Event Logs (XP/2003 only)

Plugin	**Capability (as per *Volatility --help* output)**
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
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)
hpakextract	Extract physical memory from an HPAK file
hpakinfo	Info on an HPAK file
idt	Display Interrupt Descriptor Table
iehistory	Reconstruct Internet Explorer cache / history
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
machoinfo	Dump Mach-O file format information
malfind	Find hidden and injected code
mbrparser	Scans for and parses potential Master Boot Records (MBRs)
memdump	Dump the addressable memory for a process

Plugin	**Capability (as per *Volatility --help* output)**
memmap	Print the memory map
messagehooks	List desktop and thread window message hooks
mftparser	Scans for and parses potential MFT entries
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
privs	Display process privileges
procexedump	Dump a process to an executable file sample
procmemdump	Dump a process to an executable memory sample
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)
shellbags	Prints ShellBags info
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

Plugin	**Capability (as per *Volatility --help* output)**
symlinkscan	Scan for symbolic link objects
thrdscan	Scan physical memory for _ETHREAD objects
threads	Investigate _ETHREAD and _KTHREADs
timeliner	Creates a timeline from various artifacts in memory
timers	Print kernel timers and associated module DPCs
unloadedmodules	Print list of unloaded modules
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
vboxinfo	Dump virtualbox information
vmwareinfo	Dump VMware VMSS/VMSN information
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

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 26 unique SHA1-filename matches as shown in the following table:

Table B.1: SHA1 hash vs. NSRL filenames for carved memory data files.

SHA1	Filename
048ABF0A35FFFEB7A43696EFB78290C2923F6069	icmp.dll
09105C886A83677E49CE6EF47F8CF1A047214AED	8.0.50727.762.policy
09105C886A83677E49CE6EF47F8CF1A047214AED	manifest.8.0.50727.762.68B7C6D9_1DF2_54C1_FF1F_C8B3B9A1E18E
09105C886A83677E49CE6EF47F8CF1A047214AED	ul_manifest.68B7C6D9_1DF2_54C1_FF1F_C8B3B9A1E18E
09105C886A83677E49CE6EF47F8CF1A047214AED	x1sw1o0k.9hi
09105C886A83677E49CE6EF47F8CF1A047214AED	z1sw1o0k.9hi
830D6459350DD1AB3B1F070135425A93395782B1	manifest.8.0.50727.762.74FD3CE6_2A8D_0E9C_FF1F_C8B3B9A1E18E
830D6459350DD1AB3B1F070135425A93395782B1	mfc80loc_man.7643D2EA_8E33_4EBC_B95C_9E5DF999A535
830D6459350DD1AB3B1F070135425A93395782B1	ul_manifest.74FD3CE6_2A8D_0E9C_FF1F_C8B3B9A1E18E
830D6459350DD1AB3B1F070135425A93395782B1	x86_Microsoft.VC80.MFCLOC_1fc8b3b9a1e18e3b_8.0.50727.762_x-ww_91481303.manifest
9537335B7EDA9AE3D1C125BE7BAC3161D5B853B8	COMCTL.MAN
9537335B7EDA9AE3D1C125BE7BAC3161D5B853B8	X86_POLICY.6.0.MICROSOFT.WINDOWS.COMMON-CONTROLS_6595B64144CCF1DF_6.0.2600.2180_X-WW_EB84B25E.MANIFEST
9537335B7EDA9AE3D1C125BE7BAC3161D5B853B8	comctl.man
A8139A5A5BCC413090176ECAF41510AA0FFBB987	Windows
C5B52B71F4C5F933815D7D606175EA0BB37DC548	CONTROLS.MAN
C5B52B71F4C5F933815D7D606175EA0BB37DC548	X86_MICROSOFT.WINDOWS.COMMON-CONTROLS_6595B64144CCF1DF_6.0.2600.2180_X-WW_A84F1FF9.MANIFEST
C5B52B71F4C5F933815D7D606175EA0BB37DC548	controls.man
D10440930CC994409E920D94C7C45F0405D60422	8.0.50727.762.policy
D10440930CC994409E920D94C7C45F0405D60422	manifest.8.0.50727.762.63E949F6_03BC_5C40_FF1F_C8B3B9A1E18E
D10440930CC994409E920D94C7C45F0405D60422	ul_manifest.63E949F6_03BC_5C40_FF1F_C8B3B9A1E18E

SHA1	Filename
D10440930CC994409E920D94C7C45F0405D60422	xxgs54we.kj4
D10440930CC994409E920D94C7C45F0405D60422	zxgs54we.kj4
DFC37F6C15612F7AB155E53A028A69FB5987199A	Program
F081561658705610ADAD4C30E757312491EDF9E0	8.0.50727.762.policy
F081561658705610ADAD4C30E757312491EDF9E0	manifest.8.0.50727.762.D2730D3F_3C41_5884_FF1F_C8B3B9A1E18E
F081561658705610ADAD4C30E757312491EDF9E0	ul_manifest.D2730D3F_3C41_5884_FF1F_C8B3B9A1E18E

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

In all, only one match was identified between the various scanners and it is identified below:

C.1 Avast

```
recup_dir.1/f0131152.exe        [infected by: Win32:Malware-gen] <- Match 1
recup_dir.1/f0009000.exe        [infected by: Win32:SwPatch [Wrm]]
```

C.2 AVG

```
recup_dir.2/f0050760.dll  Virus found Win32/Heur
recup_dir.1/f0197560.dll  Virus found Win32/Heur
recup_dir.1/f0131152.exe  Trojan horse Agent.APLY <- Match 1
recup_dir.1/f0217400.exe  Virus found Win32/Heur
```

C.3 BitDefender

```
recup_dir.1/f0024632.exe  infected: Gen:Variant.Boigy.1
recup.dir.1/f0130554.dll  infected: Gen:Trojan.Heur.GM.C006032100
recup.dir.1/f0136440.exe  infected: Gen:Variant.FakeAlert.47
```

C.4 Comodo

```
recup_dir.2/f0135640.dll ---> Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
recup_dir.1/f0209608.dll ---> Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
recup_dir.1/f0030696.exe ---> Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
recup_dir.1/f0170888.dll ---> Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
recup_dir.1/f0100240.exe ---> Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
recup_dir.1/f0144928.dll ---> Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
recup_dir.1/f0020984.exe ---> Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
recup_dir.1/f0204968.dll ---> Found Virus, Malware Name is TrojWare.Win32.FraudPack.P
```

C.5 F-Prot

```
recup_dir.1/f0131152.exe   <W32/Trojan3.QF (exact)> <- Match 1
```

C.6 McAfee

```
recup_dir.1/f0131152.exe ... Found the BackDoor-DTN trojan !!! <- Match 1
```

This page intentionally left blank.

Annex D Textual output for the malfind plugin

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

```
Process: csrss.exe Pid: 608 Address: 0x7f6f0000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x7f6f0000  c8 00 00 00 ff 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 ff01             INC DWORD [ECX]
0x7f6f0006 0000             ADD [EAX], AL
0x7f6f0008 ff               DB 0xff
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: winlogon.exe Pid: 632 Address: 0x2c930000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 4, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x2c930000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x2c930010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x2c930020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x2c930030  00 00 00 00 25 00 25 00 01 00 00 00 00 00 00 00   ....%.%.........

0x2c930000 0000             ADD [EAX], AL
0x2c930002 0000             ADD [EAX], AL
0x2c930004 0000             ADD [EAX], AL
0x2c930006 0000             ADD [EAX], AL
0x2c930008 0000             ADD [EAX], AL
0x2c93000a 0000             ADD [EAX], AL
0x2c93000c 0000             ADD [EAX], AL
```

```
0x2c93000e 0000             ADD [EAX], AL
0x2c930010 0000             ADD [EAX], AL
0x2c930012 0000             ADD [EAX], AL
0x2c930014 0000             ADD [EAX], AL
0x2c930016 0000             ADD [EAX], AL
0x2c930018 0000             ADD [EAX], AL
0x2c93001a 0000             ADD [EAX], AL
0x2c93001c 0000             ADD [EAX], AL
0x2c93001e 0000             ADD [EAX], AL
0x2c930020 0000             ADD [EAX], AL
0x2c930022 0000             ADD [EAX], AL
0x2c930024 0000             ADD [EAX], AL
0x2c930026 0000             ADD [EAX], AL
0x2c930028 0000             ADD [EAX], AL
0x2c93002a 0000             ADD [EAX], AL
0x2c93002c 0000             ADD [EAX], AL
0x2c93002e 0000             ADD [EAX], AL
0x2c930030 0000             ADD [EAX], AL
0x2c930032 0000             ADD [EAX], AL
0x2c930034 2500250001       AND EAX, 0x1002500
0x2c930039 0000             ADD [EAX], AL
0x2c93003b 0000             ADD [EAX], AL
0x2c93003d 0000             ADD [EAX], AL
0x2c93003f 00               DB 0x0

Process: winlogon.exe Pid: 632 Address: 0x37ec0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 4, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x37ec0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x37ec0010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x37ec0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x37ec0030  00 00 00 00 2b 00 2b 00 01 00 00 00 00 00 00 00   ....+.+.........

0x37ec0000 0000             ADD [EAX], AL
0x37ec0002 0000             ADD [EAX], AL
0x37ec0004 0000             ADD [EAX], AL
0x37ec0006 0000             ADD [EAX], AL
0x37ec0008 0000             ADD [EAX], AL
0x37ec000a 0000             ADD [EAX], AL
0x37ec000c 0000             ADD [EAX], AL
0x37ec000e 0000             ADD [EAX], AL
0x37ec0010 0000             ADD [EAX], AL
0x37ec0012 0000             ADD [EAX], AL
0x37ec0014 0000             ADD [EAX], AL
0x37ec0016 0000             ADD [EAX], AL
0x37ec0018 0000             ADD [EAX], AL
0x37ec001a 0000             ADD [EAX], AL
0x37ec001c 0000             ADD [EAX], AL
0x37ec001e 0000             ADD [EAX], AL
0x37ec0020 0000             ADD [EAX], AL
0x37ec0022 0000             ADD [EAX], AL
0x37ec0024 0000             ADD [EAX], AL
0x37ec0026 0000             ADD [EAX], AL
0x37ec0028 0000             ADD [EAX], AL
0x37ec002a 0000             ADD [EAX], AL
0x37ec002c 0000             ADD [EAX], AL
0x37ec002e 0000             ADD [EAX], AL
0x37ec0030 0000             ADD [EAX], AL
0x37ec0032 0000             ADD [EAX], AL
0x37ec0034 2b00             SUB EAX, [EAX]
0x37ec0036 2b00             SUB EAX, [EAX]
0x37ec0038 0100             ADD [EAX], EAX
0x37ec003a 0000             ADD [EAX], AL
0x37ec003c 0000             ADD [EAX], AL
0x37ec003e 0000             ADD [EAX], AL

Process: winlogon.exe Pid: 632 Address: 0x33470000
```

```
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 4, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x33470000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x33470010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x33470020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x33470030  00 00 00 00 29 00 29 00 01 00 00 00 00 00 00 00   ....).).........

0x33470000 0000             ADD [EAX], AL
0x33470002 0000             ADD [EAX], AL
0x33470004 0000             ADD [EAX], AL
0x33470006 0000             ADD [EAX], AL
0x33470008 0000             ADD [EAX], AL
0x3347000a 0000             ADD [EAX], AL
0x3347000c 0000             ADD [EAX], AL
0x3347000e 0000             ADD [EAX], AL
0x33470010 0000             ADD [EAX], AL
0x33470012 0000             ADD [EAX], AL
0x33470014 0000             ADD [EAX], AL
0x33470016 0000             ADD [EAX], AL
0x33470018 0000             ADD [EAX], AL
0x3347001a 0000             ADD [EAX], AL
0x3347001c 0000             ADD [EAX], AL
0x3347001e 0000             ADD [EAX], AL
0x33470020 0000             ADD [EAX], AL
0x33470022 0000             ADD [EAX], AL
0x33470024 0000             ADD [EAX], AL
0x33470026 0000             ADD [EAX], AL
0x33470028 0000             ADD [EAX], AL
0x3347002a 0000             ADD [EAX], AL
0x3347002c 0000             ADD [EAX], AL
0x3347002e 0000             ADD [EAX], AL
0x33470030 0000             ADD [EAX], AL
0x33470032 0000             ADD [EAX], AL
0x33470034 2900             SUB [EAX], EAX
0x33470036 2900             SUB [EAX], EAX
0x33470038 0100             ADD [EAX], EAX
0x3347003a 0000             ADD [EAX], AL
0x3347003c 0000             ADD [EAX], AL
0x3347003e 0000             ADD [EAX], AL

Process: winlogon.exe Pid: 632 Address: 0x71ee0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 4, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x71ee0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x71ee0010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x71ee0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x71ee0030  00 00 00 00 29 00 29 00 01 00 00 00 00 00 00 00   ....).).........

0x71ee0000 0000             ADD [EAX], AL
0x71ee0002 0000             ADD [EAX], AL
0x71ee0004 0000             ADD [EAX], AL
0x71ee0006 0000             ADD [EAX], AL
0x71ee0008 0000             ADD [EAX], AL
0x71ee000a 0000             ADD [EAX], AL
0x71ee000c 0000             ADD [EAX], AL
0x71ee000e 0000             ADD [EAX], AL
0x71ee0010 0000             ADD [EAX], AL
0x71ee0012 0000             ADD [EAX], AL
0x71ee0014 0000             ADD [EAX], AL
0x71ee0016 0000             ADD [EAX], AL
0x71ee0018 0000             ADD [EAX], AL
0x71ee001a 0000             ADD [EAX], AL
0x71ee001c 0000             ADD [EAX], AL
0x71ee001e 0000             ADD [EAX], AL
0x71ee0020 0000             ADD [EAX], AL
0x71ee0022 0000             ADD [EAX], AL
```

```
0x71ee0024 0000             ADD [EAX], AL
0x71ee0026 0000             ADD [EAX], AL
0x71ee0028 0000             ADD [EAX], AL
0x71ee002a 0000             ADD [EAX], AL
0x71ee002c 0000             ADD [EAX], AL
0x71ee002e 0000             ADD [EAX], AL
0x71ee0030 0000             ADD [EAX], AL
0x71ee0032 0000             ADD [EAX], AL
0x71ee0034 2900             SUB [EAX], EAX
0x71ee0036 2900             SUB [EAX], EAX
0x71ee0038 0100             ADD [EAX], EAX
0x71ee003a 0000             ADD [EAX], AL
0x71ee003c 0000             ADD [EAX], AL
0x71ee003e 0000             ADD [EAX], AL

Process: winlogon.exe Pid: 632 Address: 0x78850000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 4, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x78850000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x78850010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x78850020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x78850030  00 00 00 00 27 00 27 00 01 00 00 00 00 00 00 00   ....'.'.........
```

Annex E SHA1 and fuzzy hash matches

E.1 SHA1 duplicates for dlldump samples

This sub-annex lists all SHA1 duplicates established between various dlldump samples. The matches are as follows:

Table E.1: SHA1 duplicates established for dlldump samples (sorted by SHA1 hash).

SHA1 Hash	Dlldump
201a4aebd7be9c8e53760cb4a4b352256eeef938	module.452.4b5a980.20000000.dll
201a4aebd7be9c8e53760cb4a4b352256eeef938	module.468.10f7588.20000000.dll
201a4aebd7be9c8e53760cb4a4b352256eeef938	module.828.5c9f4d0.20000000.dll
49d2e77a2ea666a3abccfa7beda276edf14d753b	module.1088.61ef558.1000000.dll
49d2e77a2ea666a3abccfa7beda276edf14d753b	module.1148.6499b80.1000000.dll
49d2e77a2ea666a3abccfa7beda276edf14d753b	module.856.115b8d8.1000000.dll
49d2e77a2ea666a3abccfa7beda276edf14d753b	module.936.63c5560.1000000.dll
90e08ce6e5079100faef5184b8eaea99cec833f6	module.1724.4a065d0.77050000.dll
90e08ce6e5079100faef5184b8eaea99cec833f6	module.1732.10c3da0.77050000.dll
90e08ce6e5079100faef5184b8eaea99cec833f6	module.468.10f7588.77050000.dll
90e08ce6e5079100faef5184b8eaea99cec833f6	module.828.5c9f4d0.77050000.dll
d5604489e48b2913d31dc6a837c772f9e95e45ee	module.1724.4a065d0.76380000.dll
d5604489e48b2913d31dc6a837c772f9e95e45ee	module.468.10f7588.76380000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.1088.61ef558.769c0000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.1148.6499b80.769c0000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.1432.6945da0.769c0000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.1432.6945da0.76b40000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.1432.6945da0.773d0000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.1668.69d5b28.20000000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.1668.69d5b28.5d090000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.1668.69d5b28.769c0000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.1788.655fc88.20000000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.1788.655fc88.76bf0000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.1968.211ab28.773d0000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.1968.211ab28.77690000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.216.5f027e0.5cb70000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.216.5f027e0.773d0000.dll

SHA1 Hash	Dlldump
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.632.66f0978.5d090000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.676.6015020.5b860000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.676.6015020.769c0000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.688.5f47020.5d090000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.688.5f47020.662b0000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.688.5f47020.76080000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.688.5f47020.773d0000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.844.6384230.400000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.844.6384230.76780000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.844.6384230.78130000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.844.6384230.7c420000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.856.115b8d8.20000000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.856.115b8d8.769c0000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.856.115b8d8.773d0000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.936.63c5560.20000000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.936.63c5560.5d090000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.936.63c5560.769c0000.dll
da39a3ee5e6b4b0d3255bfef95601890afd80709	module.936.63c5560.773d0000.dll

E.2 Fuzzy hash matches between dlldump samples and carved memory files

This sub-annex lists all fuzzy hash matches established between the dlldump samples and the Carved Memory Files. The matches are as follows:

Table E.2: Fuzzy hash similarities established between dlldump samples and Carved Memory Files (sorted by %).

Dlldump	Carved Memory File	Match (in %)
module.1724.4a065d0.76380000.dll	f0125224_gdiext	(97)
module.468.10f7588.76380000.dll	f0125224_gdiext	(97)
module.1968.211ab28.769c0000.dll	f0089600.dll	(93)
module.1028.1122910.76d30000.dll	f0181368.dll	(79)
module.828.5c9f4d0.76d30000.dll	f0181368.dll	(75)
module.468.10f7588.76bb0000.dll	f0168728.dll	(66)
module.1432.6945da0.774e0000.dll	f0024720.exe	(55)

Dlldump	Carved Memory File	Match (in %)
module.452.4b5a980.77c00000.dll	f0148624.dll	(50)
module.1724.4a065d0.77f60000.dll	f0169224.dll	(49)
module.452.4b5a980.77f60000.dll	f0169224.dll	(47)
module.828.5c9f4d0.7c9c0000.dll	f0174120.dll	(47)
module.1028.1122910.755f0000.dll	f0215112.dll	(46)
module.1432.6945da0.20000000.dll	f0215112.dll	(44)
module.1724.4a065d0.76980000.dll	f0126432.dll	(44)
module.1724.4a065d0.77c00000.dll	f0148624.dll	(44)
module.452.4b5a980.76980000.dll	f0126432.dll	(44)
module.468.10f7588.77c00000.dll	f0148624.dll	(44)
module.828.5c9f4d0.774e0000.dll	f0172544.ttf	(44)
module.828.5c9f4d0.77c00000.dll	f0148624.dll	(44)
module.1724.4a065d0.75f60000.dll	f0112520.dll	(43)
module.432.4be97e8.7c800000.dll	f0169264.dll	(43)
module.468.10f7588.7c9c0000.dll	f0174120.dll	(43)
module.1028.1122910.77050000.dll	f0250200.exe	(41)
module.1028.1122910.77c00000.dll	f0148624.dll	(41)
module.1084.49c15f8.77c00000.dll	f0148624.dll	(41)
module.1432.6945da0.77c00000.dll	f0148624.dll	(41)
module.1724.4a065d0.774e0000.dll	f0172544.ttf	(41)
module.452.4b5a980.76bf0000.dll	f0024712.dll	(41)
module.1028.1122910.77050000.dll	f0179600.dll	(40)
module.1084.49c15f8.77c10000.dll	f0169256.dll	(40)
module.432.4be97e8.7c9c0000.dll	f0174120.dll	(40)
module.468.10f7588.77dd0000.dll	f0169528.dll	(40)
module.1028.1122910.50040000.dll	f0129960.dll	(38)
module.1028.1122910.77dd0000.dll	f0169264.dll	(38)
module.1084.49c15f8.77c10000.dll	f0174728.dll	(38)
module.1724.4a065d0.7c800000.dll	f0169264.dll	(38)
module.432.4be97e8.76780000.dll	f0186488.dll	(38)
module.432.4be97e8.77d40000.dll	f0169256.dll	(38)
module.452.4b5a980.76780000.dll	f0186488.dll	(38)
module.468.10f7588.76780000.dll	f0186488.dll	(38)
module.1028.1122910.75690000.dll	f0203192.dll	(36)

Dlldump	Carved Memory File	Match (in %)
module.1084.49c15f8.71aa0000.dll	f0179600.dll	(36)
module.1084.49c15f8.71aa0000.dll	f0250200.exe	(36)
module.1432.6945da0.77fe0000.dll	f0179600.dll	(36)
module.1432.6945da0.77fe0000.dll	f0250200.exe	(36)
module.452.4b5a980.10000000.dll	f0209656.dll	(36)
module.452.4b5a980.7c800000.dll	f0169264.dll	(36)
module.468.10f7588.7c800000.dll	f0169264.dll	(36)
module.828.5c9f4d0.75020000.dll	f0207296.reg	(36)
module.828.5c9f4d0.77e70000.dll	f0172032.dll	(36)
module.828.5c9f4d0.7c800000.dll	f0169264.dll	(36)
module.1088.61ef558.1000000.dll	f0179600.dll	(35)
module.1148.6499b80.1000000.dll	f0179600.dll	(35)
module.1432.6945da0.77c10000.dll	f0174728.dll	(35)
module.1432.6945da0.77dd0000.dll	f0169528.dll	(35)
module.1432.6945da0.7c800000.dll	f0169264.dll	(35)
module.1724.4a065d0.754d0000.dll	f0166736.exe	(35)
module.432.4be97e8.77f10000.dll	f0169528.dll	(35)
module.452.4b5a980.754d0000.dll	f0166736.exe	(35)
module.452.4b5a980.77dd0000.dll	f0169264.dll	(35)
module.452.4b5a980.77f10000.dll	f0169528.dll	(35)
module.468.10f7588.77d40000.dll	f0169256.dll	(35)
module.468.10f7588.77f10000.dll	f0169528.dll	(35)
module.608.66f0da0.77d40000.dll	f0169256.dll	(35)
module.856.115b8d8.1000000.dll	f0179600.dll	(35)
module.936.63c5560.1000000.dll	f0179600.dll	(35)
module.1028.1122910.75290000.dll	f0204464.dll	(33)
module.1088.61ef558.1000000.dll	f0250200.exe	(33)
module.1148.6499b80.1000000.dll	f0250200.exe	(33)
module.1724.4a065d0.71c10000.dll	f0157160.dll	(33)
module.1724.4a065d0.77dd0000.dll	f0169264.dll	(33)
module.468.10f7588.600a0000.dll	f0146704.dll	(33)
module.828.5c9f4d0.77dd0000.dll	f0169264.dll	(33)
module.828.5c9f4d0.77f10000.dll	f0169528.dll	(33)
module.856.115b8d8.1000000.dll	f0250200.exe	(33)

Dlldump	Carved Memory File	Match (in %)
module.936.63c5560.1000000.dll	f0250200.exe	(33)
module.1084.49c15f8.7c800000.dll	f0169264.dll	(32)
module.1724.4a065d0.77dd0000.dll	f0169528.dll	(32)
module.1724.4a065d0.77f10000.dll	f0169528.dll	(32)
module.468.10f7588.77dd0000.dll	f0169264.dll	(32)
module.828.5c9f4d0.77d40000.dll	f0169256.dll	(32)
module.828.5c9f4d0.77dd0000.dll	f0169528.dll	(32)
module.1028.1122910.71aa0000.dll	f0031400.dll	(30)
module.1028.1122910.77dd0000.dll	f0169528.dll	(30)
module.1432.6945da0.77dd0000.dll	f0169264.dll	(30)
module.1724.4a065d0.5d090000.dll	f0057504.dll	(30)
module.1724.4a065d0.5df10000.dll	f0241064.exe	(30)
module.1724.4a065d0.75f70000.dll	f0102544.dll	(30)
module.452.4b5a980.5d090000.dll	f0057504.dll	(30)
module.452.4b5a980.77dd0000.dll	f0169528.dll	(30)
module.1028.1122910.20000000.dll	f0069472.dll	(29)
module.1432.6945da0.77c10000.dll	f0169256.dll	(29)
module.1724.4a065d0.71b20000.dll	f0209656.dll	(29)
module.1724.4a065d0.76b40000.dll	f0055376.ttf	(29)
module.432.4be97e8.5d090000.dll	f0057504.dll	(29)
module.856.115b8d8.77dd0000.dll	f0169528.dll	(29)
module.1724.4a065d0.769c0000.dll	f0169176.dll	(27)
module.432.4be97e8.71aa0000.dll	f0031400.dll	(27)
module.452.4b5a980.71aa0000.dll	f0031400.dll	(27)
module.452.4b5a980.71b20000.dll	f0209656.dll	(27)
module.452.4b5a980.769c0000.dll	f0169176.dll	(27)
module.468.10f7588.77920000.dll	f0136384.dll	(27)
module.828.5c9f4d0.5d090000.dll	f0057504.dll	(27)
module.1028.1122910.7c800000.dll	f0170888.dll	(25)
module.1432.6945da0.742e0000.dll	f0024632.exe	(25)
module.432.4be97e8.400000.dll	f0180720.dll	(25)
module.468.10f7588.606b0000.dll	f0190280.dll	(25)
module.828.5c9f4d0.75290000.dll	f0204464.dll	(25)
module.1724.4a065d0.71cd0000.dll	f0064200.reg	(22)

Dlldump	Carved Memory File	Match (in %)
module.1724.4a065d0.754d0000.dll	f0016096.dll	(21)
module.452.4b5a980.754d0000.dll	f0016096.dll	(21)
module.1724.4a065d0.77be0000.dll	f0186104.exe	(19)
module.468.10f7588.77be0000.dll	f0186104.exe	(19)
module.828.5c9f4d0.77be0000.dll	f0186104.exe	(19)

E.3 Fuzzy hash similarities of 75% or greater between dlldump samples

This sub-annex lists all fuzzy hash matches established having a 90% or greater similarity between the various dlldump samples. The matches are as follows:

Table E.3: Fuzzy hash similarities established between dlldump samples (sorted by %).

Dlldump	Dlldump	Match (in %)
module.1084.49c15f8.77c00000.dll	module.1028.1122910.77c00000.dll	(100)
module.1148.6499b80.1000000.dll	module.1088.61ef558.1000000.dll	(100)
module.1432.6945da0.77c00000.dll	module.1028.1122910.77c00000.dll	(100)
module.1432.6945da0.77c00000.dll	module.1084.49c15f8.77c00000.dll	(100)
module.1732.10c3da0.77050000.dll	module.1724.4a065d0.77050000.dll	(100)
module.452.4b5a980.754d0000.dll	module.1724.4a065d0.754d0000.dll	(100)
module.452.4b5a980.77f10000.dll	module.432.4be97e8.77f10000.dll	(100)
module.468.10f7588.20000000.dll	module.452.4b5a980.20000000.dll	(100)
module.468.10f7588.4d4f0000.dll	module.1724.4a065d0.4d4f0000.dll	(100)
module.468.10f7588.76380000.dll	module.1724.4a065d0.76380000.dll	(100)
module.468.10f7588.76780000.dll	module.432.4be97e8.76780000.dll	(100)
module.468.10f7588.76f50000.dll	module.452.4b5a980.76f50000.dll	(100)
module.468.10f7588.77050000.dll	module.1724.4a065d0.77050000.dll	(100)
module.468.10f7588.77050000.dll	module.1732.10c3da0.77050000.dll	(100)
module.468.10f7588.773d0000.dll	module.432.4be97e8.773d0000.dll	(100)
module.828.5c9f4d0.20000000.dll	module.452.4b5a980.20000000.dll	(100)
module.828.5c9f4d0.20000000.dll	module.468.10f7588.20000000.dll	(100)
module.828.5c9f4d0.77050000.dll	module.1724.4a065d0.77050000.dll	(100)
module.828.5c9f4d0.77050000.dll	module.1732.10c3da0.77050000.dll	(100)
module.828.5c9f4d0.77050000.dll	module.468.10f7588.77050000.dll	(100)
module.828.5c9f4d0.773d0000.dll	module.432.4be97e8.773d0000.dll	(100)

Dlldump	Dlldump	Match (in %)
module.828.5c9f4d0.773d0000.dll	module.468.10f7588.773d0000.dll	(100)
module.856.115b8d8.1000000.dll	module.1088.61ef558.1000000.dll	(100)
module.856.115b8d8.1000000.dll	module.1148.6499b80.1000000.dll	(100)
module.936.63c5560.1000000.dll	module.1088.61ef558.1000000.dll	(100)
module.936.63c5560.1000000.dll	module.1148.6499b80.1000000.dll	(100)
module.936.63c5560.1000000.dll	module.856.115b8d8.1000000.dll	(100)
module.452.4b5a980.71aa0000.dll	module.432.4be97e8.71aa0000.dll	(99)
module.452.4b5a980.76780000.dll	module.432.4be97e8.76780000.dll	(99)
module.452.4b5a980.76980000.dll	module.1724.4a065d0.76980000.dll	(99)
module.452.4b5a980.76990000.dll	module.1724.4a065d0.76990000.dll	(99)
module.452.4b5a980.769c0000.dll	module.1724.4a065d0.769c0000.dll	(99)
module.452.4b5a980.76b20000.dll	module.1724.4a065d0.76b20000.dll	(99)
module.452.4b5a980.76c90000.dll	module.1724.4a065d0.76c90000.dll	(99)
module.452.4b5a980.76f50000.dll	module.1724.4a065d0.76f50000.dll	(99)
module.452.4b5a980.76f60000.dll	module.1724.4a065d0.76f60000.dll	(99)
module.452.4b5a980.76fd0000.dll	module.1724.4a065d0.76fd0000.dll	(99)
module.452.4b5a980.773d0000.dll	module.432.4be97e8.773d0000.dll	(99)
module.452.4b5a980.77b20000.dll	module.1724.4a065d0.77b20000.dll	(99)
module.468.10f7588.5ad70000.dll	module.432.4be97e8.5ad70000.dll	(99)
module.468.10f7588.71aa0000.dll	module.1724.4a065d0.71aa0000.dll	(99)
module.468.10f7588.71ab0000.dll	module.1724.4a065d0.71ab0000.dll	(99)
module.468.10f7588.76780000.dll	module.452.4b5a980.76780000.dll	(99)
module.468.10f7588.76b20000.dll	module.1724.4a065d0.76b20000.dll	(99)
module.468.10f7588.76b20000.dll	module.452.4b5a980.76b20000.dll	(99)
module.468.10f7588.76c90000.dll	module.1724.4a065d0.76c90000.dll	(99)
module.468.10f7588.76c90000.dll	module.452.4b5a980.76c90000.dll	(99)
module.468.10f7588.76f50000.dll	module.1724.4a065d0.76f50000.dll	(99)
module.468.10f7588.76fd0000.dll	module.1724.4a065d0.76fd0000.dll	(99)
module.468.10f7588.76fd0000.dll	module.452.4b5a980.76fd0000.dll	(99)
module.468.10f7588.773d0000.dll	module.452.4b5a980.773d0000.dll	(99)
module.468.10f7588.77b20000.dll	module.1724.4a065d0.77b20000.dll	(99)
module.468.10f7588.77b20000.dll	module.452.4b5a980.77b20000.dll	(99)
module.468.10f7588.77be0000.dll	module.1724.4a065d0.77be0000.dll	(99)
module.468.10f7588.77c00000.dll	module.1724.4a065d0.77c00000.dll	(99)

Dlldump	Dlldump	Match (in %)
module.828.5c9f4d0.5cb70000.dll	module.468.10f7588.5cb70000.dll	(99)
module.828.5c9f4d0.71aa0000.dll	module.1724.4a065d0.71aa0000.dll	(99)
module.828.5c9f4d0.71aa0000.dll	module.468.10f7588.71aa0000.dll	(99)
module.828.5c9f4d0.71ab0000.dll	module.1724.4a065d0.71ab0000.dll	(99)
module.828.5c9f4d0.71ab0000.dll	module.468.10f7588.71ab0000.dll	(99)
module.828.5c9f4d0.74ed0000.dll	module.1028.1122910.74ed0000.dll	(99)
module.828.5c9f4d0.76f60000.dll	module.1724.4a065d0.76f60000.dll	(99)
module.828.5c9f4d0.76f60000.dll	module.452.4b5a980.76f60000.dll	(99)
module.828.5c9f4d0.76fd0000.dll	module.1724.4a065d0.76fd0000.dll	(99)
module.828.5c9f4d0.76fd0000.dll	module.452.4b5a980.76fd0000.dll	(99)
module.828.5c9f4d0.76fd0000.dll	module.468.10f7588.76fd0000.dll	(99)
module.828.5c9f4d0.773d0000.dll	module.452.4b5a980.773d0000.dll	(99)
module.828.5c9f4d0.77a80000.dll	module.468.10f7588.77a80000.dll	(99)
module.828.5c9f4d0.77b20000.dll	module.1724.4a065d0.77b20000.dll	(99)
module.828.5c9f4d0.77b20000.dll	module.452.4b5a980.77b20000.dll	(99)
module.828.5c9f4d0.77b20000.dll	module.468.10f7588.77b20000.dll	(99)
module.828.5c9f4d0.77be0000.dll	module.1724.4a065d0.77be0000.dll	(99)
module.828.5c9f4d0.77be0000.dll	module.468.10f7588.77be0000.dll	(99)
module.828.5c9f4d0.77c00000.dll	module.1724.4a065d0.77c00000.dll	(99)
module.828.5c9f4d0.77c00000.dll	module.468.10f7588.77c00000.dll	(99)
module.452.4b5a980.77a80000.dll	module.1724.4a065d0.77a80000.dll	(97)
module.468.10f7588.5cb70000.dll	module.1724.4a065d0.5cb70000.dll	(97)
module.468.10f7588.6f880000.dll	module.1724.4a065d0.6f880000.dll	(97)
module.468.10f7588.76c30000.dll	module.452.4b5a980.76c30000.dll	(97)
module.828.5c9f4d0.5cb70000.dll	module.1724.4a065d0.5cb70000.dll	(97)
module.828.5c9f4d0.6f880000.dll	module.1724.4a065d0.6f880000.dll	(97)
module.828.5c9f4d0.6f880000.dll	module.468.10f7588.6f880000.dll	(97)
module.828.5c9f4d0.7c900000.dll	module.468.10f7588.7c900000.dll	(97)
module.452.4b5a980.5d090000.dll	module.432.4be97e8.5d090000.dll	(96)
module.468.10f7588.77c10000.dll	module.1724.4a065d0.77c10000.dll	(96)
module.828.5c9f4d0.5d090000.dll	module.1724.4a065d0.5d090000.dll	(96)
module.828.5c9f4d0.77f60000.dll	module.432.4be97e8.77f60000.dll	(96)
module.432.4be97e8.5d090000.dll	module.1724.4a065d0.5d090000.dll	(94)
module.452.4b5a980.5d090000.dll	module.1724.4a065d0.5d090000.dll	(94)

Dlldump	Dlldump	Match (in %)
module.468.10f7588.77f60000.dll	module.432.4be97e8.77f60000.dll	(94)
module.468.10f7588.7c900000.dll	module.452.4b5a980.7c900000.dll	(94)
module.828.5c9f4d0.5d090000.dll	module.432.4be97e8.5d090000.dll	(94)
module.828.5c9f4d0.5d090000.dll	module.452.4b5a980.5d090000.dll	(94)
module.1432.6945da0.77a80000.dll	module.1028.1122910.77a80000.dll	(93)
module.1732.10c3da0.76fd0000.dll	module.1724.4a065d0.76fd0000.dll	(93)
module.432.4be97e8.77f10000.dll	module.1724.4a065d0.77f10000.dll	(93)
module.452.4b5a980.71ab0000.dll	module.1724.4a065d0.71ab0000.dll	(93)
module.452.4b5a980.76fd0000.dll	module.1732.10c3da0.76fd0000.dll	(93)
module.452.4b5a980.77c10000.dll	module.1724.4a065d0.77c10000.dll	(93)
module.452.4b5a980.77f10000.dll	module.1724.4a065d0.77f10000.dll	(93)
module.468.10f7588.71ab0000.dll	module.452.4b5a980.71ab0000.dll	(93)
module.468.10f7588.76fd0000.dll	module.1732.10c3da0.76fd0000.dll	(93)
module.468.10f7588.77a80000.dll	module.452.4b5a980.77a80000.dll	(93)
module.828.5c9f4d0.71ab0000.dll	module.452.4b5a980.71ab0000.dll	(93)
module.828.5c9f4d0.76fd0000.dll	module.1732.10c3da0.76fd0000.dll	(93)
module.828.5c9f4d0.77a80000.dll	module.452.4b5a980.77a80000.dll	(93)
module.828.5c9f4d0.77c10000.dll	module.452.4b5a980.77c10000.dll	(93)
module.828.5c9f4d0.77f60000.dll	module.468.10f7588.77f60000.dll	(93)
module.828.5c9f4d0.7c800000.dll	module.468.10f7588.7c800000.dll	(93)
module.432.4be97e8.76d60000.dll	module.1724.4a065d0.76d60000.dll	(91)
module.452.4b5a980.76c30000.dll	module.1724.4a065d0.76c30000.dll	(91)
module.828.5c9f4d0.7c900000.dll	module.452.4b5a980.7c900000.dll	(91)
module.452.4b5a980.5ad70000.dll	module.432.4be97e8.5ad70000.dll	(90)
module.452.4b5a980.782e0000.dll	module.432.4be97e8.782e0000.dll	(90)
module.468.10f7588.5ad70000.dll	module.452.4b5a980.5ad70000.dll	(90)
module.468.10f7588.76360000.dll	module.1724.4a065d0.76360000.dll	(90)
module.468.10f7588.77a80000.dll	module.1724.4a065d0.77a80000.dll	(90)
module.468.10f7588.77d40000.dll	module.432.4be97e8.77d40000.dll	(90)
module.828.5c9f4d0.76d30000.dll	module.1028.1122910.76d30000.dll	(90)
module.828.5c9f4d0.77920000.dll	module.468.10f7588.77920000.dll	(90)
module.828.5c9f4d0.77a80000.dll	module.1724.4a065d0.77a80000.dll	(90)
module.828.5c9f4d0.77d40000.dll	module.432.4be97e8.77d40000.dll	(90)
module.828.5c9f4d0.77d40000.dll	module.468.10f7588.77d40000.dll	(90)

Dlldump	Dlldump	Match (in %)
module.828.5c9f4d0.77f10000.dll	module.1724.4a065d0.77f10000.dll	(90)
module.452.4b5a980.76360000.dll	module.1724.4a065d0.76360000.dll	(88)
module.452.4b5a980.77c00000.dll	module.1724.4a065d0.77c00000.dll	(88)
module.452.4b5a980.77fe0000.dll	module.1724.4a065d0.77fe0000.dll	(88)
module.468.10f7588.76c30000.dll	module.1724.4a065d0.76c30000.dll	(88)
module.468.10f7588.77c00000.dll	module.452.4b5a980.77c00000.dll	(88)
module.468.10f7588.77e70000.dll	module.1724.4a065d0.77e70000.dll	(88)
module.468.10f7588.77f10000.dll	module.432.4be97e8.77f10000.dll	(88)
module.468.10f7588.77f10000.dll	module.452.4b5a980.77f10000.dll	(88)
module.828.5c9f4d0.769c0000.dll	module.468.10f7588.769c0000.dll	(88)
module.828.5c9f4d0.77c00000.dll	module.452.4b5a980.77c00000.dll	(88)
module.828.5c9f4d0.77c10000.dll	module.432.4be97e8.77c10000.dll	(88)
module.828.5c9f4d0.77e70000.dll	module.1724.4a065d0.77e70000.dll	(88)
module.828.5c9f4d0.77f10000.dll	module.432.4be97e8.77f10000.dll	(88)
module.828.5c9f4d0.77f10000.dll	module.452.4b5a980.77f10000.dll	(88)
module.452.4b5a980.77c10000.dll	module.432.4be97e8.77c10000.dll	(86)
module.468.10f7588.50640000.dll	module.1028.1122910.50640000.dll	(86)
module.468.10f7588.73000000.dll	module.452.4b5a980.73000000.dll	(86)
module.468.10f7588.77c10000.dll	module.452.4b5a980.77c10000.dll	(86)
module.468.10f7588.77f10000.dll	module.1724.4a065d0.77f10000.dll	(86)
module.468.10f7588.7c800000.dll	module.1724.4a065d0.7c800000.dll	(86)
module.632.66f0978.77f10000.dll	module.1028.1122910.77f10000.dll	(86)
module.828.5c9f4d0.77c10000.dll	module.468.10f7588.77c10000.dll	(86)
module.1432.6945da0.7c900000.dll	module.1084.49c15f8.7c900000.dll	(85)
module.452.4b5a980.7c800000.dll	module.1432.6945da0.7c800000.dll	(85)
module.452.4b5a980.7c800000.dll	module.1724.4a065d0.7c800000.dll	(85)
module.828.5c9f4d0.5b860000.dll	module.452.4b5a980.5b860000.dll	(85)
module.828.5c9f4d0.76b40000.dll	module.468.10f7588.76b40000.dll	(85)
module.828.5c9f4d0.7c9c0000.dll	module.468.10f7588.7c9c0000.dll	(85)
module.1432.6945da0.77c10000.dll	module.1028.1122910.77c10000.dll	(83)
module.1724.4a065d0.7c800000.dll	module.1432.6945da0.7c800000.dll	(83)
module.432.4be97e8.77e70000.dll	module.1084.49c15f8.77e70000.dll	(83)
module.468.10f7588.77120000.dll	module.432.4be97e8.77120000.dll	(83)
module.828.5c9f4d0.769c0000.dll	module.1724.4a065d0.769c0000.dll	(83)

Dlldump	Dlldump	Match (in %)
module.828.5c9f4d0.769c0000.dll	module.452.4b5a980.769c0000.dll	(83)
module.828.5c9f4d0.76f20000.dll	module.1084.49c15f8.76f20000.dll	(83)
module.828.5c9f4d0.77920000.dll	module.452.4b5a980.77920000.dll	(83)
module.828.5c9f4d0.77c10000.dll	module.1724.4a065d0.77c10000.dll	(83)
module.828.5c9f4d0.7c900000.dll	module.1432.6945da0.7c900000.dll	(83)
module.1432.6945da0.73000000.dll	module.1084.49c15f8.73000000.dll	(82)
module.1724.4a065d0.76d60000.dll	module.1028.1122910.76d60000.dll	(82)
module.1724.4a065d0.76e80000.dll	module.1028.1122910.76e80000.dll	(82)
module.1724.4a065d0.77c10000.dll	module.1028.1122910.77c10000.dll	(82)
module.432.4be97e8.7c900000.dll	module.1084.49c15f8.7c900000.dll	(82)
module.452.4b5a980.7c900000.dll	module.1028.1122910.7c900000.dll	(82)
module.452.4b5a980.7c900000.dll	module.1432.6945da0.7c900000.dll	(82)
module.468.10f7588.77c10000.dll	module.1028.1122910.77c10000.dll	(82)
module.468.10f7588.77c10000.dll	module.432.4be97e8.77c10000.dll	(82)
module.468.10f7588.7c800000.dll	module.1432.6945da0.7c800000.dll	(82)
module.468.10f7588.7c900000.dll	module.1028.1122910.7c900000.dll	(82)
module.468.10f7588.7c900000.dll	module.1432.6945da0.7c900000.dll	(82)
module.468.10f7588.7c900000.dll	module.432.4be97e8.7c900000.dll	(82)
module.632.66f0978.77d40000.dll	module.1028.1122910.77d40000.dll	(82)
module.828.5c9f4d0.774e0000.dll	module.468.10f7588.774e0000.dll	(82)
module.828.5c9f4d0.77d40000.dll	module.1084.49c15f8.77d40000.dll	(82)
module.828.5c9f4d0.77e70000.dll	module.468.10f7588.77e70000.dll	(82)
module.828.5c9f4d0.77f10000.dll	module.468.10f7588.77f10000.dll	(82)
module.828.5c9f4d0.7c800000.dll	module.1724.4a065d0.7c800000.dll	(82)
module.828.5c9f4d0.7c900000.dll	module.1028.1122910.7c900000.dll	(82)
module.432.4be97e8.77c10000.dll	module.1724.4a065d0.77c10000.dll	(80)
module.432.4be97e8.77d40000.dll	module.1084.49c15f8.77d40000.dll	(80)
module.432.4be97e8.7c800000.dll	module.1432.6945da0.7c800000.dll	(80)
module.452.4b5a980.71aa0000.dll	module.1724.4a065d0.71aa0000.dll	(80)
module.452.4b5a980.77920000.dll	module.1724.4a065d0.77920000.dll	(80)
module.452.4b5a980.7c900000.dll	module.432.4be97e8.7c900000.dll	(80)
module.468.10f7588.71aa0000.dll	module.452.4b5a980.71aa0000.dll	(80)
module.468.10f7588.77dd0000.dll	module.1432.6945da0.77dd0000.dll	(80)
module.468.10f7588.7c900000.dll	module.1084.49c15f8.7c900000.dll	(80)

Dlldump	Dlldump	Match (in %)
module.828.5c9f4d0.71aa0000.dll	module.452.4b5a980.71aa0000.dll	(80)
module.828.5c9f4d0.75690000.dll	module.1028.1122910.75690000.dll	(80)
module.828.5c9f4d0.77120000.dll	module.1724.4a065d0.77120000.dll	(80)
module.828.5c9f4d0.7c800000.dll	module.1432.6945da0.7c800000.dll	(80)
module.828.5c9f4d0.7c900000.dll	module.1084.49c15f8.7c900000.dll	(80)
module.828.5c9f4d0.7c900000.dll	module.432.4be97e8.7c900000.dll	(80)
module.1668.69d5b28.76d60000.dll	module.1028.1122910.76d60000.dll	(79)
module.1724.4a065d0.77dd0000.dll	module.1028.1122910.77dd0000.dll	(79)
module.432.4be97e8.71aa0000.dll	module.1724.4a065d0.71aa0000.dll	(79)
module.452.4b5a980.77c10000.dll	module.1432.6945da0.77c10000.dll	(79)
module.452.4b5a980.78130000.dll	module.432.4be97e8.78130000.dll	(79)
module.452.4b5a980.7c900000.dll	module.1084.49c15f8.7c900000.dll	(79)
module.468.10f7588.71aa0000.dll	module.432.4be97e8.71aa0000.dll	(79)
module.468.10f7588.769c0000.dll	module.1724.4a065d0.769c0000.dll	(79)
module.468.10f7588.769c0000.dll	module.452.4b5a980.769c0000.dll	(79)
module.468.10f7588.76b40000.dll	module.452.4b5a980.76b40000.dll	(79)
module.468.10f7588.7c800000.dll	module.432.4be97e8.7c800000.dll	(79)
module.608.66f0da0.77d40000.dll	module.1028.1122910.77d40000.dll	(79)
module.828.5c9f4d0.71aa0000.dll	module.432.4be97e8.71aa0000.dll	(79)
module.828.5c9f4d0.77c10000.dll	module.1432.6945da0.77c10000.dll	(79)
module.1724.4a065d0.77120000.dll	module.1028.1122910.77120000.dll	(77)
module.432.4be97e8.7c800000.dll	module.1724.4a065d0.7c800000.dll	(77)
module.452.4b5a980.77c10000.dll	module.1028.1122910.77c10000.dll	(77)
module.452.4b5a980.77f60000.dll	module.1724.4a065d0.77f60000.dll	(77)
module.452.4b5a980.7c800000.dll	module.432.4be97e8.7c800000.dll	(77)
module.468.10f7588.77920000.dll	module.452.4b5a980.77920000.dll	(77)
module.468.10f7588.7c800000.dll	module.452.4b5a980.7c800000.dll	(77)
module.468.10f7588.7c9c0000.dll	module.432.4be97e8.7c9c0000.dll	(77)
module.828.5c9f4d0.77d40000.dll	module.1724.4a065d0.77d40000.dll	(77)
module.828.5c9f4d0.7c800000.dll	module.452.4b5a980.7c800000.dll	(77)
module.856.115b8d8.77dd0000.dll	module.1724.4a065d0.77dd0000.dll	(77)

E.4 Fuzzy hash matches between Moddump samples and carved memory files

This sub annex lists all fuzzy hash matches established between the Moddump samples and the carved memory files. The matches are as follows:

Table E.4: Fuzzy hash similarities established between Moddump samples carved memory files (sorted by %).

Moddump	Carved Memory File	Match (in %)
driver.fbf3a000.sys	f0220920_diskdump.sys	(100)
driver.fc053000.sys	f0134664.exe	(100)
driver.fc58b000.sys	f0129040.exe	(100)
driver.fc5cb000.sys	f0132272.exe	(100)
driver.fc5eb000.sys	f0134240.exe	(100)
driver.fc5fb000.sys	f0134816.exe	(100)
driver.fc67b000.sys	f0166736.exe	(100)
driver.fc6fb000.sys	f0219872_hidclass.sys	(100)
driver.fc763000.sys	f0126936.exe	(100)
driver.fc78b000.sys	f0130664.exe	(100)
driver.fc793000.sys	f0134352_tdi.sys	(100)
driver.fc79b000.sys	f0135024.exe	(100)
driver.fc8ab000.sys	f0260112_bootvid.dll	(100)
driver.fc93b000.sys	f0127792.exe	(100)
driver.fc967000.sys	f0136088.exe	(100)
driver.fc99f000.sys	f0260784_intelide.sys	(100)
driver.fc9a7000.sys	f0159576_usbd.sys	(100)
driver.fc9b1000.sys	f0163000_beep.sys	(100)
driver.fc9b3000.sys	f0163072_videosim.sys	(100)
driver.fcac9000.sys	f0049680_null.sys	(100)
driver.fcb25000.sys	f0241960_dxgthk.sys	(100)
driver.fcbef000.sys	f0131624_audstub.sys	(100)
driver.fbf42000.sys	f0219848.exe	(99)
driver.fc0c7000.sys	f0129560.dll	(99)
driver.fc170000.sys	f0164576_ws2ifsl.sys	(99)

Moddump	Carved Memory File	Match (in %)
driver.fc316000.sys	f0261208.exe	(99)
driver.fc5db000.sys	f0133744_raspppoe.sys	(99)
driver.fc7cb000.sys	f0163144.exe	(99)
driver.fc8af000.sys	f0260728.exe	(99)
driver.fc947000.sys	f0132728.exe	(99)
driver.fc99d000.sys	f0260504_WmiLib.sys	(99)
driver.fc9b5000.sys	f0163088_RDPCDD.SYS	(99)
driver.fc174000.sys	f0163208_rasacd.sys	(97)
driver.fc68b000.sys	f0216816.exe	(97)
driver.fc723000.sys	f0261512.exe	(97)
driver.fc9a5000.sys	f0134624_swenum.sys	(97)
driver.f3b41000.sys	f0164960.exe	(96)
driver.fc7eb000.sys	f0219960_hidparse.sys	(96)
driver.fc121000.sys	f0128008.exe	(94)
driver.fc943000.sys	f0130424.exe	(94)
driver.fc190000.sys	f0160528.exe	(93)
driver.fc9a3000.sys	f0126920.exe	(93)
driver.fc49b000.sys	f0260656_isapnp.sys	(91)
driver.fc8b3000.sys	f0260752.exe	(90)
driver.fc35b000.sys	f0260520_pci.sys	(88)
driver.fc7f3000.sys	f0240960_watchdog.sys	(88)
driver.fc8b7000.sys	f0261848.exe	(88)
driver.fc71b000.sys	f0260800_pciidex.sys	(86)
driver.fc2fe000.sys	f0261656.exe	(85)
driver.fc773000.sys	f0129160_vmx_svga.sys	(85)
driver.806ce000.sys	f0013936.dll	(80)
driver.fc98b000.sys	f0240504.exe	(80)
driver.fbf3e000.sys	f0219512.exe	(79)
driver.fc4ab000.sys	f0260856.exe	(79)
driver.fc4cb000.sys	f0262064.exe	(79)

Moddump	Carved Memory File	Match (in %)
driver.804d7000.sys	f0009912.exe	(77)
driver.fc0a3000.sys	f0130184.dll	(77)
driver.fc59b000.sys	f0130088.exe	(77)
driver.f3a85000.sys	f0216392.exe	(74)
driver.fc7ab000.sys	f0158984.exe	(74)
driver.fc2e6000.sys	f0261872.exe	(72)
driver.fc9af000.sys	f0162968.exe	(72)
driver.fc65b000.sys	f0164888_NETBIOS.SYS	(71)
driver.fc7bb000.sys	f0163016_vga.sys	(69)
driver.fc2b5000.sys	f0043352.exe	(68)
driver.fc55b000.sys	f0127880.exe	(68)
driver.fc9f7000.sys	f0019760.exe	(66)
driver.fc61b000.sys	f0135568.dll	(65)
driver.fc8ab000.sys	f0000128_bootvid.dll	(65)
driver.fc64b000.sys	f0159456.exe	(63)
driver.f3aa6000.sys	f0165832.exe	(61)
driver.fc4eb000.sys	f0045384.exe	(61)
driver.f386d000.sys	f0075448_NDISUIO.SYS	(60)
driver.fc0fe000.sys	f0128712.dll	(60)
driver.fc783000.sys	f0129424.exe	(60)
driver.fc33c000.sys	f0260944.exe	(58)
driver.fc4db000.sys	f0042984.dll	(57)
driver.f3b15000.sys	f0165424.exe	(55)
driver.fc77b000.sys	f0128488.exe	(55)
driver.fbf82000.sys	f0134648.exe	(54)
driver.fbf36000.sys	f0220960.exe	(50)
driver.fc5ab000.sys	f0129424.exe	(50)
driver.fc6bb000.sys	f0219672.exe	(50)
driver.fc7a3000.sys	f0135064.exe	(50)
driver.fc99b000.sys	f0261632.dll	(50)

Moddump	Carved Memory File	Match (in %)
driver.fc54b000.sys	f0128168.exe	(47)
driver.fc1c9000.sys	f0045168.exe	(44)
driver.fc7e3000.sys	f0220280.exe	(41)
driver.fbf82000.sys	f0134080.exe	(38)
driver.bf9d3000.sys	f0245136.dll	(36)
driver.fc0fe000.sys	f0127880.exe	(36)
driver.fc29e000.sys	f0042984.dll	(36)
driver.fc0ea000.sys	f0129208.exe	(35)
driver.fc0ea000.sys	f0128712.dll	(32)
driver.fc08c000.sys	f0132504.exe	(30)
driver.fc08c000.sys	f0132888.dll	(30)
driver.fc56b000.sys	f0128488.exe	(27)
driver.fc67b000.sys	f0016096.dll	(25)
driver.fc7e3000.sys	f0219592.exe	(25)
driver.f2ef5000.sys	f0050976.dll	(24)
driver.fc5ab000.sys	f0130554.dll	(21)

E.5 Fuzzy hash matches between Memdump samples carved memory files

This sub annex lists all fuzzy hash matches established between the Memdump samples and the carved memory files. The matches are as follows:

Table E.5: Fuzzy hash similarities established between Memdump samples and Carved Memory Files (sorted by %).

Memdump	Carved Memory File	Match (in %)
544.dmp	4.dmp	(94)
468.dmp	1084.dmp	(90)
1088.dmp	1084.dmp	(88)
1148.dmp	1084.dmp	(88)
1732.dmp	1084.dmp	(88)
608.dmp	1084.dmp	(88)
828.dmp	1084.dmp	(88)
856.dmp	1084.dmp	(88)

Memdump	Carved Memory File	Match (in %)
468.dmp	1732.dmp	(88)
608.dmp	1732.dmp	(88)
676.dmp	216.dmp	(88)
888.dmp	216.dmp	(88)
608.dmp	432.dmp	(88)
676.dmp	468.dmp	(88)
888.dmp	468.dmp	(88)
216.dmp	1084.dmp	(86)
432.dmp	1084.dmp	(86)
4.dmp	1084.dmp	(86)
632.dmp	1084.dmp	(86)
676.dmp	1084.dmp	(86)
888.dmp	1084.dmp	(86)
216.dmp	1732.dmp	(86)
888.dmp	1732.dmp	(86)
632.dmp	432.dmp	(86)
608.dmp	468.dmp	(86)
632.dmp	468.dmp	(86)
888.dmp	608.dmp	(86)
676.dmp	632.dmp	(86)
1668.dmp	1084.dmp	(85)
1788.dmp	1084.dmp	(85)
1968.dmp	1084.dmp	(85)
468.dmp	1088.dmp	(85)
676.dmp	1148.dmp	(85)
856.dmp	1148.dmp	(85)
1788.dmp	1732.dmp	(85)
856.dmp	1968.dmp	(85)
468.dmp	216.dmp	(85)
828.dmp	468.dmp	(85)
632.dmp	608.dmp	(85)
856.dmp	676.dmp	(85)
452.dmp	1084.dmp	(83)
544.dmp	1084.dmp	(83)

Memdump	Carved Memory File	Match (in %)
688.dmp	1084.dmp	(83)
1732.dmp	1088.dmp	(83)
608.dmp	1088.dmp	(83)
688.dmp	1088.dmp	(83)
1432.dmp	1148.dmp	(83)
216.dmp	1148.dmp	(83)
468.dmp	1148.dmp	(83)
4.dmp	1148.dmp	(83)
936.dmp	1148.dmp	(83)
452.dmp	1732.dmp	(83)
676.dmp	1732.dmp	(83)
468.dmp	1788.dmp	(83)
608.dmp	216.dmp	(83)
632.dmp	216.dmp	(83)
452.dmp	432.dmp	(83)
468.dmp	432.dmp	(83)
888.dmp	432.dmp	(83)
468.dmp	452.dmp	(83)
608.dmp	452.dmp	(83)
632.dmp	452.dmp	(83)
4.dmp	468.dmp	(83)
676.dmp	608.dmp	(83)
828.dmp	608.dmp	(83)
888.dmp	632.dmp	(83)
856.dmp	688.dmp	(83)
1432.dmp	1084.dmp	(82)
844.dmp	1084.dmp	(82)
1148.dmp	1088.dmp	(82)
1668.dmp	1088.dmp	(82)
1788.dmp	1088.dmp	(82)
1968.dmp	1088.dmp	(82)
856.dmp	1088.dmp	(82)
1668.dmp	1148.dmp	(82)
1732.dmp	1148.dmp	(82)

Memdump	Carved Memory File	Match (in %)
432.dmp	1148.dmp	(82)
608.dmp	1148.dmp	(82)
888.dmp	1148.dmp	(82)
1668.dmp	1432.dmp	(82)
216.dmp	1432.dmp	(82)
608.dmp	1668.dmp	(82)
828.dmp	1668.dmp	(82)
856.dmp	1668.dmp	(82)
1968.dmp	1732.dmp	(82)
432.dmp	1732.dmp	(82)
4.dmp	1732.dmp	(82)
632.dmp	1732.dmp	(82)
828.dmp	1732.dmp	(82)
1968.dmp	1788.dmp	(82)
608.dmp	1788.dmp	(82)
688.dmp	1788.dmp	(82)
828.dmp	1788.dmp	(82)
608.dmp	1968.dmp	(82)
828.dmp	1968.dmp	(82)
432.dmp	216.dmp	(82)
452.dmp	216.dmp	(82)
4.dmp	216.dmp	(82)
676.dmp	432.dmp	(82)
856.dmp	432.dmp	(82)
676.dmp	452.dmp	(82)
688.dmp	468.dmp	(82)
608.dmp	4.dmp	(82)
676.dmp	4.dmp	(82)
888.dmp	4.dmp	(82)
688.dmp	632.dmp	(82)
828.dmp	632.dmp	(82)
688.dmp	676.dmp	(82)
888.dmp	676.dmp	(82)
856.dmp	828.dmp	(82)

Memdump	Carved Memory File	Match (in %)
1148.dmp	1028.dmp	(80)
468.dmp	1028.dmp	(80)
216.dmp	1088.dmp	(80)
676.dmp	1088.dmp	(80)
828.dmp	1088.dmp	(80)
888.dmp	1088.dmp	(80)
1968.dmp	1148.dmp	(80)
452.dmp	1148.dmp	(80)
632.dmp	1148.dmp	(80)
1968.dmp	1432.dmp	(80)
432.dmp	1432.dmp	(80)
676.dmp	1432.dmp	(80)
216.dmp	1668.dmp	(80)
432.dmp	1668.dmp	(80)
468.dmp	1668.dmp	(80)
4.dmp	1668.dmp	(80)
632.dmp	1668.dmp	(80)
676.dmp	1668.dmp	(80)
688.dmp	1668.dmp	(80)
888.dmp	1668.dmp	(80)
688.dmp	1732.dmp	(80)
632.dmp	1788.dmp	(80)
676.dmp	1788.dmp	(80)
216.dmp	1968.dmp	(80)
452.dmp	1968.dmp	(80)
468.dmp	1968.dmp	(80)
688.dmp	1968.dmp	(80)
888.dmp	1968.dmp	(80)
688.dmp	216.dmp	(80)
828.dmp	216.dmp	(80)
856.dmp	216.dmp	(80)
4.dmp	432.dmp	(80)
828.dmp	452.dmp	(80)
544.dmp	468.dmp	(80)

Memdump	Carved Memory File	Match (in %)
856.dmp	468.dmp	(80)
632.dmp	4.dmp	(80)
856.dmp	4.dmp	(80)
688.dmp	608.dmp	(80)
856.dmp	608.dmp	(80)
856.dmp	632.dmp	(80)
828.dmp	676.dmp	(80)
844.dmp	676.dmp	(80)
936.dmp	676.dmp	(80)
828.dmp	688.dmp	(80)
888.dmp	688.dmp	(80)
888.dmp	828.dmp	(80)
888.dmp	856.dmp	(80)
1084.dmp	1028.dmp	(79)
1732.dmp	1028.dmp	(79)
888.dmp	1028.dmp	(79)
432.dmp	1088.dmp	(79)
4.dmp	1088.dmp	(79)
544.dmp	1088.dmp	(79)
632.dmp	1088.dmp	(79)
936.dmp	1088.dmp	(79)
828.dmp	1148.dmp	(79)
468.dmp	1432.dmp	(79)
608.dmp	1432.dmp	(79)
632.dmp	1432.dmp	(79)
856.dmp	1432.dmp	(79)
1732.dmp	1668.dmp	(79)
1968.dmp	1668.dmp	(79)
544.dmp	1732.dmp	(79)
856.dmp	1732.dmp	(79)
216.dmp	1788.dmp	(79)
856.dmp	1788.dmp	(79)
888.dmp	1788.dmp	(79)
432.dmp	1968.dmp	(79)

Memdump	Carved Memory File	Match (in %)
676.dmp	1968.dmp	(79)
544.dmp	216.dmp	(79)
828.dmp	432.dmp	(79)
688.dmp	452.dmp	(79)
888.dmp	452.dmp	(79)
828.dmp	4.dmp	(79)
608.dmp	544.dmp	(79)
676.dmp	544.dmp	(79)
888.dmp	544.dmp	(79)
856.dmp	844.dmp	(79)
936.dmp	856.dmp	(79)
1088.dmp	1028.dmp	(77)
432.dmp	1028.dmp	(77)
828.dmp	1028.dmp	(77)
936.dmp	1084.dmp	(77)
452.dmp	1088.dmp	(77)
1788.dmp	1148.dmp	(77)
544.dmp	1148.dmp	(77)
1732.dmp	1432.dmp	(77)
1788.dmp	1432.dmp	(77)
888.dmp	1432.dmp	(77)
1788.dmp	1668.dmp	(77)
432.dmp	1788.dmp	(77)
452.dmp	1788.dmp	(77)
4.dmp	1788.dmp	(77)
4.dmp	1968.dmp	(77)
632.dmp	1968.dmp	(77)
544.dmp	432.dmp	(77)
688.dmp	432.dmp	(77)
4.dmp	452.dmp	(77)
844.dmp	468.dmp	(77)
632.dmp	544.dmp	(77)
936.dmp	608.dmp	(77)
844.dmp	688.dmp	(77)

Memdump	Carved Memory File	Match (in %)
844.dmp	828.dmp	(77)
608.dmp	1028.dmp	(75)
632.dmp	1028.dmp	(75)
856.dmp	1028.dmp	(75)
936.dmp	1028.dmp	(75)
1432.dmp	1088.dmp	(75)
844.dmp	1088.dmp	(75)
688.dmp	1148.dmp	(75)
828.dmp	1432.dmp	(75)
452.dmp	1668.dmp	(75)
844.dmp	1668.dmp	(75)
844.dmp	1732.dmp	(75)
844.dmp	1788.dmp	(75)
936.dmp	216.dmp	(75)
544.dmp	452.dmp	(75)
856.dmp	452.dmp	(75)
936.dmp	468.dmp	(75)
688.dmp	4.dmp	(75)
688.dmp	544.dmp	(75)
828.dmp	544.dmp	(75)
844.dmp	544.dmp	(75)
844.dmp	608.dmp	(75)
844.dmp	632.dmp	(75)
936.dmp	632.dmp	(75)
936.dmp	688.dmp	(75)
936.dmp	828.dmp	(75)
936.dmp	888.dmp	(75)
1432.dmp	1028.dmp	(74)
216.dmp	1028.dmp	(74)
452.dmp	1028.dmp	(74)
676.dmp	1028.dmp	(74)
688.dmp	1028.dmp	(74)
844.dmp	1148.dmp	(74)
452.dmp	1432.dmp	(74)

Memdump	Carved Memory File	Match (in %)
688.dmp	1432.dmp	(74)
936.dmp	1432.dmp	(74)
544.dmp	1668.dmp	(74)
432.dmp	1724.dmp	(74)
936.dmp	1732.dmp	(74)
544.dmp	1788.dmp	(74)
544.dmp	1968.dmp	(74)
844.dmp	1968.dmp	(74)
936.dmp	1968.dmp	(74)
844.dmp	216.dmp	(74)
844.dmp	432.dmp	(74)
856.dmp	544.dmp	(74)
888.dmp	844.dmp	(74)
1668.dmp	1028.dmp	(72)
1788.dmp	1028.dmp	(72)
1968.dmp	1028.dmp	(72)
1724.dmp	1084.dmp	(72)
4.dmp	1432.dmp	(72)
544.dmp	1432.dmp	(72)
1724.dmp	1668.dmp	(72)
452.dmp	1724.dmp	(72)
468.dmp	1724.dmp	(72)
608.dmp	1724.dmp	(72)
632.dmp	1724.dmp	(72)
936.dmp	1788.dmp	(72)
936.dmp	432.dmp	(72)
844.dmp	452.dmp	(72)
844.dmp	4.dmp	(72)
1724.dmp	1088.dmp	(71)
1724.dmp	1432.dmp	(71)
844.dmp	1432.dmp	(71)
936.dmp	1668.dmp	(71)
1732.dmp	1724.dmp	(71)
1968.dmp	1724.dmp	(71)

Memdump	Carved Memory File	Match (in %)
216.dmp	1724.dmp	(71)
828.dmp	1724.dmp	(71)
936.dmp	452.dmp	(71)
1724.dmp	1028.dmp	(69)
4.dmp	1028.dmp	(69)
844.dmp	1028.dmp	(69)
1724.dmp	1148.dmp	(69)
856.dmp	1724.dmp	(69)
936.dmp	1724.dmp	(69)
936.dmp	4.dmp	(69)
936.dmp	544.dmp	(69)
1788.dmp	1724.dmp	(68)
888.dmp	1724.dmp	(68)
936.dmp	844.dmp	(68)
544.dmp	1028.dmp	(66)
676.dmp	1724.dmp	(66)
688.dmp	1724.dmp	(66)
844.dmp	1724.dmp	(60)

E.6 Fuzzy hash matches between Procexedump and dlldump samples

This sub annex lists all fuzzy hash matches established between the Procexedump and dlldump samples. The matches are as follows:

Table E.6: Fuzzy hash similarities established between Procexedump and dlldump samples (sorted by %).

Procexedump	Dlldump	Match (in %)
executable.1028.exe	module.1028.1122910.1000000.dll	(100)
executable.1084.exe	module.1084.49c15f8.400000.dll	(100)
executable.1088.exe	module.1088.61ef558.1000000.dll	(100)
executable.1088.exe	module.1148.6499b80.1000000.dll	(100)
executable.1088.exe	module.856.115b8d8.1000000.dll	(100)
executable.1088.exe	module.936.63c5560.1000000.dll	(100)
executable.1148.exe	module.1088.61ef558.1000000.dll	(100)
executable.1148.exe	module.1148.6499b80.1000000.dll	(100)

Procexedump	Dlldump	Match (in %)
executable.1148.exe	module.856.115b8d8.1000000.dll	(100)
executable.1148.exe	module.936.63c5560.1000000.dll	(100)
executable.1432.exe	module.1432.6945da0.1000000.dll	(100)
executable.1668.exe	module.1668.69d5b28.400000.dll	(100)
executable.1724.exe	module.1724.4a065d0.1000000.dll	(100)
executable.1732.exe	module.1732.10c3da0.400000.dll	(100)
executable.216.exe	module.216.5f027e0.1000000.dll	(100)
executable.432.exe	module.432.4be97e8.400000.dll	(100)
executable.452.exe	module.452.4b5a980.400000.dll	(100)
executable.468.exe	module.468.10f7588.400000.dll	(100)
executable.632.exe	module.632.66f0978.1000000.dll	(100)
executable.676.exe	module.676.6015020.1000000.dll	(100)
executable.688.exe	module.688.5f47020.1000000.dll	(100)
executable.828.exe	module.828.5c9f4d0.1000000.dll	(100)
executable.856.exe	module.1088.61ef558.1000000.dll	(100)
executable.856.exe	module.1148.6499b80.1000000.dll	(100)
executable.856.exe	module.856.115b8d8.1000000.dll	(100)
executable.856.exe	module.936.63c5560.1000000.dll	(100)
executable.936.exe	module.1088.61ef558.1000000.dll	(100)
executable.936.exe	module.1148.6499b80.1000000.dll	(100)
executable.936.exe	module.856.115b8d8.1000000.dll	(100)
executable.936.exe	module.936.63c5560.1000000.dll	(100)
executable.1028.exe	module.1088.61ef558.1000000.dll	(66)
executable.1028.exe	module.1148.6499b80.1000000.dll	(66)
executable.1028.exe	module.856.115b8d8.1000000.dll	(66)
executable.1028.exe	module.936.63c5560.1000000.dll	(66)
executable.1088.exe	module.1028.1122910.1000000.dll	(66)
executable.1148.exe	module.1028.1122910.1000000.dll	(66)
executable.856.exe	module.1028.1122910.1000000.dll	(66)
executable.936.exe	module.1028.1122910.1000000.dll	(66)
executable.1732.exe	module.468.10f7588.400000.dll	(58)
executable.468.exe	module.1732.10c3da0.400000.dll	(58)
executable.1088.exe	module.1084.49c15f8.71aa0000.dll	(43)
executable.1148.exe	module.1084.49c15f8.71aa0000.dll	(43)

Procexedump	Dlldump	Match (in %)
executable.856.exe	module.1084.49c15f8.71aa0000.dll	(43)
executable.936.exe	module.1084.49c15f8.71aa0000.dll	(43)
executable.1088.exe	module.1028.1122910.77050000.dll	(38)
executable.1148.exe	module.1028.1122910.77050000.dll	(38)
executable.856.exe	module.1028.1122910.77050000.dll	(38)
executable.936.exe	module.1028.1122910.77050000.dll	(38)
executable.1088.exe	module.1432.6945da0.77fe0000.dll	(36)
executable.1148.exe	module.1432.6945da0.77fe0000.dll	(36)
executable.856.exe	module.1432.6945da0.77fe0000.dll	(36)
executable.936.exe	module.1432.6945da0.77fe0000.dll	(36)

E.7 Fuzzy hash matches between dlldump and Dumpfiles samples

This sub annex lists all fuzzy hash matches with a threshold of 70% or more established between the dlldump and Dumpfiles samples. The matches are as follows:

Table E.7: Fuzzy hash similarities established between dlldump and Dumpfiles samples (sorted by %).

Dlldump	Dumpfiles	Match (in %)
module.1088.61ef558.1000000.dll	file.856.0xff380eb8.img	(100)
module.1148.6499b80.1000000.dll	file.856.0xff380eb8.img	(100)
module.432.4be97e8.5d360000.dll	file.432.0xff385990.img	(100)
module.452.4b5a980.10000000.dll	file.452.0xff26ebc8.img	(100)
module.468.10f7588.76bb0000.dll	file.632.0x80fca668.img	(100)
module.688.5f47020.1000000.dll	file.688.0x80f2a300.img	(100)
module.828.5c9f4d0.71f80000.dll	file.828.0xff3ad008.img	(100)
module.828.5c9f4d0.76d30000.dll	file.1028.0xff398408.img	(100)
module.856.115b8d8.1000000.dll	file.856.0xff380eb8.img	(100)
module.936.63c5560.1000000.dll	file.856.0xff380eb8.img	(100)
module.1724.4a065d0.71aa0000.dll	file.632.0x80f70880.img	(99)
module.1724.4a065d0.72d20000.dll	file.632.0x80f0af30.img	(99)
module.1724.4a065d0.76380000.dll	file.1732.0xff38f220.img	(99)
module.1724.4a065d0.77b40000.dll	file.632.0x80f77d40.img	(99)
module.1724.4a065d0.77c00000.dll	file.632.0x80f5a118.img	(99)
module.452.4b5a980.76bf0000.dll	file.632.0x80f28ea0.img	(99)

Dlldump	Dumpfiles	Match (in %)
module.468.10f7588.600a0000.dll	file.1028.0xff3bf5d0.img	(99)
module.468.10f7588.71aa0000.dll	file.632.0x80f70880.img	(99)
module.468.10f7588.76380000.dll	file.1732.0xff38f220.img	(99)
module.468.10f7588.77c00000.dll	file.632.0x80f5a118.img	(99)
module.828.5c9f4d0.71aa0000.dll	file.632.0x80f70880.img	(99)
module.828.5c9f4d0.77c00000.dll	file.632.0x80f5a118.img	(99)
module.1724.4a065d0.5cb70000.dll	file.676.0xff3653d0.img	(97)
module.1724.4a065d0.71bf0000.dll	file.632.0x80ff00f0.img	(97)
module.1724.4a065d0.76980000.dll	file.1724.0xff3874d0.img	(97)
module.1724.4a065d0.76b40000.dll	file.632.0xff36a1e0.img	(97)
module.1724.4a065d0.77bd0000.dll	file.632.0xff284f30.img	(97)
module.1724.4a065d0.77be0000.dll	file.632.0xff3662b8.img	(97)
module.452.4b5a980.76980000.dll	file.1724.0xff3874d0.img	(97)
module.452.4b5a980.76990000.dll	file.1724.0xff25cb10.img	(97)
module.452.4b5a980.7c420000.dll	file.844.0xff380a00.img	(97)
module.468.10f7588.5cb70000.dll	file.676.0xff3653d0.img	(97)
module.468.10f7588.77be0000.dll	file.632.0xff3662b8.img	(97)
module.828.5c9f4d0.5cb70000.dll	file.676.0xff3653d0.img	(97)
module.828.5c9f4d0.77be0000.dll	file.632.0xff3662b8.img	(97)
module.1432.6945da0.5e1f0000.dll	file.1432.0x80fbc2e0.img	(96)
module.1432.6945da0.75bb0000.dll	file.1432.0xff3aeef0.img	(96)
module.1724.4a065d0.72d10000.dll	file.632.0xff3b5270.img	(96)
module.1724.4a065d0.74ad0000.dll	file.1028.0x81027108.img	(96)
module.1724.4a065d0.76990000.dll	file.1724.0xff25cb10.img	(96)
module.1724.4a065d0.76c30000.dll	file.632.0x80fb0cf0.img	(96)
module.1724.4a065d0.76f60000.dll	file.632.0x80f296a0.img	(96)
module.1724.4a065d0.76fd0000.dll	file.632.0xff37cc00.img	(96)
module.1724.4a065d0.77f10000.dll	file.608.0x80f70b48.img	(96)
module.432.4be97e8.76780000.dll	file.844.0xff386870.img	(96)
module.452.4b5a980.76780000.dll	file.844.0xff386870.img	(96)
module.452.4b5a980.76f60000.dll	file.632.0x80f296a0.img	(96)
module.452.4b5a980.76fd0000.dll	file.632.0xff37cc00.img	(96)
module.452.4b5a980.771b0000.dll	file.1028.0x80f04f30.img	(96)
module.468.10f7588.76780000.dll	file.844.0xff386870.img	(96)

Dlldump	Dumpfiles	Match (in %)
module.468.10f7588.76fd0000.dll	file.632.0xff37cc00.img	(96)
module.828.5c9f4d0.74ef0000.dll	file.1788.0xff36a8a0.img	(96)
module.828.5c9f4d0.76f60000.dll	file.632.0x80f296a0.img	(96)
module.828.5c9f4d0.76fd0000.dll	file.632.0xff37cc00.img	(96)
module.1432.6945da0.1000000.dll	file.1432.0x80fb3240.img	(94)
module.1724.4a065d0.4d4f0000.dll	file.1028.0xff3af3b0.img	(94)
module.1724.4a065d0.5ad70000.dll	file.632.0xff3661e0.img	(94)
module.1724.4a065d0.5ba60000.dll	file.1724.0xff203008.img	(94)
module.1724.4a065d0.754d0000.dll	file.1028.0x80fad3a8.img	(94)
module.1724.4a065d0.76b20000.dll	file.632.0xff38a6f0.img	(94)
module.1724.4a065d0.773d0000.dll	file.632.0x80fb1108.img	(94)
module.1724.4a065d0.77920000.dll	file.632.0x81024b00.img	(94)
module.432.4be97e8.763b0000.dll	file.632.0x81002c68.img	(94)
module.452.4b5a980.754d0000.dll	file.1028.0x80fad3a8.img	(94)
module.452.4b5a980.76b20000.dll	file.632.0xff38a6f0.img	(94)
module.468.10f7588.4d4f0000.dll	file.1028.0xff3af3b0.img	(94)
module.468.10f7588.76b20000.dll	file.632.0xff38a6f0.img	(94)
module.828.5c9f4d0.5e760000.dll	file.1668.0x80fb1e38.img	(94)
module.1028.1122910.50640000.dll	file.1028.0xff1f66d0.img	(93)
module.1724.4a065d0.71c80000.dll	file.1432.0x80fef178.img	(93)
module.1724.4a065d0.71d40000.dll	file.1724.0x80fc1c60.img	(93)
module.1724.4a065d0.76c00000.dll	file.1028.0xff3b67c8.img	(93)
module.432.4be97e8.77f10000.dll	file.608.0x80f70b48.img	(93)
module.452.4b5a980.77f10000.dll	file.608.0x80f70b48.img	(93)
module.452.4b5a980.78130000.dll	file.844.0xff37dbc0.img	(93)
module.1724.4a065d0.76c90000.dll	file.632.0x80f5a880.img	(91)
module.432.4be97e8.10000000.dll	file.432.0xff3acf10.img	(91)
module.452.4b5a980.76c90000.dll	file.632.0x80f5a880.img	(91)
module.468.10f7588.76c90000.dll	file.632.0x80f5a880.img	(91)
module.1028.1122910.74ed0000.dll	file.1028.0xff3611c8.img	(90)
module.1028.1122910.76d30000.dll	file.1028.0xff398408.img	(90)
module.1028.1122910.76f60000.dll	file.632.0x80f63688.dat	(90)
module.1724.4a065d0.77a20000.dll	file.632.0xff25a008.img	(90)
module.1732.10c3da0.76fd0000.dll	file.632.0xff37cc00.img	(90)

Dlldump	Dumpfiles	Match (in %)
module.828.5c9f4d0.74ed0000.dll	file.1028.0xff3611c8.img	(90)
module.828.5c9f4d0.77f10000.dll	file.608.0x80f70b48.img	(90)
module.1724.4a065d0.6f880000.dll	file.676.0xff36a2b8.img	(88)
module.1968.211ab28.41000000.dll	file.1968.0x80f005d0.img	(88)
module.432.4be97e8.782e0000.dll	file.432.0xff388508.img	(88)
module.452.4b5a980.400000.dll	file.452.0xff281b10.img	(88)
module.452.4b5a980.77c00000.dll	file.632.0x80f5a118.img	(88)
module.468.10f7588.6f880000.dll	file.676.0xff36a2b8.img	(88)
module.632.66f0978.1000000.dll	file.632.0x80ff40c0.img	(88)
module.828.5c9f4d0.6f880000.dll	file.676.0xff36a2b8.img	(88)
module.1028.1122910.5f740000.dll	file.1028.0x80efd5d0.img	(86)
module.1028.1122910.755f0000.dll	file.1028.0xff3a28c8.img	(86)
module.1028.1122910.76d60000.dll	file.632.0xff380640.img	(86)
module.1724.4a065d0.77c10000.dll	file.632.0x80f61110.img	(86)
module.452.4b5a980.76c30000.dll	file.632.0x80fb0cf0.img	(86)
module.468.10f7588.76c30000.dll	file.632.0x80fb0cf0.img	(86)
module.468.10f7588.77f10000.dll	file.608.0x80f70b48.img	(86)
module.1724.4a065d0.75f70000.dll	file.1724.0x80f67e68.img	(85)
module.1724.4a065d0.75f70000.dll	file.1724.0xff239fa0.dat	(85)
module.452.4b5a980.782e0000.dll	file.432.0xff388508.img	(85)
module.828.5c9f4d0.692c0000.dll	file.1968.0x80faa8b0.img	(85)
module.1028.1122910.762c0000.dll	file.1028.0xff36a550.img	(83)
module.1724.4a065d0.5d090000.dll	file.632.0x80f2eca0.img	(83)
module.1724.4a065d0.7c900000.dll	file.4.0x80f49148.img	(83)
module.432.4be97e8.400000.dll	file.432.0x80fb4f30.img	(83)
module.432.4be97e8.5d090000.dll	file.632.0x80f2eca0.img	(83)
module.452.4b5a980.5d090000.dll	file.632.0x80f2eca0.img	(83)
module.468.10f7588.77c10000.dll	file.632.0x80f61110.img	(83)
module.828.5c9f4d0.5d090000.dll	file.632.0x80f2eca0.img	(83)
module.1028.1122910.76f50000.dll	file.632.0xff37e6e0.img	(82)
module.1668.69d5b28.400000.dll	file.1668.0xff3bcbc8.img	(82)
module.468.10f7588.50640000.dll	file.1028.0xff1f66d0.img	(82)
module.828.5c9f4d0.1000000.dll	file.828.0xff245990.img	(82)
module.1432.6945da0.75150000.dll	file.1028.0xff3beee0.img	(80)

Dlldump	Dumpfiles	Match (in %)
module.1724.4a065d0.1000000.dll	file.1724.0xff25aae0.img	(80)
module.432.4be97e8.78130000.dll	file.844.0xff37dbc0.img	(80)
module.452.4b5a980.71aa0000.dll	file.632.0x80f70880.img	(80)
module.468.10f7588.400000.dll	file.1732.0x80f73380.img	(80)
module.432.4be97e8.71aa0000.dll	file.632.0x80f70880.img	(79)
module.452.4b5a980.77c10000.dll	file.632.0x80f61110.img	(79)
module.468.10f7588.50940000.dll	file.1732.0xff23ad08.img	(79)
module.828.5c9f4d0.77c10000.dll	file.632.0x80f61110.img	(79)
module.1028.1122910.75690000.dll	file.1028.0xff3612c0.img	(77)
module.1668.69d5b28.76d60000.dll	file.632.0xff380640.img	(77)
module.1724.4a065d0.71ad0000.dll	file.1028.0x80fb72f0.img	(77)
module.452.4b5a980.77920000.dll	file.632.0x81024b00.img	(77)
module.1028.1122910.77e70000.dll	file.608.0x81023970.img	(75)
module.828.5c9f4d0.75020000.dll	file.1028.0x80f07958.img	(75)
module.1028.1122910.77920000.dll	file.632.0x81024b00.img	(74)
module.1028.1122910.77c10000.dll	file.632.0x80f61110.img	(74)
module.452.4b5a980.71bf0000.dll	file.632.0x80ff00f0.img	(74)
module.1084.49c15f8.400000.dll	file.1084.0xff3af0d8.img	(72)
module.1432.6945da0.742e0000.dll	file.1432.0x80f64290.img	(72)
module.1724.4a065d0.75f60000.dll	file.1724.0x81000d00.img	(72)
module.1724.4a065d0.76f50000.dll	file.632.0xff37e6e0.img	(72)
module.432.4be97e8.71ab0000.dll	file.632.0x80fb5578.img	(72)
module.432.4be97e8.77c10000.dll	file.632.0x80f61110.img	(72)
module.452.4b5a980.5ad70000.dll	file.632.0xff3661e0.img	(72)
module.452.4b5a980.76f50000.dll	file.632.0xff37e6e0.img	(72)
module.468.10f7588.75260000.dll	file.1028.0xff3c02e8.img	(72)
module.468.10f7588.76f50000.dll	file.632.0xff37e6e0.img	(72)
module.676.6015020.1000000.dll	file.676.0x81023c50.img	(72)
module.828.5c9f4d0.75690000.dll	file.1028.0xff3612c0.img	(72)
module.1028.1122910.662b0000.dll	file.688.0xff397598.img	(71)
module.1028.1122910.75310000.dll	file.1028.0xff362978.img	(71)
module.1724.4a065d0.74af0000.dll	file.1724.0xff128b70.img	(71)
module.1724.4a065d0.769c0000.dll	file.632.0x80efa2b0.img	(71)
module.1724.4a065d0.76d60000.dll	file.632.0xff380640.img	(71)

Dlldump	Dumpfiles	Match (in %)
module.452.4b5a980.769c0000.dll	file.632.0x80efa2b0.img	(71)

Annex F Commonly used registry keys in a typical malware infection

F.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 the evaluation of Tigger/Syzor:

- Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache
- Control Panel\Desktop
- Control Panel\Desktop\ScreenSaveActive
- ControlSet001\Enum\Root\LEGACY_TMRYQYRZNR2.SYS\0000
- ControlSet001\Safeboot\Minimal\TMRYQYRZNR2.SYS
- ControlSet001\Safeboot\Network\TMRYQYRZNR2.SYS
- ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\AuthorizedApplications\List
- CurrentControlSet\Services\TMRYQYRZNR2.SYS
- 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 tigger.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 E.1, used in conjunction with various command line tools including *cat*, *awk* and *sed*.

F.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.

Carbone, Richard. Malware memory analysis for non-specialists: Investigating publicly available memory image for the Stuxnet worm. Scientific Report. Defence Research and Development Canada, DRDC-RDDC-2013-R1. November 2013.

Volatility. CommandReference: Example usage cases and output for Volatility 2.0 commands. Onlinecommand 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
CAF	Canadian Armed Forces
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 Sometimes known as fuzzy hash or *ssdeep* hash
DCOM	Distribute Component Object Model
DLL	Dynamically Loaded Library
DND	Department of National Defence
DRDC	Defence Research and Development Canada
DRDKIM	Director Research and Development Knowledge and Information Management
EDT	Eastern Daylight Time
EXT4	Fourth Extended Filesystem
FAC	Forces Armés Canadiens
FOSS	Free and Open Source Software
FTP	File Transfer Protocol
GICT	Groupe intégré de la criminalité technologique
GRC	Gendarmerie Royale du Canada
GRE	General Routing Encapsulation
GUI	Graphical User Interface
HKCU	HKEY_LOCAL_USER
HKLM	HKEY_LOCAL_MACHINE
ID	Identification
IIS	Internet Information Services

IP	Internet Protocol
ITCU	Integrated Technological Crime Unit
MAC	Mandatory Access Control
MD5	Message Digest Algorithm 5
MiB	Mebibyte
N/A	Not Available
NAT	Network Address Translation
NFS	Network File System
NIST	National Institute of Standards and Technology
NSRL	National Software Reference Library
NTP	Network Time Protocol
PAE	Physical Address Extension
PE/PE32/PE64	Portable Executable/Portable Executable 32-bit/Portable Executable 64-bit
PID	Process ID
POP3	Post Office Protocol version 3
PPID	Parent Process ID
R&D	Research and 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
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.

Portable Executable

This is primarily a x86-based DOS/Windows executable file format, although certain lesser used operating systems and processor architectures also use and support this format. Typical Windows executables are 32 and 64-bit in nature although older DOS/Windows systems were 16-bit. The format is used for programs, applications, libraries, operating system kernel and device drivers.

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.

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.

This page intentionally left blank.

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.)

DRDC Valcartier Research Centre
2459 Pie-XI Blvd,
North Quebec (Quebec) G3J 1X5
Canada

2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.)

UNCLASSIFIED

2b. CONTROLLED GOODS

(NON-CONTROLLED GOODS)
DMC A
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 Tigger Trojan horse

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.)

June 2014

6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)

142

6b. NO. OF REFS (Total cited in document.)

25

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.)

DRDC Valcartier Research Centre 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.)

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.)

DRDC-RDDC-2014-R28

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 a computer forensic investigator can effectively analyse an infected Windows memory dump. The author investigates how to perform 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's previously proposed memory-specific methodology could be of aid to fellow novice memory analysts. The author examines how Volatility can be used to find evidence and indicators of infection. This report is the fifth in this series concerning Windows malware-based memory analysis. It examines a memory image infected with the Tigger/Syzor Trojan horse.

Ce rapport examine comment un enquêteur en informatique judiciaire peut analyser de façon efficace une image mémoire Windows infectée. L'auteur étudie comment effectuer une telle analyse en utilisant Volatility ainsi que d'autres outils d'enquête, tels que des utilitaires de récupération de données et des logiciels antivirus. Volatility est un cadriciel populaire et évolutif à code source ouvert pour l'analyse de mémoire et pour lequel l'auteur avait précédemment proposé une méthodologie spécifique afin d'aider des collègues analystes novices. L'auteur examine comment Volatility peut être utilisé afin de trouver des preuves et des indicateurs d'infection. Ce rapport est le cinquième d'une série sur l'analyse de logiciels malveillants Windows basée sur la mémoire. Celui-ci porte sur une image mémoire infectée par le cheval de Troie Tigger/Syzor.

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.)

Anti-virus; Antivirus; Computer forensics; Digital forensics; Digital forensic investigations; Forensics; Infection; Malware; Memory analysis; Memory forensics; Memory image; Rootkit; Scanners; Syzor; Tigger; Trojan horse; Virus scanner; Volatility; Windows