Malware memory analysis of the Jynx2 Linux rootkit (Part 1)

Investigating a publicly available Linux rootkit using the Volatility memory analysis framework

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-R176
October 2014

Abstract

This report is the second in a series that will examine Linux Volatility-specific memory malware-based analysis techniques. Windows-based malware memory analysis techniques were analysed in a previous series. Unlike these Windows-based reports, some of the techniques described therein are not applicable to Linux-based analyses including data carving and anti-virus scanning. Thus, with minimal use of scanner-based technologies, the author will demonstrate what to look for while conducting Linux-specific Volatility-based investigations. Each investigation consists of an infected memory image and its accompanying Volatility memory profile that will be used to examine a different open source rootkit. Some of the rootkits are user-land while others are kernel-based. Rootkits were chosen over Trojans, worms and viruses as rootkits tend to be more sophisticated. This specific investigation examines the Jynx2 rootkit. However, this analysis is broken into two parts. The first examines a system infected with Jynx2 but which has not yet loaded any new processes with the infected library/rootkit while the second examines a system completely infected by Jynx2. It is hoped that through the proper application of various Volatility plugins combined with an in-depth knowledge of the Linux operating system, these case studies will provide guidance to other investigators in their own analyses.

Significance to defence and security

Canadian Armed Forces' (CAF) networks are a choice target for malware and directed attacks. This series of reports will provide junior and senior incident handlers alike with the necessary knowledge to investigate and mitigate complex attacks using only a memory image and a functional knowledge of the Linux operating system. As Linux continues to play a more important role in IT and the data centres of the Government of Canada and National Defence, some of these systems will invariably become infected. Thus, when this happens and when analysts and incident handlers have to intervene, it is hoped that these reports will have helped them to prepare for just such an occasion.

Résumé

Ce rapport est le second d’une série examinant les techniques spécifiques d’analyse de logiciels malveillants en mémoire sous Linux à l’aide de l’outil Volatility. Les techniques d’analyse de logiciels malveillants en mémoire pour Windows ont été décrites dans des rapports précédents. Cependant, certaines de ces techniques, telles que la récupération de données et le balayage d’antivirus ne s’appliquent pas aux analyses sous Linux. Par conséquent, avec une utilisation minimale des technologies de balayage, l’auteur démontrera ce qu’il faut rechercher lorsqu’on effectue des investigations spécifiques à Linux avec Volatility. Chaque investigation consiste en une image mémoire infectée, accompagnée de son profile mémoire Volatility, et examinera un programme malveillant furtif à code source ouvert différent. Certains seront en mode utilisateur tandis que d’autres seront en mode noyau. Les programmes malveillants furtifs ont été préférés aux chevaux de Troie, vers et virus, car ils ont tendance à être plus sophistiqués. La présente investigation examine spécifiquement le programme malveillant furtif Jynx2. Cependant, cette analyse est divisée en deux parties. La première examine un système infecté par Jynx2 mais qui n'a pas encore chargé de nouveau processus qui utilise la librairie infectée tandis que la seconde examine un système complètement infecté par Jynx2. Il est souhaité qu’avec une utilisation adéquate de différents plugiciels Volatility et d’une connaissance approfondie du système d’exploitation Linux, ces études de cas fourniront des conseils à d’autres enquêteurs pour leurs propres analyses.

Importance pour la défense et la sécurité

Les réseaux des Forces armées canadiennes (FAC) sont une cible de choix pour les logiciels malveillants et les attaques dirigées. Cette série de rapports fournira aux analystes en réponse aux incidents, aussi bien juniors que séniors, toute la connaissance requise pour investiguer et mitiger des attaques complexes en utilisant seulement une image de la mémoire et une connaissance fonctionnelle du système d’exploitation Linux. Comme Linux joue un rôle de plus en plus important dans les TI et les centres de données du gouvernement du Canada et de la Défense nationale, certains de ces systèmes deviendront invariablement infectés. Par conséquent, quand cela arrivera et que des analystes en réponses aux incidents auront à intervenir, nous espérons que ces rapports les auront aidés à se préparer à une telle occasion.

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. Martin Salois, Defence Scientist, for improvements to the beginning portions of this text and to Philippe Charland, Defence Scientist, for translating portions of this text.

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 20 64-bit Linux operating system. Unlike the various Windows infected memory case studies, neither anti-virus (AV) nor data carving techniques worked particularly well against Linux-based memory images. As such, the former is used minimally while the latter is not at all used in this report. Consequently, the methodology presented in this series of reports will be quite different from that presented in the Windows Volatility-based series of memory malware analyses.

It is important that the reader 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 for the analysis of a Linux-based memory malware infection.

In this report, the use of the words rootkit, infection and malware are used interchangeably.

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

Target audience

The target audience for this report is the computer forensic investigator who assesses suspect computer memory images for evidence of infection and the incident handler who is called on to assess or intervene in a possible malware infection. While previous reports were targeted at investigators and incident handlers working with Windows-based memory images and malware, this new series of reports will be directed at those who must analyse Linux malware-infected memory images.

The skills amassed by incident handlers and investigators alike while using Volatility to examine Windows memory images will be of some help. However, Linux and Windows are not the same and while there is commonality in the approach used by the author throughout both series of reports (Linux and Windows series, respectively), important differences are nevertheless apparent. This is because these operating systems are fundamentally different; therefore, in order to extract the maximum value from this report, the reader should have a working knowledge of Linux, basic system administration and software compilation.

Unlike the previous Windows-based reports, it was determined that Linux-specific memory analysis case studies and reports have been left woefully unexamined by the community, at least as of the time of this writing, hence prompting the author to write this case study and its subsequent follow-up studies.

Availability of Linux memory images and profiles

Although various Linux-based memory images are available from different publicly available sources, most notables among them those from SecondLook, no Volatility-compatible memory profiles were available without cost. However, because the author desired that this work be made available to the largest possible audience, he built his own virtual machines (VMs), created the appropriate memory profiles and compiled the various rootkits, infected the VMs and then dumped their memory.

Although SecondLook's memory images are freely available, its memory profiles are not – they must be purchased. Thus, if others want to validate the author's work or use his memory images and profiles to practice their craft, they will be made available.

The author will endeavour to ensure that his memory images and profiles will be made available to anyone requesting a copy, given the constraints below. The author can be contacted at val-forensics@drdc-rddc.gc.ca. Please state your name, organization, country and mailing address including additional contact information and one will be mailed to you within a reasonable delay. No PO Boxes will be accepted – commercial and government mailing addresses only.

However, countries listed on Canada's Export Control List (ECL) are automatically forfeit. Even though the various memory images and Volatility profiles are considered as NON-CONTROLLED GOODS and are for unclassified use and unlimited distribution, ECL-listed countries will NOT be considered. For more information concerning countries listed on Canada's ECL, please consult http://www.international.gc.ca/controls-controles/export-exportation/exp_ctr_handbook-manuel_ctr_exp-p2.aspx?lang=eng#d.

1 Background

1.1 Objective

The objective of this report is to examine how a computer forensic investigator/incident handler, without specialised computer memory or software reverse engineering skills, can successfully investigate a Linux-based memory image suspected of infection.

In order to successfully investigate such an image, this report will use an applied plugin-based approach as it uses demonstrable procedures that intermediate-level investigators and incident handlers can use as a basis for investigating suspected memory images.

The work carried out herein is based on the publicly available source code for the Jynx2 rootkit. This document is the second in a series of reports that examines Linux-based malware memory analysis, which in of itself is the first part of two reports. This specific report examines what to look for when working with LD_PRELOAD rootkits. Ultimately, these reports will provide a foundational framework that novice and experienced investigators alike can rely on for guidance when investigating infected Linux memory images.

1.2 Project support

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

1.3 Of potential use to

The results of this project may 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 law enforcement-related cyber investigation teams.

1.4 Jynx2 rootkit background

In contrast to the KBeast rootkit where no analyses or information was available concerning the rootkit beyond its author's claims, some information is available online concerning Jynx2 which includes [1][2][3][4]. Moreover, the rootkit can be augmented to include PAM hooking [5]. However, Jynx2 was not compiled or configured for such hooking although [5] readily explains how this is achieved.

As with the KBeast rootkit, no technical analysis was immediately available from the various AV vendors concerning the Jynx2 rootkit. What is mostly known about it comes from information made available by its author, Blackhat Academy. The rootkit's source code was released in March 2012. The rootkit's first iteration, Jynx, was released in October 2011.

Jynx2 is a userland rootkit, which in stark contrast to KBeast, was a LKM rootkit. This userland rootkit achieves its objectives via LD_PRELOAD, which is examined in [2]. However, this rootkit requires that an attacker already have access to the target system and have succeeded in acquiring root-level permission in order for the rootkit to suitably infect the underlying system.

According to the rootkit's author, it has the following capabilities [1]:

- Hiding from *netstat*;
- Hiding from *ps*/*top* and */proc*;
- File hiding;
- SSL connect *accept()* hook;
- Multi-factor authentication;
- Improved anti-removal features, and
- SUID Drop-shell with environment variable.

The rootkit has a configuration file, *config.h*, which allows an attacker to make certain customizations to the rootkit. However, as for the compilation of this rootkit, the current author made no changes to its configuration and instead accepted its default setup.

While a brief analysis of the source code by the current author indicates that these capabilities appear to be valid claims, the author has not verified these capabilities in-depth. The reader is free to do so at his leisure. However, there is no reason to believe these claims to be false. Unfortunately, insufficient information is available to determine which kernels the rootkit is capable of running on. Finally, rootkit compilation specifics are found in Section 1.7.

1.5 Information concerning the host and guest systems

1.5.1 Host system

The physical host system was a Dell Inspiron 17R 5737 laptop. It was equipped with an Intel i7-4500 1.80 GHz quad-core CPU (hyper-threading was not available for this CPU), 16 GB (15.6 GiB) RAM, 500 GB SATA hard drive, 8x DVD +/- RW and Intel HD300 video card. It also had various USB2/3 ports, WiFi and one mini-SD slot for additional connectivity but which were all disabled in the BIOS and were therefore unimportant for this particular work. The operating system running on the laptop was Fedora 20 x64, specifically kernel 3.15.3-200. Oracle VirtualBox 4.3.12 was installed and was found to be functioning correctly. Finally, VirtualBox 4.3.12 Extension Pack was installed atop VirtualBox.

1.5.2 Guest system (VM)

The Linux test virtual machine (VM) which was infected with Jynx2 was built atop Ubuntu 10.10 x86 and was installed from DVD media. The VM was allocated 1 CPU and 2 GiB RAM and the default Ubuntu VirtualBox parameters for the VM were used. Once the VM's operating system was installed and found to be functional, VirtualBox's Guest Additions were installed therein.

The system appeared to be in good working order except that *dwarfdump* and its required dependencies were not installed from the media installation and the online repositories for Ubuntu 10.10 were no longer available. Thus, the source code for the variously required packages had to be downloaded from the web, compiled and then installed within the VM. Once this was done, the operating system was then temporarily shut down.

Upon a successful reboot, the VM's memory was dumped. This was done by restarting the VM using the following command:

```
$ virtualbox --debug --startvm "Ubuntu 10.10 x86"
```

VM memory was dumped using the following command:

```
$ vboxmanage debugvm "Ubuntu 10.10 x86" dumpguestcore --filename ubuntu10_10_jynx2.mem
```

The author-generated Volatility profile, *ubuntu_10_10_profile.zip*, was generated as per the instructions found in [6]. The profile is available to reader as per the eligibility requirements set out on *page xiii*.

1.6 Memory image metadata

Two memory images were taken of the VM. One was taken just prior to infection (see Section 1.5 for details) and the other just after rootkit infection. In so doing, it is possible to compare a clean system to an infected system in the event that such comparative information is required during the analysis of the infected memory image.

For these two memory images, similarities in their fuzzy hashes have been identified in the tables below so that the reader can readily identify large memory structures that have more or less remained the same.

Both acquired memory images should have been exactly 2 GiB in size, but as it turned out were not. Instead, they were each approximately 2.13 GiB in size, thereby indicating that the VirtualBox-specific overhead for this memory dump was non-negligible.

The following two tables as found in the subsequent subsections have had the similarities in their fuzzy hashes paired up, as seen in pink.

1.6.1 Uninfected baseline memory image metadata

The following metadata accurately describes the uninfected baseline memory image:

Table 1: Linux Ubuntu 10.10 x86 uninfected memory image metadata.

Memory image name	Ubuntu_10_10.mem
Actual size (exact)	2,285,979,432 bytes
Expected size (exact)	2,147,483,648 bytes
SHA1 hash	7f557359ee9c64b90a32f72cbd575b1de4e81107
Fuzzy hash	6291456:hYw8PWLUj+Q/TFUHvsfw+yVH67T8nU7Rp+0wgYjtVw08:PWnNj

1.6.2 Infected memory image metadata

The following metadata accurately describes the infected memory image:

Table 2: Linux Ubuntu 10.10 x86 Jynx2 infected memory image metadata.

Memory image name	ubuntu_10_10_jynx2.mem
Actual size (exact)	2,285,979,432 bytes
Expected size (exact)	2,147,483,648 bytes
SHA1 hash	d371451e4cd25ec982108c25181ee2ca1a73d4e2
Fuzzy hash	6291456:kdG30VYbf1sBWr0Fc+yFU0/NVw+b9V77CnUqR+HaLBu9rtVw0w:B3gzsI+q

1.7 Compiling and loading the rootkit

The rootkit's source code, found in downloaded file *jynx2.tgz* (SHA1 hash of da750d4db065480cc6243c34a55edd7e901ce63b), was copied over to the VM through the mounting of a *Shared Folder* (mounted read-only) to directory */tmp,* where it was unpackaged and compiled according to the following commands:

```
$ tar xzf jynx2.tgz
$ make all
$ make install
```

Upon successful compilation, the rootkit is then loaded into userland once a new process is run. If the underlying system does not support or have */etc/ld.so.preload* then the LD_PRELOAD environment variable will have to be set [1]; this was not necessary under Ubuntu 10.10.

Finally, although configuration changes can be made to *config.h*, which stores the user (or attacker – depending on the case) configuration settings, only the defaults were used for this infection.

The rootkit is now compiled but the system is not yet been infected. In order for the system to be infected, the rootkit/library must be loaded by some process, preferably some long living process that will draw less attention to itself.

However, the author has decided that rather than launch additional processes that would infect the system, he has instead opted not to in order to make the investigation more challenging.

1.8 AV scanners used

This report makes use of six anti-virus scanners, the same six as those used in report [7]. These scanners continue to represent a wide cross-section of various detection mechanisms necessary for the detection of diverse malware. Each scanner was updated July 23, 2014; the analysis was then carried out. Scanner specifics are listed in the following table:

Table 3: 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

2 Peripheral concerns

2.1 Why examine Linux memory images or make them available?

After extensively searching the available public literature, it became clear that few detailed Linux-based memory analyses could be found. In addition, those few reports or documents that were found were not of sufficient quality to enable others to readily learn the necessary techniques or approaches to conduct their own analyses that were specifically targeted towards non-memory specialists and non-reverse engineers.

The author asserts that by methodically conducting various Linux-based memory analyses using Volatility and sharing the techniques and methods used for these analyses with the digital forensics community, it will help to further advance the capabilities of investigators and incident handlers alike when dealing with potentially infected Linux memory images. Just as with the now completed Windows series of reports, which provide a detailed methodology for conducting Volatility-based malware memory analysis for non-experts, this series of Linux-based reports hopes to have the same impact for the same audience.

In researching what has been done thus far by the community with respect to Linux-based memory analyses, the author has found that there are few usable sources of malware-infected Linux memory images available for public consumption. However, the exception to this is the set of memory images made available by Raytheon Pikewerks (also known as SecondLook). While SecondLook makes various images available there are no freely associated memory profiles found with their images. Instead, in order to use these memory images, users must either build their own systems matching the kernel version and platform (x86 or x64) of the underlying memory image or use SecondLook Forensics, a commercial product that can automatically find the appropriate memory profile of an arbitrary Linux memory image. In order to reach the largest possible audience the author has opted to build not only his own infected VMs and acquire their memory but also to use Volatility because it is readily available to anyone who needs it without cost.

2.2 Volatility background

Volatility 2.4 is used throughout this work for the analysis of the memory image infected by the Jynx2 rootkit. The version of this framework, at the time of this writing, is considered the stable public release and is suitable for use by both the general public and investigators alike, although it may not necessarily have the most recent or bleeding-edge plugins. It was released for public use August 2014.

Originally written by Aaron Walters of Volatile Systems, Volatility has become a full-fledged memory analysis framework. It is written entirely in Python and can therefore be run atop Windows, Linux and various other operating systems supporting Python. Moreover, it has begun to support Linux-based memory analysis, although its Windows support is currently both more robust and reliable. Currently, it is developed by a variety of contributors, although the most well-known of these are Michael Ligh, Jamie Levy, Brendan Dolan-Gavitt, Andrew Case and Mike Auty. Furthermore, each of these individuals has made significant contributions to the digital forensic community over the last few years. Michael Cohen, who was formerly with the project, has gone on to found *Rekall* (https://code.google.com/p/rekall/), a memory analysis framework similar to Volatility that at the time of this writing is not yet ready for public use.

The only other currently supported and maintained Linux memory forensics framework worth mentioning is SecondLook, which is as powerful for Linux as Volatility is for Windows. However, it is expected that eventually the Linux capabilities of Volatility will rival those of SecondLook.

The Linux plugins currently supported by this version of Volatility are described in Annex A.

2.3 Purpose of these tutorials

Although online tutorials exist in various locations concerning certain infected Linux-based memory images, these tutorials are generally written for a highly technical audience already familiar with software reverse engineering and memory forensics. Moreover, they typically provide either too little information or are too technical to be of much use to most investigators and incident handlers. Instead, it could be argued that these tutorials are altogether insufficient in aiding investigators learn the necessary techniques and procedures required to apply digital forensics to memory analysis.

Thus, the author asserts that by re-examining and thoroughly documenting the steps and procedures used to identify various rootkit-based infections it will aid the reader in unravelling their own malware-based investigations. Furthermore, it is hoped that these reports will build a compendium of knowledge that will serve the forensic community as learning guides and tutorials.

The author has made all efforts to ensure that this document and the investigation of the Jynx2 rootkit is comprehensible to the general computer forensic practitioner, in the hopes of reaching as wide an audience as possible, in order to have a more significant impact.

2.4 About VirtualBox's VM memory dump

VM-based memory dumps are typically very reliable, both those of VMware and VirtualBox. However, under VirtualBox, an image of the VM's memory is saved to disk only when the VM's state is saved, not when it is paused. It is important to note that when the VM and its state are saved, the memory image is not in a "raw" like format that can be used directly by Volatility. Fortunately, VirtualBox provides a mechanism for generating raw-like memory dumps which is accomplished by placing the VM into debugging mode [8]. These raw-like memory dumpfiles are as reliable as those obtained using VMware, albeit they are in a different format.

Older versions of Volatility did not support VirtualBox memory dumps, which therefore required investigators and incident handlers to manually identify where the actual memory image resided in order to carve it out [8]. However, recent versions of Volatility fully support VirtualBox 32 and 64-bit memory dumps thereby simplifying things.

Although VMware memory snapshots and VirtualBox core dumps are similar, a VirtualBox core dump adds an ELF-based header and other overhead to the beginning of the dumpfile. As such, that overhead must be properly handled by the memory analysis program/tool.

However, unlike software-based memory acquisition where the investigator/incident handler must interact with the actual target system, generating a VM-based memory or core dump does not require any direct contact, interaction or introduction of new software with the current VM. Because direct VM memory acquisition is considered serialized, the issue of interaction/contamination is no longer an issue. That is to say, the memory image captured should be a complete representation of the VM's memory state at the moment of acquisition. This significantly diminishes any potential concerns that could be raised about the integrity of the memory image due to the lack of interaction between the investigator/incident handler with the VM for memory acquisition.

2.5 Issues concerning data carving, AV analysis and the NSRL

Unlike Windows-based memory images, it turns out that data carving is not particularly effective against Linux-based memory images. Experimentation by the author has revealed that once a Linux binary, whether an executable or a compiled library file has been loaded into memory, it loses its ELF header thereby making its detection very difficult. It is possible that this phenomenon is already known but the author could not find any literature indicating this fact. Thus, without a header from which to start the various data carvers and recovery software the author attempted for the recovery of executables and libraries from different test memory images did not succeed. In ten different memory experiments using both 32 and 64-bit Linux operating systems, only one ELF-based file was ever recovered. The other files recovered were mostly text-based data files.

However, the reader may recall that while these same data carving techniques worked relatively well against Windows-based memory images (as based on the Windows series of memory analysis reports), this is because Windows executables and libraries have distinctive headers and structures which are loaded into memory thereby making them readily identifiable and recoverable. Of course, ELF executables and libraries also have recognizable headers and structures; however, they are just not loaded into memory. As of Volatility 2.4, a new plugin, *linux_elf*, has been designed to help investigators determine where ELF files are residing within a memory image using alternate means.

Moreover, the various techniques examined in the Windows series of reports by the author found that occasionally some of the malware carved from a memory image matched those dumped from the memory image using Volatility. What this means is that data recovery tools and software are more likely to recover intact (or partially intact) malware from Windows memory images as compared to those from Linux. Furthermore, the various MD5/SHA1 and fuzzy hashing (file similarity matching) conducted throughout the various Windows reports has confirmed this assertion.

Further complicating Linux-based malware memory analysis is the lack of Linux-specific malware detection using various AV scanners. While the various scanners used throughout the

Windows reports (Avast, AVG, BitDefender, ClamAV[1], Comodo[1], Frisk F-Prot and McAfee) worked well against both Volatility-dumped and data-carved files, these very same AV scanners fared poorly against the Linux-based rootkits examined in this and subsequent follow-up reports. Moreover, most of the scanners could not detect anything malicious concerning these rootkits' source code. However, quite the opposite was in fact expected. Since these rootkits were all open source, it would have followed that the various scanner vendors would have included some basic signature or heuristic detection capability into their products, since these rootkits would inevitably be used as the basis for future rootkits. Unfortunately, this was not at all the case.

Thus, both this report and series of reports will make little use of AV scanners. That unfortunately requires the reader to have a better understanding of Linux in order to make up for what the scanners fail to detect. Nevertheless, certain portions of each Linux-based report will use AV scanners in the hope that they may be able to reveal something pertinent concerning a rootkit or its source code. Specifics are available in the analysis portion of this and subsequent follow-up reports. Moreover, these scanners are a representative cross-section of the various technologies employed by different scanners making their application herein aptly suitable.

Finally, the NSRL (National Software Reference Library) as a standardised and trustworthy source of computer operating system and application file names and hashes (MD5/SHA1) is not particularly suitable for Linux-based investigations. There are far too many Linux distributions (hundreds of publicly available distributions are known to exist) to be covered by the NSRL, including all the various kernel versions in use. As such, it does not make sense to rely on the NSRL for file name listings and hashes for comparative purposes against data recovered/carved from a Linux memory image. For that reason, these reports and their examination of the various infected Linux memory images will not derive assistance from the NSRL as was done for the Windows series of reports.

[1] This AV was used in some Windows memory malware reports but not others.

3 Memory investigation and analysis of Jynx2 using Volatility

3.1 Step 1: AV analysis of memory images and source code

This step examines the infected memory image, source code and compiled rootkit using various scanners in the hope of identifying if any of them could be identified as infected. The scanners used in this step are the same as those listed in Section 1.8.

3.1.1 Memory image analysis

The following results were obtained after scanning the infected memory image, *ubtunu_10_10_jynx2.mem*:

Table 4: Results for the AV scanning of the infected memory image.

Scanner	Image	Results
Avast	ubuntu_10_10_jynx2.mem	Nothing found
AVG	ubuntu_10_10_jynx2.mem	Nothing found
BitDefender	ubuntu_10_10_jynx2.mem	Nothing found
Comodo	ubuntu_10_10_jynx2.mem	Nothing found
F-Prot	ubuntu_10_10_jynx2.mem	Nothing found
McAfee	ubuntu_10_10_jynx2.mem	Nothing found

3.1.2 Source code analysis

The rootkit's source code was then analysed using the aforementioned scanners. Upon scanning all the various files included therein, nothing was identified as malicious or infected by any of the scanners.

3.1.3 Rootkit analysis

The two compiled library files were obtained by manually mounting the VM disk image in order to copy them to the host system's disk, and from there was scanned using the aforementioned scanners (see Section 1.8). Upon their having been thoroughly scanned, no infection could be identified by any of the scanners for either library file. At this time, both libraries were submitted to VirusTotal for inspection against a total of 53 scanners, all of which failed to detect either sample as malicious or infected. The two reports can be found in Annexes C.1 and C.2, respectively.

Library file *jynx2.so* is the actual rootkit while library file *reality.so* is used to reveal what the rootkit actually hides. Additional specifics can be found in the rootkit's source code and README file.

The two library files had the following metadata:

Table 5: Compiled rootkit metadata.

Library	Size (in bytes)	SHA1 Hash
jynx2.so	27,509	BD9254C99C2DE1BF39912C5A273FE66D90F86D5D
reality.so	11,886	FF8380BBF159F9F1E594DF7B3EE574C6F636F2B8

3.2 Step 2: Volatility system information extraction

This next step examines the infected memory image using Volatility plugins that provide system information about the underlying suspected computer and its operating system.

3.2.1 Plugin linux_banner

This plugin is used to determine information about the Linux kernel, its revision and architecture. The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_jynx2.mem linux_banner
```

The plugin then generated the following output:

```
Linux version 2.6.35-22-generic (buildd@rothera) (gcc version 4.4.5
(Ubuntu/Linaro 4.4.4-14ubuntu4) ) #33-Ubuntu SMP Sun Sep 19 20:34:50 UTC
2010 (Ubuntu 2.6.35-22.33-generic 2.6.35.4)
```

The output indicates that a Linux 2.6 generation kernel was running, specifically revision 2.6.35-22. It was a SMP-enabled kernel and was compiled using GCC version 4.4.5 (September 19, 2010). Finally, it was a 32-bit (x86) kernel but did not support PAE.

3.2.2 Plugin linux_cpuinfo

This plugin is used to identify the type and number of CPUs running atop the suspect computer system. The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_jynx2.mem linux_cpuinfo
```

The following output was then generated by the plugin:

Processor	Vendor	Model
------------	----------------	-----
0	GenuineIntel	Intel(R) i7-4500 CPU @ 1.80Hz

The make and model of the processor identified by this plugin are correct and indicate that the system was allocated a single processor.

3.2.3 Plugin linux_dmesg

This plugin is used to identify important boot-up information and kernel-based messages about the underlying computer system. The UNIX/Linux *dmesg* command, upon which this plugin is based, identifies kernel and various device driver boot-up information and output structures found residing in memory that are typically found in system file */var/log/dmesg*[2].

Thus, using this plugin it may be possible to identify what kernel (and its revision) was running, the number and type of CPUs, instantiated system services, the map of system memory, networking and many other essential capabilities (both software and hardware) that a typical Linux system will have. The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_jynx2.mem linux_dmesg
```

The output is too long to list here, but a full listing can be found in Annex B.1. After a detailed inspection of the output, nothing out of the ordinary was identified.

3.2.4 Plugin linux_iomem

This plugin provides the physical memory mapping of the suspect computer system, which in this case is a virtual machine. An in-depth examination of this virtual machine's physical memory mapping is outside the scope of this report; however, additional information concerning the interpretation of this data can be found in [4].

The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_jynx2.mem linux_iomem
```

This plugin then resulted in three of the following five columns as shown in the following table. The fourth and fifth (right-handed) columns were added by the author to facilitate the table's reading.

[2] Not all UNIX systems necessarily use this specific file. Mileage will vary according to the underlying operating system.

Table 6: VM physical memory mapping for suspected system.

Hardware	Starting Address	Ending Address	Size Difference	Size (in bytes)
reserved	0x0	0xFFF	0xFFF	4,096
System RAM	**0x1000**	**0x1FFF**	**0xFFF**	**4,096**
reserved	0x2000	0xFFFF	0xDFFF	57,344
System RAM	**0x10000**	**0x9FBFF**	**0x8FBFF**	**588,800**
reserved	0x9FC00	0x9FFFF	0x3FF	1,024
Video RAM area	0xA0000	0xBFFFF	0x1FFFF	131,072
Video ROM	0xC0000	0xC7FFF	0x7FFF	32,768
Adapter ROM	0xE2000	0xE2FFF	0xFFF	4,096
reserved	0xF0000	0xFFFFF	0xFFFF	65,536
System ROM	0xF0000	0xFFFFF	0xFFFF	65,536
System RAM	**0x100000**	**0x7FFEFFFF**	**0x7FEEFFFF**	**2,146,369,536**
Kernel code	0x100000	0x5D029D	0x4D029D	5,046,942
Kernel data	0x5D029E	0x818667	0x2483C9	2,393,034
Kernel bss	0x8CA000	0x9A0ADB	0xD6ADB	879,324
ACPI Tables	0x7FFF0000	0x7FFFFFFF	0xFFFF	65,536
0000:00:02.0	0xE0000000	0xE7FFFFFF	0x7FFFFFF	134,217,728
0000:00:03.0	0xF0000000	0xF001FFFF	0x1FFFF	131,072
e1000	0xF0000000	0xF001FFFF	0x1FFFF	131,072
0000:00:04.0	0xF0400000	0xF07FFFFF	0x3FFFFF	4,194,304
vboxguest	0xF0400000	0xF07FFFFF	0x3FFFFF	4,194,304
0000:00:04.0	0xF0800000	0xF0803FFF	0x3FFF	16,384
0000:00:06.0	0xF0804000	0xF0804FFF	0xFFF	4,096
ohci_hcd	0xF0804000	0xF0804FFF	0xFFF	4,096
0000:00:0b.0	0xF0805000	0xF0805FFF	0xFFF	4,096
ehci_hcd	0xF0805000	0xF0805FFF	0xFFF	4,096
0000:00:0d.0	0xF0806000	0xF0807FFF	0x1FFF	8,192
ahci	0xF0806000	0xF0807FFF	0x1FFF	8,192
Local APIC	0xFEE00000	0xFEE00FFF	0xFFF	4,096
reserved	0xFFFC0000	0xFFFFFFFF	0x3FFFF	262,144

The VM, allocated a total of 2 GiB (2,147,483,648 bytes) RAM is able to use 2,146,962,432 bytes, leaving 521,216 (509 KiB) bytes left hidden to the VM's operating system. This hidden memory was used and reserved by the VM's BIOS.

The reason an investigator/incident handler should use this plugin is so that he can be aware of the different address ranges in use by the hardware (virtualized or not) and operating system. Although this information is not of direct use, it can be used to validate that the memory image suspected of harbouring malware has not modified the operating system's virtual memory manager or other kernel components into thinking the system has less memory than it physically has. However, had the amount of unseen memory been much larger than the 509 KiB used by the BIOS, this could then have indicated that the malware had been busy making changes to the system in order to hide itself. While this capability has not yet been seen in Linux malware, this does not preclude it from existing.

3.2.5 Plugin linux_slabinfo

This specific plugin is used to provide kernel slab-based information. The kernel slab structure is a specific structure kept in */proc* used to keep track of different kernel structures that rely on various caches. These include, but are not limited to, filesystem buffers, network buffers and caches, inodes and many others.

This plugin only supports SLAB-based kernels and as such will only work with memory images using kernel 2.6.22 and earlier. Kernels 2.6.23 and later, by default, use SLUB-based memory management. [11] [12] [13].

3.2.6 Plugin linux_mount_cache

This plugin is the preferred method of obtaining a list of mounted disk, kernel and virtual filesystems. Filesystem identification is an important step in acquiring system information in order to assess where pertinent data or information may reside.

This plugin only supports SLAB-based kernels and as such will only work with memory images using kernel 2.6.22 and earlier. Kernels 2.6.23 and later, by default, use SLUB-based memory management. [11] [12] [13]

3.2.7 Plugin linux_mount

Although this plugin is not the preferred manner for obtaining a list of mounted disk, kernel and virtual filesystems, unlike the previous plugin, it did work, even if some of the output is not the same, as would have been obtained using the *linux_mount_cache* plugin. The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_jynx2.mem linux_mount
```

This plugin then resulted in the following output, which has been reformatted and whose order has been rearranged to improve its legibility:

- /dev/disk/by-uuid/b13dedba-11eb-497f-96b2-e06d37b3aef1 / ext4 rw,relatime
- binfmt_misc /proc/sys/fs/binfmt_misc binfmt_misc rw,relatime,nosuid,nodev,noexec

- fusectl /sys/fs/fuse/connections fusectl rw,relatime
- gvfs-fuse-daemon /root/.gvfs fuse rw,relatime,nosuid,nodev
- none /dev devtmpfs rw,relatime
- none /dev/pts devpts rw,relatime,nosuid,noexec
- none /dev/shm tmpfs rw,relatime,nosuid,nodev
- none /proc proc rw,relatime,nosuid,nodev,noexec
- none /sys sysfs rw,relatime,nosuid,nodev,noexec
- none /sys/kernel/debug debugfs rw,relatime
- none /sys/kernel/security securityfs rw,relatime
- none /var/lock tmpfs rw,relatime,nosuid,nodev,noexec
- none /var/run tmpfs rw,relatime,nosuid
- ------ /media/dvd vboxsf rw,relatime,nodev

Upon closer examination of the output, nothing appeared out of the ordinary. The above list is what would be expected from a typical modern Linux desktop.

3.2.8 Summary

In this step, various plugins were run in order to establish information about the VM's underlying operating system. Despite the many pages of output generated by the various plugins, no clues or hints as to this memory image's infection could thus far be identified.

However, these plugins do provide important basic information about the underlying hardware and operating system which may be of use when correlating the output from the other various plugins in the ensuing steps.

The author is of the opinion that the two most important plugins are in this step are *linux_banner*, and *linux_dmesg*. However, plugins *linux_iomem* and *linux_mount* may provide additional indications of malware presence, but only if the malware is capable of modifying the kernel's perception of available "System RAM" or mount points, respectively.

It is important that analysts use the appropriate Volatility plugins supporting the memory image's underlying kernel version.

3.3 Step 3: Volatility process listings and analysis

In this step, specific Volatility plugins will be used to identify process-based information concerning the infected memory image.

3.3.1 Plugin linux_psaux

This Volatility plugin is used to provide a full process listing of the system. Its output is approximately the same as would be obtained running the *ps -aux* command via a terminal. The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_jynx2.mem linux_psaux
```

The resulting output consisted of 150 listed processes. However, as this output is too long to list in this section it can instead be found in Annex B.2. Everything in this long list of processes appears altogether normal.

3.3.2 Plugin linux_pslist

This interesting Volatility plugin is also used to list all running processes on a system. It works by walking the *task_stuct->tasks* linked list [10], similar to Volatility's Windows process listing plugins, except only for Linux. The plugin can list all active processes (except for the system swapping process(es)) [10]. According to Volatility's documentation [10], if the output under the DTB column is blank then it is very likely a kernel thread. This includes drivers and other kernel modules visible from userland.

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_jynx2.mem linux_pslist
```

The output generated by this plugin is too long to include here but can be found in Annex B.3. Importantly, the same numbers of processes were found using this plugin as with the previous plugin, 150 processes in all. Again, nothing out of the ordinary was identified.

3.3.3 Plugin linux_pslist_cache

This plugin attempts to build a list of active processes from *kmem_cache*, the kernel's memory cache [10]. In effect, it should reproduce the same results as the *linux_pslist* plugin using a different mechanism, which is useful in corroborating the results of the other available process listing plugins.

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_jynx2.mem linux_pslist_cache
```

However, in running this command the following output indicating that the plugin failed to function correctly:

```
INFO : volatility.plugins.linux.slab_info: SLUB is currently unsupported.
```

The reason this plugin does not work is because it supports SLAB-only based kernels, not SLUB-based kernels [11] [12] [13]. SLUB kernels became the default memory management mechanism for Linux as of kernel 2.6.23 [13].

3.3.4 Plugin linux_pstree

The purpose of this plugin is to identify the relationship between processes, in effect to identify a given process' parent (or PPID). The reader may have noticed that to date none of the Linux process listing plugins provides the PPID of the variously identified processes. Thus, to identify these relationships, the following command was issued:

```
$	volatility --profile=Linuxubuntu_10_10_profilex86 -f
	ubuntu_10_10_jynx2.mem linux_pstree
```

Again, this plugin has identified 150 processes, the same number as identified by *linux_psaux* and *linux_pslist*. The output of this plugin, too long to be listed herein, but can instead be found in Annex B.4.

3.3.5 Plugin linux_pidhashtable

This interesting plugin can be used to identify hidden or previously unseen processes. However, it is not the same as the Windows *psxview* plugin. Instead, it works by walking the PID hash table [10]. The *plugin* validates that a given process forms part of the PID hash table maintained by the operating system. This lookup (or hash) table is similar to that used by Windows in that they are both doubly linked lists. In the same manner that rogue Windows processes can unlink themselves from the Windows process table, rogue Linux processes can unlink themselves from the PID hash table and this plugin can aid in identifying them. Moreover, its output is not that different from that of the *linux_pslist* plugin.

The plugin was run using the following command:

```
$	volatility --profile=Linuxubuntu_10_10_profilex86 -f
	ubuntu_10_10_jynx2.mem linux_pidhashtable
```

The plugin's output is too long to list here; as such, it can instead be found in Annex B.5. Looking at that long list, 265 processes in all, nothing appears to be particularly out of place. Many more items are listed than with the previous process listing plugins, but ultimately nothing looked out of the ordinary could be found.

3.3.6 Plugin linux_psxview

This plugin is related to the *psxview* plugin used for Windows memory investigations. However, this Linux-specific plugin makes use of very different data structures found in Linux-based memory images.

Memory offsets are specified in terms of virtual addresses and the plugin uses five distinct memory analysing algorithms. The first of these is *pslist*, which uses the same technique used by the *pslist* plugin (see Section 3.3.2 for details). The second is *pid-hash* and it helps to identify hidden processes (see Section 3.3.5 for details). The third is *kmem_cache* which examines the kernel's memory cache (see Section 3.3.3). Specifically, this cache stores information about not only ongoing processes but also metadata concerning terminated processes, sometimes even those which may have completed long ago, depending on the degree of process creation within the operating system. [10]

However, the plugin has changed since version 2.3.1 of Volatility. The current version, 2.4, provides two new fields for the *linux_psxview* plugin. Specifically, the *Parents* and *Leaders* fields have been added. Accorded to [9], the field *Parents* "is populated by following the parent pointers of processes and threads found in the PID hash table" while the *Leaders* field "is populated by gathering the thread group leader pointer of each process and thread."

When working with this plugin, it is important to identify those processes or threads that are obvious outliers. It is normal that the various field values will vary a lot, but those that are too different from those surrounding it warrant additional inspection.

The plugin was run using the following command:

```
$    volatility --profile=Linuxubuntu_10_10_profilex86 -f
     ubuntu_10_10_jynx2.mem linux_psxview
```

The output from the plugin is far too long to list here; as such, it can be found listed in Annex B.6. Looking at that long list, 265 processes in all, the very same processes identified by plugin *linux_pidhashtable* (see Section 3.3.5 for details) with the exception of the *swapper* *** process (PID 0), was not found using the former plugin. However, as it turns out, this process scan is entirely legitimate for Linux and UNIX-like systems where PID 0 is reserved for kernel process *swapper* or *sched* (system scheduler) [14].

3.3.7 Summary

The use of process listing and process scanning plugins is an important step in any memory investigation that should not be skipped as malware may leave behind indications of its presence. Each of the plugins presented in this step have the ability to provide important clues or contextual information concerning the relationship between various processes and threads.

It is important that the reader remember that this rootkit, which is userland-based, has not yet been loaded into memory. Instead, the system has been infected but the rootkit is not yet active. Thus, only in the second part of this study will an analysis of a "live" infection be analysed.

3.4 Step 4: Volatility history listing

In this step, various command shell listing plugins will be used to attempt to identify pertinent shell histories.

3.4.1 Plugin linux_bash

This particular plugin searches a memory image for command shell histories, similar to Volatility's Windows-based command history plugins. Moreover, this is a brute force plugin in that it scans the entire memory image for signs of shell histories and as such may at times output erroneous information [10].

The plugin was run using the following command:

```
$    volatility --profile=Linuxubuntu_10_10_profilex86 -f
     ubuntu_10_10_jynx2.mem linux_bash -A
```

Although the plugin can be used with the *-A* option, which when used can help to ensure that all processes are scanned for shell history information, it can take many hours to process a large memory image. It is also possible that the plugin will crash when used with this option. However, the option is useful because attackers may have copied the shell program (i.e., *bash*) to another name (i.e., */tmp/hsab*) and ran it to circumvent the detection of shell histories in memory.

Tests by the author have found that running the plugin with *-A* parameter took several hours to complete. Moreover, it was determined that the same amount of information was identified when the plugin was used without the option. Of course, mileage will vary.

In a typical investigation, there could be many hundreds or even thousands of lines of shell histories to go over. Typically, after a system reboot, pre-existing shell histories will no longer be recoverable; this, of course, is only a rule of thumb and there are times when pre-reboot data will remain intact in memory for recovery.

The output generated by the plugin, having been pruned by the author to remove non-pertinent shell history entries, is listed below and details the compilation of the Jynx2 rootkit by the author.

Table 7: Pertinent plugin output for linux_bash (pruned and sorted chronologically).

Pid	Name	Command Time	Command
2224	bash	2014-05-24 01:02:06 UTC+0000	cd /tmp/
2224	bash	2014-05-24 01:02:09 UTC+0000	unzip Jynx2-master.zip
2224	bash	2014-05-24 01:02:11 UTC+0000	cd Jynx2-master
2224	bash	2014-05-24 01:02:12 UTC+0000	ls
2224	bash	2014-05-24 01:02:13 UTC+0000	make
2224	bash	2014-05-24 01:02:19 UTC+0000	make install
2224	bash	2014-05-24 01:02:25 UTC+0000	ls -al /

However, investigators and incident handlers should not expect that in every case they encounter that an attacker's shell commands will be retrievable, as this will depend on many factors.

3.4.2 Plugin linux_bash_env

This new plugin has the ability to find various environment variables used by a given command line shell. As such, this plugin has the potential to provide important clues concerning an attacker's actions against a suspected system since environment variables added or changed by the attacker via a command line shell (and possibly shell scripts) may be uncovered using this plugin.

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_jynx2.mem linux_bash_env
```

Environment variables were identified for PIDs 1819 and 2224, the two previously identified *bash* shells. However, careful analysis of this plugin's output has revealed that nothing of use or importance could be found within its relatively short output:

Table 8: Plugin output for linux_bash_env.

PID	Name	Variables
1819	bash	SHELL=/bin/bash TERM=linux XDG_SESSION_COOKIE=1d4ca871baf530534fac470f00000002-1400893280.821095-1009310097 HUSHLOGIN=FALSE USER=root LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lz=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.rar=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.axv=01;35:*.anx=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.axa=00;36:*.oga=00;36:*.spx=00;36:*.xspf=00;36: MAIL=/var/mail/root PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games PWD=/root LANG=en_CA.UTF-8 SHLVL=1 HOME=/root LOGNAME=root LESSOPEN=| /usr/bin/lesspipe %s LESSCLOSE=/usr/bin/lesspipe %s %s _=/usr/bin/startx

PID	Name	Variables
2224	bash	ORBIT_SOCKETDIR=/tmp/orbit-root SSH_AGENT_PID=1927 SHELL=/bin/bash TERM=xterm XDG_SESSION_COOKIE=1d4ca871baf530534fac470f00000002-1400893284.243018-255177899 WINDOWID=60817412 HUSHLOGIN=FALSE GNOME_KEYRING_CONTROL=/tmp/keyring-bbrbSI GTK_MODULES=canberra-gtk-module USER=root LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lz=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.rar=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.axv=01;35:*.anx=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.axa=00;36:*.oga=00;36:*.spx=00;36:*.xspf=00;36: SSH_AUTH_SOCK=/tmp/keyring-bbrbSI/ssh SESSION_MANAGER=local/ubuntu1010:@/tmp/.ICE-unix/1939,unix/ubuntu1010:/tmp/.ICE-unix/1939 MAIL=/var/mail/root PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games PWD=/tmp/Jynx2-master LANG=en_CA.UTF-8 SHLVL=2 HOME=/root GNOME_DESKTOP_SESSION_ID=this-is-deprecated LOGNAME=root DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-KgcVSENV3U,guid=d68ee437de1ffefe95c631410000001d XDG_DATA_DIRS=/usr/share/gnome:/usr/local/share/:/usr/share/ LESSOPEN=| /usr/bin/lesspipe %s WINDOWPATH=7 DISPLAY=:0.0 LESSCLOSE=/usr/bin/lesspipe %s %s XAUTHORITY=/root/.Xauthority COLORTERM=gnome-terminal _=/bin/ls OLDPWD=/tmp

3.4.3 Plugin linux_bash_hash

A unique and new plugin, its objective is to recover the *bash* hash table kept in memory by the *bash* command line shell. *Bash* uses a hash table to keep track of commands and the number of times they were run. This plugin also provides the *-A* command line parameters which scan the entire memory image for additional hash tables. However, in so doing, if the memory image is too large the plugin could crash.

The plugin was run using the following command:

```
$	volatility --profile=Linuxubuntu_10_10_profilex86 -f
	ubuntu_10_10_jynx2.mem linux_bash_hash
```

The plugin was originally run with the *-A* but it crashed instead of producing useable results. Thus, the command was rerun without the additional parameter which produced a short table with no meaningful information whatsoever with respect to the investigation.

3.4.4 Summary

Looking for shell command histories can produce rewards, especially if a system's memory is acquired within at least several hours of compromise (or possibly more if the system is quiescent). However, it is not known if these plugins will work with non-*bash* shells.

What is recovered and its pertinence can vary greatly between investigations. As such, investigators and incident handlers must remember that these *bash*-based plugins represent only one small piece of the analysis. Fortunately, *bash* is the default shell for many Linux distributions.

In using these three plugins, only the *linux_bash* plugin found information that was directly pertinent as it found the various commands used for building the rootkit. However, the memory was acquired shortly thereafter. Instead, had acquisition occurred many hours or even one or more days after the fact these *bash* remnants may very well have been overwritten in memory by other data or information.

3.5 Step 5: Volatility file detection and dumping

In this step, various plugins will be used to attempt to isolate and dump important or suspicious files for further analysis.

3.5.1 Plugin linux_lsof

As the name of this plugin entails, it lists all open files, sockets, pipes, directories and other objects that the system currently has opened for a given process or list of processes (or all processes). This plugin functions similarly to the UNIX/Linux command *lsof*; however, it does not in any way list the same number of details the real *lsof* command does. For example, a typical Linux system running with X Windows will have at least several thousand open filesystem objects. However, in using this plugin, it is likely that less than half of the actual number of open objects will be listed.

The plugin was run using the following command:

```
$	volatility --profile=Linuxubuntu_10_10_profilex86 -f
	ubuntu_10_10_jynx2.mem linux_lsof
```

Correlating the output from this plugin against those from the *linux_psaux* plugin resulted in no actionable information or additional clues about the underlying infection. Moreover, this plugin does not have the ability to provide information concerning hidden processes. The plugin identified 1,159 objects (this number will be compared against the subsequent plugins). After taking much time to go over this output, nothing suspicious could be found therein.

3.5.2 Plugin linux_kernel_opened_files

This new plugin is used to list files and other filesystem objects that are opened or used from within the kernel itself.

The plugin was run using the following command:

```
$    volatility --profile=Linuxubuntu_10_10_profilex86 -f
     ubuntu_10_10_jynx2.mem linux_kernel_opened_files
```

The plugin did not work as it emitted an error concerning its inability to find object *hlist_bl_head* in the generated memory profile. This error was likely caused by a missing Python library or Volatility dependency. The reader is free to experiment with Volatility running atop other operating systems to determine if he can get it to work.

3.5.3 Plugin linux_proc_maps

This very powerful plugin can be used to learn important information about the underlying system as a whole or about one or more specific processes. Specifically, this plugin is used to identify process metadata including the name and location of the running process, shared libraries, stacks, inodes and memory address ranges.

The plugin was run using the following commands:

```
$    volatility --profile=Linuxubuntu_10_10_profilex86 -f
     ubuntu_10_10_jynx2.mem linux_proc_maps | tail -n +3 > proc_maps.txt

$    cat proc_maps.txt | awk '{print $9}' | sort | uniq > proc_maps_2.txt
```

The first command used was *tail* in order to remove the first two lines from the plugin's output (which is appended by Volatility) and then redirect the output to file *proc_maps.txt*. The second command reduces the plugin's 11,961 lines of output to a manageable 549. Although the new output is far shorter than the original output, it is still too lengthy to include herein; however, it can be easily regenerated by the reader at his convenience.

After analysing the shorter output, nothing out of the ordinary is found. Nevertheless, if an infection would have been active, this plugin might have identified something suspicious, in so long as the malware is not a hidden process, as this plugin does not appear to have abilities against hidden processes.

3.5.4 Plugin linux_find_file

3.5.4.1 Running the plugin

This particular plugin can be used to not only dump from the memory image pre-identified files (using other plugins) but it can also list all filesystem objects with an open handle in memory, which will often list far more objects than *linux_proc_maps* or *linux_lsof*. However, these two aforementioned plugins (*linux_proc_maps* and *linux_lsof*) do different things. Thus, when seeking out abnormal libraries and process names, plugin *linux_proc_maps* should be used before plugin *linux_find_file* but after *linux_lsof*.

The output of the *linux_find_file* plugin lists not only the inode number and inode memory reference but also provides the full name of the filesystem object with the open handle. Thus, this plugin provides much useful information that can be used to readily dump one or more objects from the memory image, but only if they are cached in memory.

However, this plugin is not designed for at large data recovery of cached filesystem objects held within the memory image. For that, the *linux_recover_filesystem* plugin should be used. Nevertheless, not every file with a handle in memory can necessarily be recovered from the memory image, as that file may not currently be residing within the filesystem cache.

The plugin was run using the following command:

```
$    volatility --profile=Linuxubuntu_10_10_profilex86 -f
     ubuntu_10_10_jynx2.mem linux_find_file -L > find_files.txt
```

This command resulted in text file *find_files.txt* that contained a listing of 12,624 unique filesystem objects. Unless actionable intelligence was made available from one of the previous plugins, this list of objects will have to be examined manually while looking for anomalies. Such a task may take several or more hours to thoroughly inspect.

While examining the plugin's output, various files including libraries and distinct directory names may standout. Those that were found for this investigation are as follows:

Table 9: Plugin output for linux_find_file (suspicious objects only).

Inode Number	Inode	File Path	Dumpable
393219	0xf4c64ce0	/XxJynx	Yes (but not useful)
393268	0xf4c651d0	/XxJynx/reality.so	Yes
393267	0xf4c65448	/XxJynx/jynx2.so	Yes
262420	0xf4c4cce0	/tmp/Jynx2-master	Yes (but not useful)
----------------	0x0	/tmp/Jynx2-master/install	No
----------------	0x0	/tmp/Jynx2-master/reality.c.gch	No
262548	0xf4c65bb0	/tmp/Jynx2-master/reality.so	Yes
----------------	0x0	/tmp/Jynx2-master/jynx2.c.gch	No
262546	0xf4c63bc8	/tmp/Jynx2-master/jynx2.so	Yes
----------------	0x0	/tmp/Jynx2-master/all	No

Inode Number	Inode	File Path	Dumpable
----------------	0x0	/tmp/Jynx2-master/SCCS	No
----------------	0x0	/tmp/Jynx2-master/RCS	No
262475	0xf4c4dbb0	/tmp/Jynx2-master/reality.c	Yes
262473	0xf4c4d938	/tmp/Jynx2-master/packer.sh	Yes
262435	0xf4c4d6c0	/tmp/Jynx2-master/jynx2.c	Yes
262432	0xf4c4d448	/tmp/Jynx2-master/config.h	Yes
262426	0xf4c4d1d0	/tmp/Jynx2-master/README	Yes
262425	0xf4c4cf58	/tmp/Jynx2-master/Makefile	Yes
262363	0xf4c4c7f0	/tmp/Jynx2-master.zip	Yes

*N.B.: The fourth column was added by the author and is not part of the plugin's output. The files highlighted in pink above have been confirmed to be extractable from recovered file **Jynx2-master.zip**. The ***.so** files are only available after compilation using the rootkit Makefile while files **all**, **SCCS**, **RCS**, **install**, **reality.c.gch** and **jynx2.gch** appear to have become available only after installation of the rootkit.*

These files have the distinct look of a rootkit installation package (with source code). However, file recovery is carried out only in the ensuing sub-step.

3.5.4.2 Dumping files of interest from the memory image

The next step is to actually dump as many of these memory-resident files from the image as possible using the following commands:

```
$     volatility --profile=Linuxubuntu_10_10_profilex86 -f
      ubuntu_10_10_jynx2.mem linux_find_file -i 0xf4c651d0 -O reality.so

…

$     volatility --profile=Linuxubuntu_10_10_profilex86 -f
      ubuntu_10_10_jynx2.mem l linux_find_file -i 0xf4c4c7f0 -O Jynx2-
      master.zip
```

In all, eleven files were dumped and the two instances of the two suspicious library files *reality.so* and *jynx2.so* had the same hashes (SHA1 and fuzzy hash) as the original rootkit when it was analysed, as found in Section 3.1.3.

Unzipping the file *Jynx2-master.zip* revealed the rootkit's source code and the README text file was particularly revealing. However, in a real-life situation, the source code would likely be long gone from system memory. This is because the source code, like anything else in memory, if it is not active for an extended period time (that is to say that it is not being used, compiled, edited, etc.) will eventually be cleared from the operating system's filesystem cache to make room for new objects. The same principle applies to rootkits and malware in memory. Because this rootkit is not currently active, in time it too would be expunged. However, if the rootkit or other malware were active in some form then it should be possible to detect it and extract it from memory using this plugin.

Thus, while it is important to conduct this step, if it is possible, the investigator/incident handler should not get use to this being the case where the source code and malware are all in memory at the same time. Moreover, rootkits detected and dumped using this plugin will not always be classifiable as a rootkit without further advanced analyses including AV scanning and possibly reverse engineering.

These files, both those recovered and unzipped could be readily scanned using the various AV scanners set out in Section 1.8. However, the scanners will not detect anything suspicious, infected or nefarious. As such, it is not always possible to know the true nature of the variously recovered filesystem objects without the efforts of reverse engineering.

3.5.5 Summary

This step has demonstrated how even rootkits can sometimes be detected using means other than process listing and scanning plugins. However, as already stated, it is important to bear in mind that the rootkit is not yet active[3]. Analysis of this rootkit in a live state will be conducted in the second part of this work. Nevertheless, the work demonstrated in this step has important implications for those investigating compromised systems within "a reasonable timeframe after the event" as this will likely leave behind important telltale signs of infection or compromise.

It is important to note that the *linux_lsof* plugin did not identify anything out of the ordinary; nevertheless, it is a useful plugin to use but is not a substitute for the actual *lsof* command.

The *linux_proc_maps* plugin is a powerful plugin that can display both important process-related metadata and identify process-specific libraries and other dependencies. However, in this specific investigation, this plugin did not succeed in revealing any information about the rootkit.

Other plugins, such as Volatility 2.4-specific *linux_library_list* could have been used to list only those libraries associated with the variously identified (and non-hidden only) processes and threads. However, there was no need as plugin *linux_proc_maps* was more than sufficient. At the same time, newer plugin *linux_enumerate_files* could have also been used instead of *linux_find_file*, but then the required inode information necessary for dumping suspicious files from the memory image to a storage location on disk was not available using the former.

Thus, by using the *linux_find_file* plugin, it was possible to successfully identify and dump the two suspicious libraries and rootkit installation kit from the memory image. Upon further inspection, it was determined that this kit was in fact the source code for the rootkit. Thus, exposing investigators and incident handlers to the capabilities of this plugin was of significant value and is of use in investigations in so long as not too much time has elapsed from the time of infection to the moment memory acquisition takes place as the kit and libraries will have eventually been flushed from the operating system's filesystem cache.

Plugin *linux_elfs* could have been run in this step too; however, it was very unlikely to have brought anything new to the investigation. This plugin is useful for directly associating processes,

[3] If a process such as a memory acquisition tool had been run rather than acquiring memory directly through VirtualBox's debug feature, then the rootkit would have become activated. At the time of acquisition, no additional commands, tools or processes were yet running which would have activated the rootkit. Some details will become clearer in the upcoming second report concerning the rootkit.

and hence running ELFs in memory, directly to disk-mappable files, especially the variously required libraries of these processes. However, the output quickly gets confusing and it is easy to become overwhelmed. For this reason, this plugin was not used and in its stead *linux_find_file* and *linux_proc_maps* were used.

Finally, the key to using the plugins presented in this step is seeking out aberrations or odd-looking filenames, library and or dependencies. This, of course, requires that an investigator/incident handler have some familiarity with UNIX-based systems in order to readily differentiate between normal and abnormal-looking indications.

3.6 Step 6: Volatility kernel-specific analyses

In this step, various plugins will be used to attempt to determine specific information about kernel modules and if necessary dump suspicious modules for further analysis.

3.6.1 Plugin linux_lsmod

This plugin, as its name implies, performs a listing of all visible Linux kernel modules running on the system, similar to the Linux *lsmod* command. However, unlike *lsmod*, this plugin provides the base address for every detected kernel module.

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_jynx2.mem linux_lsmod
```

This resulted in a significant number of modules that can be found in Annex B.7. The *-P* parameter could have been used to provide a more verbose output that would have instructed the plugin to list all specified module input parameters. In addition, the *-S* parameter could have been used to supplement the previous parameter to list all memory areas used a given kernel module. However, in order to reduce the reader's need for knowledge of reverse engineering, neither parameter was used.

Looking at the output in Annex B.7, nothing appeared out of the ordinary. In fact, all the kernel modules listed therein appear legitimate.

3.6.2 Plugin linux_hidden_modules

This plugin, as its name implies, is used to find hidden kernel modules that have been unlinked from the list of modules. This is a highly useful plugin as it can reveal information about hard to detect rootkits. While the *linux_check_modules* plugin also has the ability to detect hidden kernel modules, they work through vastly different mechanisms.

The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_jynx2.mem linux_hidden_modules
```

This plugin took many minutes to complete but succeeded in finding one "hidden" process that is as follows:

Table 10: Plugin output for linux_hidden_modules.

Offset (V)	Name
0xf80b12f8	????

Interestingly, the suspicious module's name could not be identified, but its base address was. Before attempting the dump this possibly suspicious module from the memory image using the *linux_moddump* plugin, it would be prudent to validate these results using the *linux_check_modules* plugin.

3.6.3 Plugin linux_check_modules

Possibly the most powerful of Volatility's Linux-based kernel module checking plugins, it performs kernel module differencing. Specifically, it looks for inconsistencies between the kernel module lists. On Linux systems, this information is obtained by comparing the results of kernel structure */proc/modules* against */sys/modules*. Thus, through this plugin it may be possible to corroborate the results of the *linux_hidden_modules* plugin.

The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_jynx2.mem linux_check_modules
```

This plugin took many minutes to complete but succeeded in finding the very same "hidden" process found using the previous plugin that is as follows:

Table 11: Plugin output for linux_check_modules.

Offset (V)	Name
0xf80b12f8	????

While the same base address was also marked as suspicious by this plugin, it is now time to attempt to dump it from memory using the *linux_moddump* plugin.

3.6.4 Plugin linux_moddump

The *linux_moddump* program is very similar to its Windows counterpart, *moddump*. It detects and dumps all visible kernel modules (in the case of Windows these are device drivers) and both plugins (Windows and Linux) have the ability to dump hidden modules (drivers) if a base address is specified. Thus, if there are indications of kernel-level malware activity and a module is not hidden, or at least a base address is known, the investigator/incident handler can attempt to dump it.

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_jynx2.mem linux_moddump -b 0xf80b12f8 -D .
```

The plugin attempted to create dump file *????.0xf80b12f8.lkm* but did not succeed in actually dumping any contents to this file. Instead, the following error message was emitted by Volatility:

```
ERROR : volatility.plugins.linux.lsmod: No section .symtab found. Unable to
properly re-create ELF file.
```

Trying with a slightly different address gave a completely different error, thereby indicating that the base address obtained by the two previous plugins was correct but that what was in memory was likely a corrupt remnant of some previous module that could not be successfully dumped.

3.6.5 Summary

This step has shown the various plugins that can be used to query a memory image for important information about listed and hidden kernel modules, and where appropriate dump a module, hidden and visible alike, from the memory image.

Although the *linux_tmpfs* plugin could have been used to examine the kernel in further detail, it was not the right plugin to use because typical kernel modules do not use or leave behind temporary files in a typical Linux system. These temporary locations usually include, but are not necessarily limited to */var/run*, */dev/shm* and */var/lock*, all locations where various process information and data are stored until reboot.

While plugins *linux_hidden_modules* and *linux_check_modules* identified some unknown module as hidden, this does not necessarily indicate that it is in fact hidden or suspicious. Instead, based on the information and evidence obtained thus far it may have been some unknown module that was loaded and then unloaded leaving behind a remnant of itself in memory which was detected by these two plugins. Considering the fact that the module could not be successfully dumped should not fuel the investigator/incident handler's belief that it is necessarily suspicious or nefarious.

Moreover, even though the *linux_moddump* plugin could have been used to dump all visible kernel modules from the memory image, the only way to reliably discern if any of them were infected would be to perform reverse engineering on all of them. Furthermore, as has been clearly demonstrated, none of the 51 scanners used by VirusTotal was able to identify the compiled rootkit (see Section 3.1.3 and Annex C for details) as infected or suspicious. Thus, using the aforementioned six scanners (see Section 1.8 for details) would make little sense. Instead, the investigator/incident handler would have to rely on the information provided by the *linux_lsmod* plugin in order to determine if additional analysis were warranted against one or more kernel modules.

3.7 Step 7: Volatility network-specific plugins

This step will examine the use of various network-based plugins as they pertain to this investigation.

3.7.1 Plugin linux_route_cache

This plugin produces information concerning the system's routing cache, which includes both ongoing and recently terminated communications. Of course, this plugin also lists additional important information including the underlying system's IP address and the various gateway addresses in use for the aforementioned communications, which could potentially be modified by certain malware to avoid detection.

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_jynx2.mem linux_route_cache | sort | uniq
```

This plugin then produced the following line-reduced output:

Table 12: Plugin output for linux_route_cache.

Interface	Destination	Gateway
eth0	192.168.0.1	10.0.2.2
eth0	224.0.0.22	224.0.0.22
eth0	224.0.0.251	224.0.0.251
eth0	91.189.89.199	10.0.2.2
eth0	91.189.94.4	10.0.2.2
lo	10.0.2.15	10.0.2.15
lo	127.0.0.1	127.0.0.1
lo	127.0.1.1	127.0.1.1
lo	224.0.0.1	224.0.0.1
virbr0	224.0.0.251	224.0.0.251

From this information, several items can be immediately picked out. Firstly, the system's IP address is 10.0.2.15 (remember that this system is in a VM). Gateway 224.0.0.251 is VirtualBox-related as per interface *virbr0* so there is nothing to be done for this address. Furthermore, gateways 224.0.0.1 and 224.0.0.22 are multicast addresses, specifically for "all hosts" and IGMP, respectively [15]. The only addresses that cannot be accounted for are 91.189.89.199 and 91.189.94.4. However, further investigation has revealed that these two addresses belong to Canonical Ltd., and since this is an Ubuntu distribution, there is no reason to suspect these two addresses are malicious. Thus, this plugin, while informative has not shed any additional light on this investigation.

3.7.2 Plugin linux_netstat

This plugin performs the equivalent of the UNIX/Linux *netstat* command in that it is used to print information concerning network connections (the actual *netstat* command does far more).

The plugin was run using the following command:

```
$    volatility --profile=Linuxubuntu_10_10_profilex86 -f
     ubuntu_10_10_jynx2.mem linux_netstat -v | grep -P '(TCP|UDP)'
```

This command resulted in the printing of all local and established network connections using either TCP or UDP-based connections. The output of this command was as follows:

```
UDP   0.0.0.0       : 5353 0.0.0.0     :   0              avahi-daemon/648
UDP   ::            : 5353 ::          :   0              avahi-daemon/648
UDP   0.0.0.0       :48756 0.0.0.0     :   0              avahi-daemon/648
UDP   ::            :51768 ::          :   0              avahi-daemon/648
UDP   0.0.0.0       :   68 0.0.0.0     :   0              dhclient/658
TCP   0.0.0.0       :    0 0.0.0.0     :   0 CLOSE        libvirtd/746
UDP   0.0.0.0       :   67 0.0.0.0     :   0              dnsmasq/945
TCP   192.168.122.1 :   53 0.0.0.0     :   0 LISTEN       dnsmasq/945
UDP   192.168.122.1 :   53 0.0.0.0     :   0              dnsmasq/945
TCP   ::1           :  631 ::          :   0 LISTEN       cupsd/1106
TCP   127.0.0.1     :  631 0.0.0.0     :   0 LISTEN       cupsd/1106
```

This information has revealed that there was one ongoing connection to a DNS server, 192.168.122.1. Further research indicated that this DNS server was found on the router/firewall of the inside-facing portion of the network the host system (non-VM) was connected to. There is nothing here to indicate any suspicious activity.

3.7.3 Summary

There is little reason to believe, in the author's opinion, that running the *linux_arp* plugin would have revealed any additional information of interest.

That said, the two plugins used in this section have revealed that there is no suspicious network activity going on.

3.8 Step 8: Additional checks

This step will run two additional sets of checks to search for injected code and credential escalation attacks common of certain types of rootkits.

3.8.1 Plugin linux_malfind

This new plugin is similar to the Windows version of this plugin. Its searches memory images for indications of code injection.

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_jynx2.mem linux_malfind
```

The plugin found no indication whatsoever of injected code within this memory image.

3.8.2 Plugin linux_check_creds

This plugin is used to check for elevated processes where the user/group credentials used to run a process have been maliciously raised, which is typical of certain rootkits.

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_jynx2.mem linux_check_creds
```

The plugin found no indication of process elevation.

3.8.3 Plugin linux_apihooks

This plugin is used to check for API hooking, which is sometimes known as inline hooking. This hooking technique is used by various malware to infect a system.

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_jynx2.mem linux_apihooks
```

Unfortunately, the plugin failed to work correctly and terminated in a series of errors that may have been caused by either an issue with Volatility or an incorrectly implemented/compatible Python library.

3.8.4 Summary

The first two plugins, while useful in certain circumstances, did not detect any injected code or processes that had their privileges elevated. In short, these two plugins may be of use in certain cases but were unhelpful in this investigation.

The third plugin failed to function; as such, little can be said about its ability to aid investigators/incident handlers in their investigations.

3.9 Step 9: Reconsider the type of malware in use

3.9.1 Reconsideration

Thus far, a number of plugins have been brought to bear on this memory image and other than vague references of one possibly hidden kernel module, nothing suspicious or out of the ordinary has been determined, considering that to date a significant amount of information has been sifted through. These plugins have produced lots of information concerning filesystem objects that the investigator/incident handler had to parse in order to identify if something suspicious was at play in memory, but thus far, nothing substantial has been found.

Despite the information presented about this rootkit at the beginning of this report, the details of which an investigator/incident handler will almost never have prior to an investigation, a simple technique can be used to readily determine if the malware that has infected the system is in fact a rootkit, and if so, its type. If the malware were a kernel-level rootkit then the various kernel listing plugins should have found it, even if it were attempting to evade detection. Had code injection (plugin *linux_malfind*) or API hooking (plugin *linux_apihooks*, which failed to work) been detected then the infection could have been attributed to non-rootkit sources; however, this was not the case.

Moreover, no unusual processes were detected nor were any abnormal libraries found to be in use. Thus, based on the information on hand, the infection is neither likely a kernel rootkit nor a userland memory resident infection (e.g., virus, worm, etc.). That leaves a userland rootkit.

3.9.2 About userland rootkits

Consider that Section 3.4.2 explored the use of the *linux_bash_env* plugin that was used to find and display the various environment variables associated with one or more instances of *bash* where nothing was found. Had something been found, as will be shown in the following paragraphs, this would have led an investigator/incident handler to possibly suspect a userland rootkit.

Userland rootkits are compiled library-based executables for Linux and UNIX. The library must contain at least one function that is to supersede the correct function being loaded from the appropriate library file. The superseding function then adds its own code into the mix to subvert the programmer's original intent. Nevertheless, all that is needed is a compiled rootkit-based library, which Jynx2 provides, as per its libraries, *jynx2.so* and *reality.so* (see Section 1.7 for details).

Userland rootkits are typically loaded in one of two ways. The first is via the current user's shell where the *LD_PRELOAD* environment variable is set to some arbitrary library that may (or may not) be legitimate. For example, running this command would load a library-based rootkit into the process space of command *ls* in so long as the program was running:

```
$ LD_PRELOAD=/path/malicious/library.so /bin/ls -alR
```

To remain in memory after the command has terminated the rootkit would need to subvert some system process using code injection, process hollowing or hooking, assuming the user is not root. If the user were root then process subversion should be easier.

However, configuring the user shell environment as follows would permit the rootkit to be loaded with every program run from the user's shell:

```
$ export LD_PRELOAD=/path/to/malicious/library.so
$ ./myprogram.elf
```

This variation will result in a semi-persistent state of infection. All that is required is that the shell remain open, the LD_PRELOAD variable is not modified and that various commands or scripts are executed back-to-back or regularly from within the shell.

Of course, advanced userland rootkits would be able to detect the presence of duplicate rootkits running in order to ensure that only one copy runs at any given time. Moreover, some rootkits may have the ability to jump into the process space of other processes (using injection, process hollowing or hooking) but it is not known if this type of userland rootkit yet exists under Linux.

The second method is used for maintaining a persistent infection. A line must be appended to the beginning of system file */etc/ld.so.preload*. For example, consider the following:

```
/path/malicious/library.so
```

A reboot should not be necessary for this change to take effect.

It is important to point out that LD_PRELOAD based rootkits are effective only against C and C++ based programs. This technique, which is actually one form of API hooking, will not work against assembly and other non-C/C++ compiled programs.

Finally, a less obvious manner of maintaining system persistence, but which is less noticeable to system administrators, is for the attacker to subvert the system into allowing him to modify a system initialization script where the LD_PRELOAD can be set for that script only. This would permit the process/daemon instantiated from the script to maintain the infection for the duration of the system's uptime. Typical Linux initialization scripts are updated and upgraded on a regular basis in order to maintain system security through the application of approved patches, updates and upgrades. In contrast to system file */etc/ld.so.preload*, it is rarely modified and due to the very few lines contained therein, it is typically easy to recognize that something is amiss.

3.9.3 Validating out ld.so.preload

In order to find and extract file */etc/ld.so.preload* from the memory image, assuming it is intact within the system's filesystem object cache, it suffices to consult again with the *linux_find_file* plugin (see Section 3.5.4). It suffices to rerun this plugin while "grepping" for keyword "ld.so.preload." This will result in the following output:

```
1049885  0xf4c65938  /etc/ld.so.preload
```

Armed with this information, it is possible to dump this file from the memory image using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_jynx2.mem linux_find_file -i 0xf4c65938 -O ld.so.preload

$       cat ld.so.preload

        /XxJynx/jynx2.so
```

Thus, after having worked through all the various plugins and running a few relatively straightforward commands, it is possible to definitively identify the source of the infection which has not yet taken hold but which has been configured for persistence. This library-based rootkit will survive multiple reboots and upon its first post-infection reboot, it is very likely that all the operating system's processes will be infected, which may lead to an uncoordinated infection, unless the rootkit has the ability to terminate other instances of itself or coordinate its work among its duplicates.

3.9.4 Summary

This step provided important additional information concerning the nature of userland rootkits with respect to system modifications, both at the level of the system's configuration and the user's environment. If a userland rootkit was launched from either the command line shell or script, an attentive investigator will be able to find and isolate the appropriate environment variables. On the other hand, if it was launched via */etc/ld.so.preload*, then the system may be teeming with multiple infections.

4 Conclusion

Linux userland-based rootkits stand in stark contrast to kernel module rootkits (LKM) with respect to their instantiation and system persistence. Userland rootkits are not necessarily less advanced than LKM rootkits, although Jynx2 is by far not as advanced as it could have been. Of course, Linux rootkits do not receive anywhere the malicious developmental effort Windows-based rootkits do.

It is difficult to compare Jynx2 to KBeast or to the capabilities of other Linux rootkits as they are less commonly encountered. In contrast, Windows rootkits are relatively common and there is a wide range in their capabilities, some of which are far less capable than Jynx2 while others are tremendously sophisticated both in terms of their ability to hide and in terms of their overall set of features. That said, based on the current capabilities of Jynx2, it could be considered as average.

It was not possible to detect Jynx2 in this memory image because it had not yet been loaded into memory through the launching or instantiation of additional user or system processes. However, the second part of this case will do just that – examine what userland rootkit persistence use looks like.

This report has shown investigators/incident handlers what to look for when the majority of useful (and non-reverse engineering) Volatility plugins have been exhausted and turned up empty – that is to say no evidence of malware infection. However, searching for traces of tampering with */etc/ld.so.preload* can sometimes provide insight into what may have been left behind after network or system infiltration.

It was also rather surprising that at the time the rootkit libraries files were submitted to VirusTotal, not one of the 53 scanners detected either as malicious or infected. This is somewhat shocking considering that the rootkit is nearly two years old and its source code is available for anyone to modify or use.

Finally, this case study will have hopefully demonstrated to investigators/incident handlers how to systematically proceed with investigating a suspected Linux-based memory image and determine if it has been infected or set up for use by a userland rootkit.

References

[1] Blackhat Academy. Jynx Rootkit/2.0. Online documentation. BlackHat Academy. November 2013. http://www.blackhatlibrary.net/Jynx_Rootkit/2.0.

[2] Blackhat Academy. LD Preload. Online documentation. BlackHat Academy. September 2012. http://www.blackhatlibrary.net/LD_Preload.

[3] Blackhat Academy. Jynx2 Sneak Peek & Analysis. Online documentation. BlackHat Academy. March 2012. http://resources.infosecinstitute.com/jynx2-sneak-peek-analysis/.

[4] Vandeven, Sally. Linux Rootkit Detecgtion With OSSEC. White paper. SANS.org. March 2014. http://www.sans.org/reading-room/whitepapers/detection/rootkit-detection-ossec-34555.

[5] Blackhat Academy. Jynx/2.0/PAM. Online documentation. BlackHat Academy. June 2013. http://www.blackhatlibrary.net/Hooking_PAM.

[6] Volatility team. LinuxMemoryForensics: Instructions on how to access and use the Linux support. Online documentation. September 2013. http://code.google.com/p/volatility/wiki/LinuxMemoryForensics.

[7] Carbone, Richard. Malware memory analysis of the KBeast rootkit: Investigating publicly available Linux rootkits using the Volatility memory analysis framework. Scientific Report. DRDC-RDDC-2015-RXXX. DRDC – Valcartier Research Centre. September 2014.

[8] YobiWiki. RAM analysis: Misc notes on physical RAM analysis. Online documentation. October 2013. http://wiki.yobi.be/wiki/RAM_analysis.

[9] Hale Ligh, Michael; Case, Andrew et al. The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Max Memory. Book. John Wiley & Sons. July 2014.

[10] Volatility. LinuxCommandReference23: A command reference for Linux. Unknown date. http://code.google.com/p/volatility/wiki/LinuxCommandReference23.

[11] Kim, Joonsoo. How does the SLUB allocator work. Presentation. LGE CTO SWP Lab. Unknown date. http://events.linuxfoundation.org/images/stories/pdf/klf2012_kim.pdf.

[12] Wikipedia. Slab allocation. Online encyclopaedic entry. Wikimedia Foundation Inc. July 2014. http://en.wikipedia.org/wiki/Slab_allocation.

[13] Wikipedia. SLUB (software). Online encyclopaedic entry. Wikimedia Foundation Inc. August 2014. http://en.wikipedia.org/wiki/SLUB_(software).

[14] Wikipedia. Process identifier. Online encyclopaedic entry. Wikimedia Foundation Inc. April 2014. http://en.wikipedia.org/wiki/Process_identifier.

[15] Wikipedia. Multicast address. Online encyclopaedic entry. Wikimedia Foundation Inc. August 2014.http://en.wikipedia.org/wiki/Multicast_address.

This page intentionally left blank.

Annex A Volatility Linux-based plugins

The following is a complete list of the default Linux-based plugins provided by Volatility 2.3.1:

Table A.1: List of Volatility 2.3.1 plugins.

Plugin	**Capability (as per Volatility --info output)**
linux_arp	Print the ARP table
linux_banner	Prints the Linux banner information
linux_bash	Recover bash history from bash process memory
linux_check_afinfo	Verifies the operation function pointers of network protocols
linux_check_creds	Checks if any processes are sharing credential structures
linux_check_evt_arm	Checks the Exception Vector Table to look for syscall table hooking
linux_check_fop	Check file operation structures for rootkit modifications
linux_check_idt	Checks if the IDT has been altered
linux_check_modules	Compares module list to sysfs info, if available
linux_check_syscall	Checks if the system call table has been altered
linux_check_syscall_arm	Checks if the system call table has been altered
linux_check_tty	Checks tty devices for hooks
linux_cpuinfo	Prints info about each active processor
linux_dentry_cache	Gather files from the dentry cache
linux_dmesg	Gather dmesg buffer
linux_dump_map	Writes selected memory mappings to disk
linux_find_file	Recovers tmpfs filesystems from memory
linux_ifconfig	Gathers active interfaces
linux_iomem	Provides output similar to /proc/iomem
linux_keyboard_notifiers	Parses the keyboard notifier call chain
linux_lsmod	Gather loaded kernel modules
linux_lsof	Lists open files

Plugin	**Capability (as per Volatility --info output)**
linux_memmap	Dumps the memory map for linux tasks
linux_moddump	Extract loaded kernel modules
linux_mount	Gather mounted fs/devices
linux_mount_cache	Gather mounted fs/devices from kmem_cache
linux_netstat	Lists open sockets
linux_pidhashtable	Enumerates processes through the PID hash table
linux_pkt_queues	Writes per-process packet queues out to disk
linux_proc_maps	Gathers process maps for linux
linux_psaux	Gathers processes along with full command line and start time
linux_pslist	Gather active tasks by walking the task_struct->task list
linux_pslist_cache	Gather tasks from the kmem_cache
linux_pstree	Shows the parent/child relationship between processes
linux_psxview	Find hidden processes with various process listings
linux_route_cache	Recovers the routing cache from memory
linux_sk_buff_cache	Recovers packets from the sk_buff kmem_cache
linux_slabinfo	Mimics /proc/slabinfo on a running machine
linux_tmpfs	Recovers tmpfs filesystems from memory
linux_vma_cache	Gather VMAs from the vm_area_struct cache
linux_volshell	Shell in the memory image
linux_yarascan	A shell in the Linux memory image

Annex B Plugin output and listings

This annex provides the various output and listings for the different plugins used throughout this report which are too lengthy to fit within a given subsection.

B.1 Output for plugin linux_dmesg

The following output was generated by the Volatility *linux_dmesg* plugin, as found in Section 3.4.3:

```
[2314885531810281020.2314885531] ] Initializing cgroup subsys cpuset
<6>[    0.000000] Initializing cgroup subsys cpu
<5>[    0.000000] Linux version 2.6.35-22-generic (buildd@rothera) (gcc
version 4.4.5 (Ubuntu/Linaro 4.4.4-14ubuntu4) ) #33-Ubuntu SMP Sun Sep 19
20:34:50 UTC 2010 (Ubuntu 2.6.35-22.33-generic 2.6.35.4)
<6>[    0.000000] BIOS-provided physical RAM map:
<6>[    0.000000]  BIOS-e820: 0000000000000000 - 000000000009fc00
(usable)
<6>[    0.000000]  BIOS-e820: 000000000009fc00 - 00000000000a0000
(reserved)
<6>[    0.000000]  BIOS-e820: 00000000000f0000 - 0000000000100000
(reserved)
<6>[    0.000000]  BIOS-e820: 0000000000100000 - 000000007fff0000
(usable)
<6>[    0.000000]  BIOS-e820: 000000007fff0000 - 0000000080000000 (ACPI
data)
<6>[    0.000000]  BIOS-e820: 00000000fffc0000 - 0000000100000000
(reserved)
<5>[    0.000000] Notice: NX (Execute Disable) protection cannot be
enabled: non-PAE kernel!
<6>[    0.000000] DMI 2.5 present.
<7>[    0.000000] e820 update range: 0000000000000000 - 0000000000001000
(usable) ==> (reserved)
<7>[    0.000000] e820 remove range: 00000000000a0000 - 0000000000100000
(usable)
<6>[    0.000000] last_pfn = 0x7fff0 max_arch_pfn = 0x100000
<7>[    0.000000] MTRR default type: uncachable
<7>[    0.000000] MTRR variable ranges disabled:
<6>[    0.000000] x86 PAT enabled: cpu 0, old 0x7040600070406, new
0x7010600070106
<6>[    0.000000] CPU MTRRs all blank - virtualized system.
<7>[    0.000000] e820 update range: 0000000000002000 - 0000000000010000
(usable) ==> (reserved)
<6>[    0.000000] Scanning 1 areas for low memory corruption
<6>[    0.000000] modified physical RAM map:
<6>[    0.000000]  modified: 0000000000000000 - 0000000000001000
(reserved)
<6>[    0.000000]  modified: 0000000000001000 - 0000000000002000 (usable)
<6>[    0.000000]  modified: 0000000000002000 - 0000000000010000
(reserved)
<6>[    0.000000]  modified: 0000000000010000 - 000000000009fc00 (usable)
<6>[    0.000000]  modified: 000000000009fc00 - 00000000000a0000
(reserved)
<6>[    0.000000]  modified: 00000000000f0000 - 0000000000100000
(reserved)
<6>[    0.000000]  modified: 0000000000100000 - 000000007fff0000 (usable)
<6>[    0.000000]  modified: 000000007fff0000 - 0000000080000000 (ACPI
data)
<6>[    0.000000]  modified: 00000000fffc0000 - 0000000100000000
(reserved)
<7>[    0.000000] initial memory mapped : 0 - 00c00000
```

```
<6>[    0.000000] init_memory_mapping: 0000000000000000-00000000377fe000
<7>[    0.000000]  0000000000 - 0000400000 page 4k
<7>[    0.000000]  0000400000 - 0037400000 page 2M
<7>[    0.000000]  0037400000 - 00377fe000 page 4k
<7>[    0.000000] kernel direct mapping tables up to 377fe000 @ 15000-
1a000
<6>[    0.000000] RAMDISK: 375ad000 - 37ff0000
<6>[    0.000000] Allocated new RAMDISK: 009a5000 - 013e78c9
<6>[    0.000000] Move RAMDISK from 00000000375ad000 - 0000000037fef8c8
to 009a5000 - 013e78c8
<4>[    0.000000] ACPI: RSDP 000e0000 00024 (v02 VBOX  )
<4>[    0.000000] ACPI: XSDT 7fff0030 00034 (v01 VBOX   VBOXXSDT 00000001
ASL  00000061)
<4>[    0.000000] ACPI: FACP 7fff00f0 000F4 (v04 VBOX   VBOXFACP 00000001
ASL  00000061)
<4>[    0.000000] ACPI: DSDT 7fff0410 01B96 (v01 VBOX   VBOXBIOS 00000002
INTL 20100528)
<4>[    0.000000] ACPI: FACS 7fff0200 00040
<4>[    0.000000] ACPI: SSDT 7fff0240 001CC (v01 VBOX   VBOXCPUT 00000002
INTL 20100528)
<5>[    0.000000] 1159MB HIGHMEM available.
<5>[    0.000000] 887MB LOWMEM available.
<6>[    0.000000]   mapped low ram: 0 - 377fe000
<6>[    0.000000]   low ram: 0 - 377fe000
<4>[    0.000000] Zone PFN ranges:
<4>[    0.000000]   DMA      0x00000001 -> 0x00001000
<4>[    0.000000]   Normal   0x00001000 -> 0x000377fe
<4>[    0.000000]   HighMem  0x000377fe -> 0x0007fff0
<4>[    0.000000] Movable zone start PFN for each node
<4>[    0.000000] early_node_map[3] active PFN ranges
<4>[    0.000000]     0: 0x00000001 -> 0x00000002
<4>[    0.000000]     0: 0x00000010 -> 0x0000009f
<4>[    0.000000]     0: 0x00000100 -> 0x0007fff0
<7>[    0.000000] On node 0 totalpages: 524160
<7>[    0.000000]  free_area_init_node:  node  0,  pgdat  c07ffd40,
node_mem_map c13e9020
<7>[    0.000000]   DMA zone: 32 pages used for memmap
<7>[    0.000000]   DMA zone: 0 pages reserved
<7>[    0.000000]   DMA zone: 3952 pages, LIFO batch:0
<7>[    0.000000]   Normal zone: 1744 pages used for memmap
<7>[    0.000000]   Normal zone: 221486 pages, LIFO batch:31
<7>[    0.000000]   HighMem zone: 2320 pages used for memmap
<7>[    0.000000]   HighMem zone: 294626 pages, LIFO batch:31
<6>[    0.000000] Using APIC driver default
<6>[    0.000000] ACPI: PM-Timer IO Port: 0x4008
<6>[    0.000000] SMP: Allowing 1 CPUs, 0 hotplug CPUs
<6>[    0.000000] Found and enabled local APIC!
<7>[    0.000000] nr_irqs_gsi: 16
<6>[    0.000000]  PM:  Registered  nosave  memory:  0000000000002000  -
0000000000010000
<6>[    0.000000]  PM:  Registered  nosave  memory:  000000000009f000  -
00000000000a0000
<6>[    0.000000]  PM:  Registered  nosave  memory:  00000000000a0000  -
00000000000f0000
<6>[    0.000000]  PM:  Registered  nosave  memory:  00000000000f0000  -
0000000000100000
<6>[    0.000000] Allocating PCI resources starting at 80000000 (gap:
80000000:7ffc0000)
<6>[    0.000000] Booting paravirtualized kernel on bare hardware
<6>[    0.000000] setup_percpu: NR_CPUS:8 nr_cpumask_bits:8 nr_cpu_ids:1
nr_node_ids:1
<7>[    0.000000] early_res array is doubled to 64 at [16000 - 167ff]
<6>[    0.000000]  PERCPU:  Embedded  14  pages/cpu  @c2400000  s36416  r0
d20928 u4194304
<6>[    0.000000] pcpu-alloc: s36416 r0 d20928 u4194304 alloc=1*4194304
<6>[    0.000000] pcpu-alloc: [0] 0
<4>[    0.000000] Built 1 zonelists in Zone order, mobility grouping on.
Total pages: 520064
```

```
<5>[    0.000000] Kernel command line: BOOT_IMAGE=/boot/vmlinuz-2.6.35-
22-generic root=UUID=b13dedba-11eb-497f-96b2-e06d37b3aef1 ro quiet splash
<6>[    0.000000] PID hash table entries: 4096 (order: 2, 16384 bytes)
<6>[    0.000000] Dentry cache hash table entries: 131072 (order: 7,
524288 bytes)
<6>[    0.000000] Inode-cache hash table entries: 65536 (order: 6, 262144
bytes)
<6>[    0.000000] Enabling fast FPU save and restore... done.
<6>[    0.000000] Enabling unmasked SIMD FPU exception support... done.
<6>[    0.000000] Initializing CPU#0
<6>[    0.000000] allocated 10485420 bytes of page_cgroup
<6>[    0.000000] please try 'cgroup_disable=memory' option if you don't
want memory cgroups
<6>[    0.000000] Subtract (39 early reservations)
<6>[    0.000000]   #1 [0000001000 - 0000002000]   EX TRAMPOLINE
<6>[    0.000000]   #2 [0000100000 - 00009a0adc]   TEXT DATA BSS
<6>[    0.000000]   #3 [000009f800 - 0000100000]   BIOS reserved
<6>[    0.000000]   #4 [00009a1000 - 00009a410c]             BRK
<6>[    0.000000]   #5 [0000010000 - 0000011000]      TRAMPOLINE
<6>[    0.000000]   #6 [0000011000 - 0000015000]     ACPI WAKEUP
<6>[    0.000000]   #7 [0000015000 - 0000016000]         PGTABLE
<6>[    0.000000]   #8 [00009a5000 - 00013e8000]     NEW RAMDISK
<6>[    0.000000]   #9 [00013e8000 - 00013e9000]         BOOTMEM
<6>[    0.000000]   #10 [00013e9000 - 00023e9000]         BOOTMEM
<6>[    0.000000]   #11 [00023e9000 - 00023e9004]         BOOTMEM
<6>[    0.000000]   #12 [00023e9040 - 00023e9100]         BOOTMEM
<6>[    0.000000]   #13 [00023e9100 - 00023e9154]         BOOTMEM
<6>[    0.000000]   #14 [00023e9180 - 00023ec180]         BOOTMEM
<6>[    0.000000]   #15 [00023ec180 - 00023ec1f0]         BOOTMEM
<6>[    0.000000]   #16 [00023ec200 - 00023f2200]         BOOTMEM
<6>[    0.000000]   #17 [00023f2200 - 00023f22fc]         BOOTMEM
<6>[    0.000000]   #18 [00023f2300 - 00023f2340]         BOOTMEM
<6>[    0.000000]   #19 [00023f2340 - 00023f2380]         BOOTMEM
<6>[    0.000000]   #20 [00023f2380 - 00023f23c0]         BOOTMEM
<6>[    0.000000]   #21 [00023f23c0 - 00023f2400]         BOOTMEM
<6>[    0.000000]   #22 [00023f2400 - 00023f2440]         BOOTMEM
<6>[    0.000000]   #23 [00023f2440 - 00023f2480]         BOOTMEM
<6>[    0.000000]   #24 [00023f2480 - 00023f2490]         BOOTMEM
<6>[    0.000000]   #25 [00023f24c0 - 00023f24d0]         BOOTMEM
<6>[    0.000000]   #26 [00023f2500 - 00023f256a]         BOOTMEM
<6>[    0.000000]   #27 [00023f2580 - 00023f25ea]         BOOTMEM
<6>[    0.000000]   #28 [0002400000 - 000240e000]         BOOTMEM
<6>[    0.000000]   #29 [00023f4600 - 00023f4604]         BOOTMEM
<6>[    0.000000]   #30 [00023f4640 - 00023f4644]         BOOTMEM
<6>[    0.000000]   #31 [00023f4680 - 00023f4684]         BOOTMEM
<6>[    0.000000]   #32 [00023f46c0 - 00023f46c4]         BOOTMEM
<6>[    0.000000]   #33 [00023f4700 - 00023f47b0]         BOOTMEM
<6>[    0.000000]   #34 [00023f47c0 - 00023f4868]         BOOTMEM
<6>[    0.000000]   #35 [00023f4880 - 00023f8880]         BOOTMEM
<6>[    0.000000]   #36 [000240e000 - 000248e000]         BOOTMEM
<6>[    0.000000]   #37 [000248e000 - 00024ce000]         BOOTMEM
<6>[    0.000000]   #38 [00024ce000 - 0002ecdeac]         BOOTMEM
<6>[    0.000000] Initializing HighMem for node 0 (000377fe:0007fff0)
<6>[    0.000000] Memory: 2049740k/2097088k available (4928k kernel code,
46900k reserved, 2336k data, 684k init, 1187784k highmem)
<6>[    0.000000] virtual kernel memory layout:
<6>[    0.000000]     fixmap  : 0xfff16000 - 0xfffff000   ( 932 kB)
<6>[    0.000000]     pkmap   : 0xff800000 - 0xffc00000   (4096 kB)
<6>[    0.000000]     vmalloc : 0xf7ffe000 - 0xff7fe000   ( 120 MB)
<6>[    0.000000]     lowmem  : 0xc0000000 - 0xf77fe000   ( 887 MB)
<6>[    0.000000]       .init : 0xc0819000 - 0xc08c4000   ( 684 kB)
<6>[    0.000000]       .data : 0xc05d029e - 0xc0818668   (2336 kB)
<6>[    0.000000]       .text : 0xc0100000 - 0xc05d029e   (4928 kB)
<6>[    0.000000] Checking if this processor honours the WP bit even in
supervisor mode...Ok.
<6>[    0.000000] SLUB: Genslabs=13, HWalign=64, Order=0-3, MinObjects=0,
CPUs=1, Nodes=1
<6>[    0.000000] Hierarchical RCU implementation.
```

```
<6>[    0.000000]   RCU dyntick-idle grace-period acceleration is
enabled.
<6>[    0.000000]   RCU-based detection of stalled CPUs is disabled.
<6>[    0.000000]   Verbose stalled-CPUs detection is disabled.
<6>[    0.000000] NR_IRQS:2304 nr_irqs:256
<4>[    0.000000] Console: colour VGA+ 80x25
<6>[    0.000000] console [tty0] enabled
<4>[    0.000000] Fast TSC calibration using PIT
<4>[    0.000000] Detected 2395.693 MHz processor.
<6>[    0.004002] Calibrating delay loop (skipped), value calculated
using timer frequency.. 4791.38 BogoMIPS (lpj=9582772)
<6>[    0.004005] pid_max: default: 32768 minimum: 301
<6>[    0.004017] Security Framework initialized
<6>[    0.004026] AppArmor: AppArmor initialized
<6>[    0.004027] Yama: becoming mindful.
<4>[    0.004055] Mount-cache hash table entries: 512
<6>[    0.004108] Initializing cgroup subsys ns
<6>[    0.004110] Initializing cgroup subsys cpuacct
<6>[    0.004112] Initializing cgroup subsys memory
<6>[    0.004116] Initializing cgroup subsys devices
<6>[    0.004117] Initializing cgroup subsys freezer
<6>[    0.004118] Initializing cgroup subsys net_cls
<6>[    0.004165] mce: CPU supports 0 MCE banks
<6>[    0.004179] using mwait in idle threads.
<6>[    0.004182] Performance Events: unsupported p6 CPU model 69 no PMU
driver, software events only.
<6>[    0.007885] SMP alternatives: switching to UP code
<6>[    0.019384] Freeing SMP alternatives: 24k freed
<6>[    0.019407] ACPI: Core revision 20100428
<4>[    0.021649] ACPI: setting ELCR to 0200 (from 0e00)
<6>[    0.022233] ftrace: converting mcount calls to 0f 1f 44 00 00
<6>[    0.022355] ftrace: allocating 21758 entries in 43 pages
<6>[    0.036025] Enabling APIC mode:  Flat.  Using 0 I/O APICs
<4>[    0.036035] weird, boot CPU (#0) not listed by the BIOS.
<5>[    0.036039] SMP motherboard not detected.
<6>[    0.040000] SMP disabled
<6>[    0.040000] Brought up 1 CPUs
<6>[    0.040000] Total of 1 processors activated (4791.38 BogoMIPS).
<6>[    0.040000] devtmpfs: initialized
<6>[    0.040000] regulator: core version 0.5
<4>[    0.040000] Time:  1:00:49  Date: 05/24/14
<6>[    0.040000] NET: Registered protocol family 16
<6>[    0.040000] EISA bus registered
<6>[    0.040000] ACPI: bus type pci registered
<6>[    0.040000] PCI: PCI BIOS revision 2.10 entry at 0xfda26, last
bus=0
<6>[    0.040000] PCI: Using configuration type 1 for base access
<4>[    0.040000] bio: create slab <bio-0> at 0
<7>[    0.040000] ACPI: EC: Look up EC in DSDT
<4>
[8027507190610277166.8027507190] 6>[    0.040000] devtmpfs: initialized
<6>[    0.040000] regulator: core version 0.5
<4>[    0.040000] Time:  1:00:49  Date: 05/24/14
<6>[    0.040000] NET: Registered protocol family 16
<6>[    0.040000] EISA bus registered
<6>[    0.040000] ACPI: bus type pci registered
<6>[    0.040000] PCI: PCI BIOS revision 2.10 entry at 0xfda26, last
bus=0
<6>[    0.040000] PCI: Using configuration type 1 for base access
<4>[    0.040000] bio: create slab <bio-0> at 0
<7>[    0.040000] ACPI: EC: Look up EC in DSDT
<4>[    0.040000] ACPI: Executed 1 blocks of module-level executable AML
code
<6>[    0.040000] ACPI: Interpreter enabled
<6>[    0.040000] ACPI: (supports S0 S5)
<6>[    0.040000] ACPI: Using PIC for interrupt routing
<6>[    0.040412] ACPI: No dock devices found.
```

```
<6>[    0.040415] PCI: Ignoring host bridge windows from ACPI; if necessary, use "pci=use_crs" and report a bug
<6>[    0.040453] ACPI: PCI Root Bridge [PCI0] (domain 0000 [bus 00-ff])
<7>[    0.040518] pci_root PNP0A03:00: host bridge window [io  0x0000-0x0cf7] (ignored)
<7>[    0.040519] pci_root PNP0A03:00: host bridge window [io  0x0d00-0xffff] (ignored)
<7>[    0.040521] pci_root PNP0A03:00: host bridge window [mem 0x000a0000-0x000bffff] (ignored)
<7>[    0.040522] pci_root PNP0A03:00: host bridge window [mem 0x80000000-0xffdfffff] (ignored)
<7>[    0.041177] pci 0000:00:01.1: reg 20: [io  0xd000-0xd00f]
<7>[    0.044020] pci 0000:00:02.0: reg 10: [mem 0xe0000000-0xe7ffffff pref]
<7>[    0.045364] pci 0000:00:03.0: reg 10: [mem 0xf0000000-0xf001ffff]
<7>[    0.045421] pci 0000:00:03.0: reg 18: [io  0xd010-0xd017]
<7>[    0.045728] pci 0000:00:04.0: reg 10: [io  0xd020-0xd03f]
<7>[    0.048051] pci 0000:00:04.0: reg 14: [mem 0xf0400000-0xf07fffff]
<7>[    0.049048] pci 0000:00:04.0: reg 18: [mem 0xf0800000-0xf0803fff pref]
<7>[    0.049262] pci 0000:00:05.0: reg 10: [io  0xd100-0xd1ff]
<7>[    0.049286] pci 0000:00:05.0: reg 14: [io  0xd200-0xd23f]
<7>[    0.050464] pci 0000:00:06.0: reg 10: [mem 0xf0804000-0xf0804fff]
<7>[    0.052190] pci 0000:00:0b.0: reg 10: [mem 0xf0805000-0xf0805fff]
<7>[    0.052750] pci 0000:00:0d.0: reg 10: [io  0xd240-0xd247]
<7>[    0.052806] pci 0000:00:0d.0: reg 18: [io  0xd250-0xd257]
<7>[    0.052859] pci 0000:00:0d.0: reg 20: [io  0xd260-0xd26f]
<7>[    0.053844] pci 0000:00:0d.0: reg 24: [mem 0xf0806000-0xf0807fff]
<7>[    0.054070] pci_bus 0000:00: on NUMA node 0
<7>[    0.054090] ACPI: PCI Interrupt Routing Table [\_SB_.PCI0._PRT]
<6>[    0.057663] ACPI: PCI Interrupt Link [LNKA] (IRQs 5 9 10 *11)
<6>[    0.057812] ACPI: PCI Interrupt Link [LNKB] (IRQs 5 9 10 *11)
<6>[    0.057864] ACPI: PCI Interrupt Link [LNKC] (IRQs 5 9 *10 11)
<6>[    0.057915] ACPI: PCI Interrupt Link [LNKD] (IRQs 5 *9 10 11)
<6>[    0.057945] HEST: Table is not found!
<6>[    0.058010] vgaarb: device added: PCI:0000:00:02.0,decodes=io+mem,owns=io+mem,locks=none
<6>[    0.058012] vgaarb: loaded
<5>[    0.058109] SCSI subsystem initialized
<7>[    0.058161] libata version 3.00 loaded.
<6>[    0.058196] usbcore: registered new interface driver usbfs
<6>[    0.058202] usbcore: registered new interface driver hub
<6>[    0.058222] usbcore: registered new device driver usb
<6>[    0.058275] ACPI: WMI: Mapper loaded
<6>[    0.058278] PCI: Using ACPI for IRQ routing
<7>[    0.058279] PCI: pci_cache_line_size set to 64 bytes
<7>[    0.058439] reserve RAM buffer: 0000000000002000 - 000000000000ffff
<7>[    0.058442] reserve RAM buffer: 000000000009fc00 - 000000000009ffff
<7>[    0.058444] reserve RAM buffer: 000000007fff0000 - 000000007fffffff
<6>[    0.058498] NetLabel: Initializing
<6>[    0.058499] NetLabel:  domain hash size = 128
<6>[    0.058500] NetLabel:  protocols = UNLABELED CIPSOv4
<6>[    0.058507] NetLabel:  unlabeled traffic allowed by default
<6>[    0.058525] Switching to clocksource tsc
<6>[    0.061595] AppArmor: AppArmor Filesystem Enabled
<6>[    0.061603] pnp: PnP ACPI init
<6>[    0.061612] ACPI: bus type pnp registered
<3>[    0.061688] ERROR: Unable to locate IOAPIC for GSI 1
<3>[    0.061737] ERROR: Unable to locate IOAPIC for GSI 12
<3>[    0.061756] ERROR: Unable to locate IOAPIC for GSI 7
<6>[    0.062096] pnp: PnP ACPI: found 5 devices
<6>[    0.062097] ACPI: ACPI bus type pnp unregistered
<6>[    0.062099] PnPBIOS: Disabled by ACPI PNP
<7>[    0.097230] pci_bus 0000:00: resource 0 [io  0x0000-0xffff]
<7>[    0.097232] pci_bus 0000:00: resource 1 [mem 0x00000000-0xffffffff]
<6>[    0.097251] NET: Registered protocol family 2
<6>[    0.097281] IP route cache hash table entries: 32768 (order: 5, 131072 bytes)
```

```
<6>[    0.097399] TCP established hash table entries: 131072 (order: 8,
1048576 bytes)
<6>[    0.097554] TCP bind hash table entries: 65536 (order: 7, 524288
bytes)
<6>[    0.097613] TCP: Hash tables configured (established 131072 bind
65536)
<6>[    0.097614] TCP reno registered
<6>[    0.097615] UDP hash table entries: 512 (order: 2, 16384 bytes)
<6>[    0.097618] UDP-Lite hash table entries: 512 (order: 2, 16384
bytes)
<6>[    0.097657] NET: Registered protocol family 1
<6>[    0.097663] pci 0000:00:00.0: Limiting direct PCI/PCI transfers
<6>[    0.097682] pci 0000:00:01.0: Activating ISA DMA hang workarounds
<7>[    0.097702] pci 0000:00:02.0: Boot video device
<7>[    0.097810] PCI: CLS 0 bytes, default 64
<6>[    0.097853] platform rtc_cmos: registered platform RTC device (no
PNP device found)
<6>[    0.097900] cpufreq-nforce2: No nForce2 chipset.
<6>[    0.097909] Scanning for low memory corruption every 60 seconds
<6>[    0.097995] audit: initializing netlink socket (disabled)
<5>[    0.098000] type=2000 audit(1400893249.096:1): initialized
<6>[    0.104893] Trying to unpack rootfs image as initramfs...
<4>[    0.114003] highmem bounce pool size: 64 pages
<6>[    0.114007] HugeTLB registered 4 MB page size, pre-allocated 0
pages
<5>[    0.118081] VFS: Disk quotas dquot_6.5.2
<4>[    0.118109] Dquot-cache hash table entries: 1024 (order 0, 4096
bytes)
<6>[    0.121591] fuse init (API version 7.14)
<6>[    0.121638] msgmni has been set to 1683
<6>[    0.130888] Block layer SCSI generic (bsg) driver version 0.4
loaded (major 253)
<6>[    0.130890] io scheduler noop registered
<6>[    0.130892] io scheduler deadline registered
<6>[    0.130899] io scheduler cfq registered (default)
<6>[    0.130951] pci_hotplug: PCI Hot Plug PCI Core version: 0.5
<6>[    0.130961] pciehp: PCI Express Hot Plug Controller Driver version:
0.4
<6>[    0.131075] ACPI: AC Adapter [AC] (on-line)
<6>[    0.131104] input: Power Button as
/devices/LNXSYSTM:00/LNXPWRBN:00/input/input0
<6>[    0.131108] ACPI: Power Button [PWRF]
<6>[    0.131152] input: Sleep Button as
/devices/LNXSYSTM:00/LNXSLPBN:00/input/input1
<6>[    0.131154] ACPI: Sleep Button [SLPF]
<7>[    0.131289] ACPI: acpi_idle registered with cpuidle
<6>[    0.131919] ERST: Table is not found!
<6>[    0.131981] Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
<6>[    0.132642] brd: module loaded
<6>[    0.132872] loop: module loaded
<7>[    0.132948] ata_piix 0000:00:01.1: version 2.13
<7>[    0.132990] ata_piix 0000:00:01.1: setting latency timer to 64
<6>[    0.133047] scsi0 : ata_piix
<6>[    0.133552] ACPI: Battery Slot [BAT0] (battery present)
<6>[    0.133558] isapnp: Scanning for PnP cards...
<6>[    0.141655] scsi1 : ata_piix
<6>[    0.141678] ata1: PATA max UDMA/33 cmd 0x1f0 ctl 0x3f6 bmdma 0xd000
irq 14
<6>[    0.141680] ata2: PATA max UDMA/33 cmd 0x170 ctl 0x376 bmdma 0xd008
irq 15
<6>[    0.141820] Fixed MDIO Bus: probed
<6>[    0.141836] PPP generic driver version 2.4.2
<6>[    0.141856] tun: Universal TUN/TAP device driver, 1.6
<6>[    0.141857] tun: (C) 1999-2004 Max Krasnyansky <maxk@qualcomm.com>
<6>[    0.141890] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI)
Driver
<4>[    0.142101] ACPI: PCI Interrupt Link [LNKC] enabled at IRQ 10
<7>[    0.142103] PCI: setting IRQ 10 as level-triggered
```

```
<6>[    0.142108] ehci_hcd 0000:00:0b.0: PCI INT A -> Link[LNKC] -> GSI
10 (level, low) -> IRQ 10
<7>[    0.142127] ehci_hcd 0000:00:0b.0: setting latency timer to 64
<6>[    0.142134] ehci_hcd 0000:00:0b.0: EHCI Host Controller
<6>[    0.142157] ehci_hcd 0000:00:0b.0: new USB bus registered, assigned
bus number 1
<6>[    0.142297] ehci_hcd 0000:00:0b.0: irq 10, io mem 0xf0805000
<6>[    0.190916] ehci_hcd 0000:00:0b.0: USB 2.0 started, EHCI 1.00
<6>[    0.191004] hub 1-0:1.0: USB hub found
<6>[    0.191006] hub 1-0:1.0: 8 ports detected
<6>[    0.191055] ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
<4>[    0.191275] ACPI: PCI Interrupt Link [LNKB] enabled at IRQ 11
<7>[    0.191277] PCI: setting IRQ 11 as level-triggered
<6>[    0.191282] ohci_hcd 0000:00:06.0: PCI INT A -> Link[LNKB] -> GSI
11 (level, low) -> IRQ 11
<7>[    0.191302] ohci_hcd 0000:00:06.0: setting latency timer to 64
<6>[    0.191309] ohci_hcd 0000:00:06.0: OHCI Host Controller
<6>[    0.191331] ohci_hcd 0000:00:06.0: new USB bus registered, assigned
bus number 2
<6>[    0.191352] ohci_hcd 0000:00:06.0: irq 11, io mem 0xf0804000
<6>[    0.281803] hub 2-0:1.0: USB hub found
<6>[    0.281811] hub 2-0:1.0: 8 ports detected
<6>[    0.281914] uhci_hcd: USB Universal Host Controller Interface
driver
<6>[    0.281957] PNP: PS/2 Controller [PNP0303:PS2K,PNP0f03:PS2M] at
0x60,0x64 irq 1,12
<6>[    0.282479] serio: i8042 KBD port at 0x60,0x64 irq 1
<6>[    0.282483] serio: i8042 AUX port at 0x60,0x64 irq 12
<6>[    0.282516] mice: PS/2 mouse device common for all mice
<6>[    0.282622] rtc_cmos rtc_cmos: rtc core: registered rtc_cmos as
rtc0
<6>[    0.282642] rtc0: alarms up to one day, 114 bytes nvram
<6>[    0.282695] device-mapper: uevent: version 1.0.3
<6>[    0.282980] input: AT Translated Set 2 keyboard as
/devices/platform/i8042/serio0/input/input2
<6>[    0.285462] device-mapper: ioctl: 4.17.0-ioctl (2010-03-05)
initialised: dm-devel@redhat.com
<6>[    0.289447] device-mapper: multipath: version 1.1.1 loaded
<6>[    0.289449] device-mapper: multipath round-robin: version 1.0.0
loaded
<6>[    0.293492] EISA: Probing bus 0 at eisa.0
<4>[    0.293508] Cannot allocate resource for EISA slot 4
<6>[    0.293522] EISA: Detected 0 cards.
<6>[    0.297502] cpuidle: using governor ladder
<6>[    0.297503] cpuidle: using governor menu
<6>[    0.297619] TCP cubic registered
<6>[    0.297683] NET: Registered protocol family 10
<6>[    0.297831] lo: Disabled Privacy Extensions
<6>[    0.297911] NET: Registered protocol family 17
<6>[    0.297934] Using IPI No-Shortcut mode
<7>[    0.297976] PM: Resume from disk failed.
<4>[    0.297983] registered taskstats version 1
<4>[    0.298293]   Magic number: 14:859:3
<6>[    0.298342] rtc_cmos rtc_cmos: setting system clock to 2014-05-24
01:00:49 UTC (1400893249)
<6>[    0.298343] BIOS EDD facility v0.16 2004-Jun-25, 0 devices found
<6>[    0.298344] EDD information not available.
<6>[    0.302541] ata2.00: ATAPI: VBOX CD-ROM, 1.0, max UDMA/133
<6>[    0.302831] ata2.00: configured for UDMA/33
<6>[    0.468317] Freeing initrd memory: 10508k freed
<6>[    0.647573] isapnp: No Plug & Play device found
<5>[    0.647880] scsi 1:0:0:0: CD-ROM            VBOX     CD-ROM
1.0  PQ: 0 ANSI: 5
<4>[    0.648950] sr0: scsi3-mmc drive: 32x/32x xa/form2 tray
<6>[    0.648952] Uniform CD-ROM driver Revision: 3.20
<7>[    0.649009] sr 1:0:0:0: Attached scsi CD-ROM sr0
<5>[    0.649038] sr 1:0:0:0: Attached scsi generic sg0 type 5
<6>[    0.649061] Freeing unused kernel memory: 684k freed
```

```
<6>[    0.649160] Write protecting the kernel text: 4932k
<6>[    0.649176] Write protecting the kernel read-only data: 1976k
<6>[    0.667566] udev[67]: starting version 163
<6>[    0.755460] e1000: Intel(R) PRO/1000 Network Driver - version
7.3.21-k6-NAPI
<6>[    0.755462] e1000: Copyright (c) 1999-2006 Intel Corporation.
<6>[    0.755493] e1000 0000:00:03.0: PCI INT A -> Link[LNKC] -> GSI 10
(level, low) -> IRQ 10
<7>[    0.755509] e1000 0000:00:03.0: setting latency timer to 64
<6>[    1.053204] usb 2-1: new full speed USB device using ohci_hcd and
address 2
<6>[    1.101521] e1000 0000:00:03.0: eth0: (PCI:33MHz:32-bit)
08:00:27:17:bd:56
<6>[    1.101535] e1000 0000:00:03.0: eth0: Intel(R) PRO/1000 Network
Connection
<7>[    1.101572] ahci 0000:00:0d.0: version 3.0
<4>[    1.103446] ACPI: PCI Interrupt Link [LNKA] enabled at IRQ 11
<6>[    1.103457] ahci 0000:00:0d.0: PCI INT A -> Link[LNKA] -> GSI 11
(level, low) -> IRQ 11
<6>[    1.103732] ahci: SSS flag set, parallel bus scan disabled
<6>[    1.103967] ahci 0000:00:0d.0: AHCI 0001.0100 32 slots 1 ports 3
Gbps 0x1 impl SATA mode
<6>[    1.103972] ahci 0000:00:0d.0: flags: 64bit ncq stag only ccc
<7>[    1.104006] ahci 0000:00:0d.0: setting latency timer to 64
<6>[    1.105598] scsi2 : ahci
<6>[    1.105770] ata3: SATA max UDMA/133 abar m8192@0xf0806000 port
0xf0806100 irq 11
<6>[    1.424451] ata3: SATA link up 3.0 Gbps (SStatus 123 SControl 300)
<6>[    1.424888] ata3.00: ATA-8: VBOX HARDDISK, 1.0, max UDMA/133
<6>[    1.424896] ata3.00: 62914560 sectors, multi 128: LBA48 NCQ (depth
31/32)
<6>[    1.425597] ata3.00: configured for UDMA/133
<5>[    1.425794] scsi 2:0:0:0: Direct-Access     ATA      VBOX HARDDISK
1.0  PQ: 0 ANSI: 5
<5>[    1.426057] sd 2:0:0:0: Attached scsi generic sg1 type 0
<5>[    1.426502] sd 2:0:0:0: [sda] 62914560 512-byte logical blocks:
(32.2 GB/30.0 GiB)
<5>[    1.426588] sd 2:0:0:0: [sda] Write Protect is off
<7>[    1.426593] sd 2:0:0:0: [sda] Mode Sense: 00 3a 00 00
<5>[    1.426621] sd 2:0:0:0: [sda] Write cache: enabled, read cache:
enabled, doesn't support DPO or FUA
<6>[    1.426821]  sda: sda1 sda2 < sda5 >
<5>[    1.428804] sd 2:0:0:0: [sda] Attached SCSI disk
<6>[    1.450204] usbcore: registered new interface driver hiddev
<6>[    1.480591] input: VirtualBox USB Tablet as
/devices/pci0000:00/0000:00:06.0/usb2/2-1/2-1:1.0/input/input3
<6>[    1.480719] generic-usb 0003:80EE:0021.0001: input,hidraw0: USB HID
v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-1/input0
<6>[    1.480749] usbcore: registered new interface driver usbhid
<6>[    1.480757] usbhid: USB HID core driver
<6>[    1.497028] EXT4-fs (sda1): mounted filesystem with ordered data
mode. Opts: (null)
<6>[    1.728841] Adding 1340412k swap on /dev/sda5.  Priority:-1
extents:1 across:1340412k SS
<6>[    1.759560] EXT4-fs (sda1): re-mounted. Opts: errors=remount-ro
<6>[    1.783890] udev[256]: starting version 163
<3>[    1.973621] piix4_smbus 0000:00:07.0: SMBus base address
uninitialized - upgrade BIOS or use force_addr=0xaddr
<6>[    2.083577] lp: driver loaded but no devices found
<6>[    2.100328] Linux agpgart interface v0.103
<7>[    2.125662] psmouse serio1: ID: 10 00 64
<6>[    2.157074] input: ImExPS/2 Generic Explorer Mouse as
/devices/platform/i8042/serio1/input/input4
<6>[    2.170344] parport_pc 00:04: reported by Plug and Play ACPI
<6>[    2.208021] ppdev: user-space parallel port driver
<6>[    2.213317] [drm] Initialized drm 1.1.0 20060810
<6>[    2.214108] pci 0000:00:02.0: PCI INT A -> Link[LNKB] -> GSI 11
(level, low) -> IRQ 11
```

```
<7>[    2.214125] pci 0000:00:02.0: setting latency timer to 64
<6>[    2.214236] [drm] Initialized vboxvideo 1.0.0 20090303 for
0000:00:02.0 on minor 0
<4>[    2.235949] ACPI: PCI Interrupt Link [LNKD] enabled at IRQ 9
<7>[    2.235951] PCI: setting IRQ 9 as level-triggered
<6>[    2.235955] vboxguest 0000:00:04.0: PCI INT A -> Link[LNKD] -> GSI
9 (level, low) -> IRQ 9
<6>[    2.237093] input: Unspecified device as
/devices/pci0000:00/0000:00:04.0/input/input5
<4>[    2.237838] vboxguest: major 0, IRQ 9, I/O port d020, MMIO at
00000000f0400000 (size 0x400000)
<7>[    2.237839] vboxguest: Successfully loaded version 4.3.12
(interface 0x00010004)
<5>[    2.301159] type=1400 audit(1400893251.499:2): apparmor="STATUS"
operation="profile_load" name="/sbin/dhclient3" pid=531
comm="apparmor_parser"
<5>[    2.301577] type=1400 audit(1400893251.499:3): apparmor="STATUS"
operation="profile_load" name="/usr/lib/NetworkManager/nm-dhcp-
client.action" pid=531 comm="apparmor_parser"
<5>[    2.301799] type=1400 audit(1400893251.499:4): apparmor="STATUS"
operation="profile_load" name="/usr/lib/connman/scripts/dhclient-script"
pid=531 comm="apparmor_parser"
<6>[    2.405950] Intel ICH 0000:00:05.0: PCI INT A -> Link[LNKA] -> GSI
11 (level, low) -> IRQ 11
<7>[    2.405970] Intel ICH 0000:00:05.0: setting latency timer to 64
<6>[    2.586556] ADDRCONF(NETDEV_UP): eth0: link is not ready
<6>[    2.588703] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow
Control: RX
<6>[    2.589086] ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
<5>[    2.634473] type=1400 audit(1400893251.831:5): apparmor="STATUS"
operation="profile_load" name="/usr/share/gdm/guest-session/Xsession"
pid=666 comm="apparmor_parser"
<5>[    2.637554] type=1400 audit(1400893251.835:6): apparmor="STATUS"
operation="profile_replace" name="/sbin/dhclient3" pid=667
comm="apparmor_parser"
<5>[    2.638045] type=1400 audit(1400893251.835:7): apparmor="STATUS"
operation="profile_replace" name="/usr/lib/NetworkManager/nm-dhcp-
client.action" pid=667 comm="apparmor_parser"
<5>[    2.638279] type=1400 audit(1400893251.835:8): apparmor="STATUS"
operation="profile_replace" name="/usr/lib/connman/scripts/dhclient-
script" pid=667 comm="apparmor_parser"
<5>[    2.645042] type=1400 audit(1400893251.843:9): apparmor="STATUS"
operation="profile_load" name="/usr/bin/evince" pid=669
comm="apparmor_parser"
<5>[    2.651536] type=1400 audit(1400893251.847:10): apparmor="STATUS"
operation="profile_load" name="/usr/bin/evince-previewer" pid=669
comm="apparmor_parser"
<6>[    2.732499] intel8x0_measure_ac97_clock: measured 56144 usecs
(12791 samples)
<6>[    2.732499] intel8x0: measured clock 227824 rejected
<5>[    2.804002] Bridge firewalling registered
<6>[    2.848068] ip_tables: (C) 2000-2006 Netfilter Core Team
<7>[    2.859358] vboxsf: Successfully loaded version 4.3.12 (interface
0x00010004)
<6>[    2.924542] nf_conntrack version 0.5.0 (16384 buckets, 65536 max)
<4>[    2.925251] CONFIG_NF_CT_ACCT is deprecated and will be removed
soon. Please use
<4>[    2.925253] nf_conntrack.acct=1 kernel parameter, acct=1
nf_conntrack module option or
<4>[    2.925254] sysctl net.netfilter.nf_conntrack_acct=1 to enable it.
<6>[    3.092856] intel8x0_measure_ac97_clock: measured 56342 usecs
(13540 samples)
<6>[    3.092858] intel8x0: measured clock 240318 rejected
<6>[    3.280678] EXT4-fs (sda1): re-mounted. Opts: errors=remount-
ro,commit=0
<6>[    3.453206] intel8x0_measure_ac97_clock: measured 56192 usecs
(13555 samples)
<6>[    3.453214] intel8x0: measured clock 241226 rejected
```

```
<6>[    3.453218] intel8x0: clocking to 48000
<4>[        4.229187]  VBoxGuestCommonIOCtl:  HGCM_CALL:  Invalid  handle.
u32Client=b

...
...

<6>[   11.149978] ip6_tables: (C) 2000-2006 Netfilter Core Team
<6>[   11.183184] lo: Disabled Privacy Extensions
<7>[   13.200150] virbr0: no IPv6 routers present
<7>[   13.232191] eth0: no IPv6 routers present
<6>[      30.852215]  EXT4-fs  (sda1):  re-mounted.  Opts:  errors=remount-
ro,commit=0
```

B.2 Output for plugin linux_psaux

The following output was generated by the Volatility *linux_psaux* plugin, as found in Section 3.3.1:

Table B.1: Plugin output for linux_psaux (sorted by PID).

Pid	Uid	Gid	Arguments
1	0	0	/sbin/init ro quiet splash
2	0	0	[kthreadd]
3	0	0	[ksoftirqd/0]
4	0	0	[migration/0]
5	0	0	[watchdog/0]
6	0	0	[events/0]
7	0	0	[cpuset]
8	0	0	[khelper]
9	0	0	[netns]
10	0	0	[async/mgr]
11	0	0	[pm]
12	0	0	[sync_supers]
13	0	0	[bdi-default]
14	0	0	[kintegrityd/0]
15	0	0	[kblockd/0]
16	0	0	[kacpid]
17	0	0	[kacpi_notify]
18	0	0	[kacpi_hotplug]
19	0	0	[ata_aux]
20	0	0	[ata_sff/0]
21	0	0	[khubd]
22	0	0	[kseriod]
23	0	0	[kmmcd]
25	0	0	[khungtaskd]
26	0	0	[kswapd0]
27	0	0	[ksmd]
28	0	0	[aio/0]
29	0	0	[ecryptfs-kthrea]
30	0	0	[crypto/0]
36	0	0	[scsi_eh_0]
38	0	0	[scsi_eh_1]
40	0	0	[kstriped]
41	0	0	[kmpathd/0]
42	0	0	[kmpath_handlerd]

Pid	Uid	Gid	Arguments
43	0	0	[ksnapd]
44	0	0	[kondemand/0]
45	0	0	[kconservative/0]
180	0	0	[scsi_eh_2]
186	0	0	[usbhid_resumer]
200	0	0	[jbd2/sda1-8]
201	0	0	[ext4-dio-unwrit]
251	0	0	upstart-udev-bridge --daemon
256	0	0	udevd --daemon
373	0	0	udevd --daemon
406	0	0	udevd --daemon
435	0	0	[kpsmoused]
494	0	0	[iprt/0]
587	101	103	rsyslogd -c4
620	102	105	dbus-daemon --system --fork
640	0	0	NetworkManager
647	0	0	/usr/sbin/modem-manager
648	104	109	avahi-daemon: ru
650	104	109	avahi-daemon: ch
658	0	0	/sbin/dhclient -d -sf /usr/lib/NetworkManager/nm-dhcp-client.action -pf /var/run/dhclient-eth0.pid -lf /var/lib/dhcp3/dhclient-23f38b76-1743-4bbd-9e2f-f9ba32d635a4-eth0.lease -cf /var/run/nm-dhclient-eth0.conf eth0
663	0	0	/sbin/wpa_supplicant -u -s
707	0	0	/sbin/getty -8 38400 tty4
713	0	0	/sbin/getty -8 38400 tty5
724	0	0	/sbin/getty -8 38400 tty2
726	0	0	/sbin/getty -8 38400 tty3
730	0	0	/sbin/getty -8 38400 tty6
733	0	0	acpid -c /etc/acpi/events -s /var/run/acpid.socket
734	0	0	cron
736	0	0	atd
746	0	0	/usr/sbin/libvirtd -d
844	0	0	/usr/sbin/VBoxService
862	0	0	/usr/sbin/console-kit-daemon --no-daemon
945	65534	30	dnsmasq --strict-order --bind-interfaces --pid-file=/var/run/libvirt/network/default.pid --conf-file= --listen-address 192.168.122.1 --except-interface lo --dhcp-range 192.168.122.2,192.168.122.254 --dhcp-lease-max=253
1018	0	0	[flush-1:0]

Pid	Uid	Gid	Arguments
1019	0	0	[flush-1:1]
1020	0	0	[flush-1:2]
1021	0	0	[flush-1:3]
1022	0	0	[flush-1:4]
1023	0	0	[flush-1:5]
1024	0	0	[flush-1:6]
1025	0	0	[flush-1:7]
1026	0	0	[flush-1:8]
1027	0	0	[flush-1:9]
1028	0	0	[flush-1:10]
1029	0	0	[flush-1:11]
1030	0	0	[flush-1:12]
1031	0	0	[flush-1:13]
1032	0	0	[flush-1:14]
1033	0	0	[flush-1:15]
1034	0	0	[flush-7:0]
1035	0	0	[flush-7:1]
1036	0	0	[flush-7:2]
1037	0	0	[flush-7:3]
1038	0	0	[flush-7:4]
1039	0	0	[flush-7:5]
1040	0	0	[flush-7:6]
1041	0	0	[flush-7:7]
1042	0	0	[flush-8:0]
1077	0	0	/bin/login --
1096	0	0	gdm-binary
1106	0	0	/usr/sbin/cupsd -C /etc/cups/cupsd.conf
1819	0	0	-bash
1842	0	0	/bin/sh /usr/bin/startx
1859	0	0	xinit /etc/X11/xinit/xinitrc -- /etc/X11/xinit/xserverrc :0 -auth /tmp/serverauth.ozl9o3tr2R
1860	0	0	/usr/bin/X -nolisten tcp :0 -auth /tmp/serverauth.ozl9o3tr2R
1863	0	0	/usr/bin/ck-launch-session /usr/bin/dbus-launch --exit-with-session x-session-manager
1906	0	0	/usr/bin/VBoxClient --clipboard
1917	0	0	/usr/bin/VBoxClient --display
1922	0	0	/usr/bin/VBoxClient --seamless
1927	0	0	/usr/bin/ssh-agent /usr/bin/ck-launch-session /usr/bin/dbus-launch --exit-with-session x-session-manager
1928	0	0	/usr/bin/VBoxClient --draganddrop

Pid	Uid	Gid	Arguments
1939	0	0	x-session-manager
1942	0	0	/usr/bin/dbus-launch --exit-with-session x-session-manager
1943	0	0	/bin/dbus-daemon --fork --print-pid 5 --print-address 7 --session
1948	0	0	/usr/lib/libgconf2-4/gconfd-2
1949	0	0	gnome-power-manager
1955	0	0	/usr/bin/gnome-keyring-daemon --start --components=ssh
1960	0	0	/usr/lib/gnome-settings-daemon/gnome-settings-daemon
1964	0	0	/usr/lib/upower/upowerd
1972	0	0	/usr/lib/gvfs/gvfsd
1982	0	0	/usr/lib/gvfs//gvfs-fuse-daemon /root/.gvfs
1987	0	0	bluetooth-applet
1990	0	0	/usr/bin/compiz
1993	0	0	/usr/lib/evolution/2.30/evolution-alarm-notify
1996	0	0	nautilus
1999	0	0	gnome-panel
2002	0	0	nm-applet --sm-disable
2009	0	0	/usr/lib/policykit-1-gnome/polkit-gnome-authentication-agent-1
2013	0	0	/usr/lib/gvfs/gvfs-gdu-volume-monitor
2019	0	0	/usr/lib/policykit-1/polkitd
2028	0	0	/usr/lib/udisks/udisks-daemon
2031	0	0	udisks-daemon: polling /dev/sr
2069	0	0	/usr/lib/gvfs/gvfs-gphoto2-volume-monitor
2083	0	0	/usr/lib/gvfs/gvfs-afc-volume-monitor
2092	0	0	/usr/lib/gvfs/gvfsd-trash --spawner :1.15 /org/gtk/gvfs/exec_spaw/0
2145	0	0	/usr/lib/bonobo-activation/bonobo-activation-server --ac-activate --ior-output-fd=20
2167	0	0	/usr/lib/gnome-panel/wnck-applet --oaf-activate-iid=OAFIID:GNOME_Wncklet_Factory --oaf-ior-fd=22
2168	0	0	/usr/lib/gnome-applets/trashapplet --oaf-activate-iid=OAFIID:GNOME_Panel_TrashApplet_Factory --oaf-ior-fd=28
2186	0	0	/usr/lib/indicator-applet/indicator-applet-session --oaf-activate-iid=OAFIID:GNOME_FastUserSwitchApplet_Factory --oaf-ior-fd=23
2187	0	0	/usr/lib/gnome-panel/clock-applet --oaf-activate-iid=OAFIID:GNOME_ClockApplet_Factory --oaf-ior-fd=32
2188	0	0	/usr/lib/indicator-applet/indicator-applet --oaf-activate-iid=OAFIID:GNOME_IndicatorApplet_Factory --oaf-ior-fd=38
2189	0	0	/usr/lib/gnome-panel/notification-area-applet --oaf-activate-iid=OAFIID:GNOME_NotificationAreaApplet_Factory --oaf-ior-fd=44
2196	0	0	/usr/lib/indicator-sound/indicator-sound-service
2199	0	0	/usr/lib/indicator-messages/indicator-messages-service
2200	0	0	/usr/lib/indicator-application/indicator-application-service

Pid	Uid	Gid	Arguments
2205	0	0	/usr/lib/indicator-session/indicator-session-service
2208	0	0	/usr/lib/indicator-me/indicator-me-service
2210	0	0	/bin/sh -c /usr/bin/compiz-decorator
2211	0	0	/usr/bin/gtk-window-decorator
2215	0	0	/usr/lib/gvfs/gvfsd-burn --spawner :1.15 /org/gtk/gvfs/exec_spaw/1
2218	0	0	gnome-screensaver
2220	0	0	gnome-terminal
2223	0	0	gnome-pty-helper
2224	0	0	bash
2237	0	0	/usr/lib/gnome-disk-utility/gdu-notification-daemon
2240	0	0	/usr/bin/python /usr/share/system-config-printer/applet.py

B.3 Output for plugin linux_pslist

The following output was generated by the Volatility *linux_pslist* plugin, as found in Section 3.3.2:

Table B.2: Plugin output for linux_pslist (sorted by PID).

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf7070000	init	1	0	0	0x371ec000	2014-05-24 01:00:54 UTC+0000
0xf7070cb0	kthreadd	2	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7071960	ksoftirqd/0	3	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7072610	migration/0	4	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70732c0	watchdog/0	5	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7073f70	events/0	6	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7074c20	cpuset	7	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70758d0	khelper	8	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7076580	netns	9	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7077230	async/mgr	10	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7098000	pm	11	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7098cb0	sync_supers	12	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7099960	bdi-default	13	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709a610	kintegrityd/0	14	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709b2c0	kblockd/0	15	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709bf70	kacpid	16	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709cc20	kacpi_notify	17	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709d8d0	kacpi_hotplug	18	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709e580	ata_aux	19	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709f230	ata_sff/0	20	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70f8000	khubd	21	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70f8cb0	kseriod	22	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70f9960	kmmcd	23	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70fb2c0	khungtaskd	25	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70fbf70	kswapd0	26	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70fcc20	ksmd	27	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70fd8d0	aio/0	28	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70fe580	ecryptfs-kthrea	29	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70ff230	crypto/0	30	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf73fbf70	scsi_eh_0	36	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf73fd8d0	scsi_eh_1	38	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf73fa610	kstriped	40	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf73f8000	kmpathd/0	41	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf73f8cb0	kmpath_handlerd	42	0	0	----------	2014-05-24 01:00:54 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf73fe580	ksnapd	43	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf73ff230	kondemand/0	44	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf6878000	kconservative/0	45	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf69f0cb0	scsi_eh_2	180	0	0	----------	2014-05-24 01:00:55 UTC+0000
0xf69f32c0	usbhid_resumer	186	0	0	----------	2014-05-24 01:00:55 UTC+0000
0xf69f7230	jbd2/sda1-8	200	0	0	----------	2014-05-24 01:00:55 UTC+0000
0xf69f3f70	ext4-dio-unwrit	201	0	0	----------	2014-05-24 01:00:55 UTC+0000
0xf69b1960	upstart-udev-br	251	0	0	0x36a4f000	2014-05-24 01:00:55 UTC+0000
0xf69b0cb0	udevd	256	0	0	0x3696e000	2014-05-24 01:00:55 UTC+0000
0xf54f2610	udevd	373	0	0	0x354fc000	2014-05-24 01:00:55 UTC+0000
0xf55b4c20	udevd	406	0	0	0x355bd000	2014-05-24 01:00:55 UTC+0000
0xf54f58d0	kpsmoused	435	0	0	----------	2014-05-24 01:00:56 UTC+0000
0xf55f8000	iprt/0	494	0	0	----------	2014-05-24 01:00:56 UTC+0000
0xf5430000	rsyslogd	587	101	103	0x3738d000	2014-05-24 01:00:56 UTC+0000
0xf55f9960	dbus-daemon	620	102	105	0x36904000	2014-05-24 01:00:56 UTC+0000
0xf5433f70	NetworkManager	640	0	0	0x3554e000	2014-05-24 01:00:56 UTC+0000
0xf5470000	modem-manager	647	0	0	0x35593000	2014-05-24 01:00:56 UTC+0000
0xf5470cb0	avahi-daemon	648	104	109	0x36b1a000	2014-05-24 01:00:56 UTC+0000
0xf54f32c0	avahi-daemon	650	104	109	0x3544b000	2014-05-24 01:00:56 UTC+0000
0xf55b6580	dhclient	658	0	0	0x35416000	2014-05-24 01:00:56 UTC+0000
0xf55b1960	wpa_supplicant	663	0	0	0x35450000	2014-05-24 01:00:56 UTC+0000
0xf73fb2c0	getty	707	0	0	0x36a72000	2014-05-24 01:00:56 UTC+0000
0xf54f6580	getty	713	0	0	0x3558f000	2014-05-24 01:00:56 UTC+0000
0xf55fcc20	getty	724	0	0	0x36a71000	2014-05-24 01:00:56 UTC+0000
0xf55fa610	getty	726	0	0	0x35610000	2014-05-24 01:00:56 UTC+0000
0xf687cc20	getty	730	0	0	0x35514000	2014-05-24 01:00:56 UTC+0000
0xf69b58d0	acpid	733	0	0	0x354ff000	2014-05-24 01:00:56 UTC+0000
0xf54332c0	cron	734	0	0	0x35457000	2014-05-24 01:00:56 UTC+0000
0xf5431960	atd	736	0	0	0x3555a000	2014-05-24 01:00:56 UTC+0000
0xf54f7230	libvirtd	746	0	0	0x3544e000	2014-05-24 01:00:56 UTC+0000
0xf55fe580	VBoxService	844	0	0	0x3545d000	2014-05-24 01:00:56 UTC+0000
0xf70fa610	console-kit-dae	862	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5437230	dnsmasq	945	65534	30	0x35540000	2014-05-24 01:00:57 UTC+0000
0xf50b7230	flush-1:0	1018	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50b6580	flush-1:1	1019	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50b1960	flush-1:2	1020	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50b32c0	flush-1:3	1021	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50b4c20	flush-1:4	1022	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50b58d0	flush-1:5	1023	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf509a610	flush-1:6	1024	0	0	----------	2014-05-24 01:00:57 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf5099960	flush-1:7	1025	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf509cc20	flush-1:8	1026	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf509f230	flush-1:9	1027	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf509bf70	flush-1:10	1028	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf509d8d0	flush-1:11	1029	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf5430cb0	flush-1:12	1030	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d0000	flush-1:13	1031	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d0cb0	flush-1:14	1032	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d1960	flush-1:15	1033	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d2610	flush-7:0	1034	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d32c0	flush-7:1	1035	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d3f70	flush-7:2	1036	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d4c20	flush-7:3	1037	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d58d0	flush-7:4	1038	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d6580	flush-7:5	1039	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d7230	flush-7:6	1040	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50e8000	flush-7:7	1041	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50e8cb0	flush-8:0	1042	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50ecc20	login	1077	0	0	0x351a9000	2014-05-24 01:00:57 UTC+0000
0xf50b3f70	gdm-binary	1096	0	0	0x355d0000	2014-05-24 01:00:57 UTC+0000
0xf512cc20	cupsd	1106	0	0	0x35103000	2014-05-24 01:00:57 UTC+0000
0xf54732c0	bash	1819	0	0	0x35542000	2014-05-24 01:01:20 UTC+0000
0xf54f0000	startx	1842	0	0	0x36bc5000	2014-05-24 01:01:22 UTC+0000
0xf5129960	xinit	1859	0	0	0x35108000	2014-05-24 01:01:22 UTC+0000
0xf512bf70	Xorg	1860	0	0	0x35590000	2014-05-24 01:01:22 UTC+0000
0xf512a610	ck-launch-sessi	1863	0	0	0x3510b000	2014-05-24 01:01:23 UTC+0000
0xf5186580	VBoxClient	1906	0	0	0x36bff000	2014-05-24 01:01:23 UTC+0000
0xf50ea610	VBoxClient	1917	0	0	0x35588000	2014-05-24 01:01:23 UTC+0000
0xf50b2610	VBoxClient	1922	0	0	0x351a2000	2014-05-24 01:01:23 UTC+0000
0xf509e580	ssh-agent	1927	0	0	0x36952000	2014-05-24 01:01:23 UTC+0000
0xf69b0000	VBoxClient	1928	0	0	0x35143000	2014-05-24 01:01:23 UTC+0000
0xf6b37230	x-session-manag	1939	0	0	0x35100000	2014-05-24 01:01:23 UTC+0000
0xf6b36580	dbus-launch	1942	0	0	0x354c6000	2014-05-24 01:01:23 UTC+0000
0xf6b31960	dbus-daemon	1943	0	0	0x35163000	2014-05-24 01:01:23 UTC+0000
0xf6b358d0	gconfd-2	1948	0	0	0x36bd4000	2014-05-24 01:01:23 UTC+0000
0xf6b33f70	gnome-power-man	1949	0	0	0x36b84000	2014-05-24 01:01:23 UTC+0000
0xf687b2c0	gnome-keyring-d	1955	0	0	0x354cd000	2014-05-24 01:01:24 UTC+0000
0xf6bf58d0	gnome-settings-	1960	0	0	0x354c4000	2014-05-24 01:01:24 UTC+0000
0xf6bf0000	upowerd	1964	0	0	0x35494000	2014-05-24 01:01:24 UTC+0000
0xf5474c20	gvfsd	1972	0	0	0x35486000	2014-05-24 01:01:24 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf69f1960	gvfs-fuse-daemo	1982	0	0	0x35401000	2014-05-24 01:01:24 UTC+0000
0xf5182610	bluetooth-apple	1987	0	0	0x36bd7000	2014-05-24 01:01:24 UTC+0000
0xf5180cb0	compiz	1990	0	0	0x351af000	2014-05-24 01:01:24 UTC+0000
0xf50ed8d0	evolution-alarm	1993	0	0	0x36bc2000	2014-05-24 01:01:24 UTC+0000
0xf50eb2c0	nautilus	1996	0	0	0x36909000	2014-05-24 01:01:24 UTC+0000
0xf50b0cb0	gnome-panel	1999	0	0	0x35402000	2014-05-24 01:01:24 UTC+0000
0xf6b30cb0	nm-applet	2002	0	0	0x36991000	2014-05-24 01:01:24 UTC+0000
0xf6bf2610	polkit-gnome-au	2009	0	0	0x35188000	2014-05-24 01:01:24 UTC+0000
0xf5472610	gvfs-gdu-volume	2013	0	0	0x36b74000	2014-05-24 01:01:24 UTC+0000
0xf512f230	polkitd	2019	0	0	0x35196000	2014-05-24 01:01:24 UTC+0000
0xf73fcc20	udisks-daemon	2028	0	0	0x3519e000	2014-05-24 01:01:24 UTC+0000
0xf69b6580	udisks-daemon	2031	0	0	0x3519f000	2014-05-24 01:01:24 UTC+0000
0xf69f6580	gvfs-gphoto2-vo	2069	0	0	0x36bd9000	2014-05-24 01:01:24 UTC+0000
0xf69b7230	gvfs-afc-volume	2083	0	0	0x35169000	2014-05-24 01:01:24 UTC+0000
0xf6bf3f70	gvfsd-trash	2092	0	0	0x3516b000	2014-05-24 01:01:24 UTC+0000
0xf5128000	bonobo-activati	2145	0	0	0x3516d000	2014-05-24 01:01:24 UTC+0000
0xf54f1960	wnck-applet	2167	0	0	0x35162000	2014-05-24 01:01:25 UTC+0000
0xf69f4c20	trashapplet	2168	0	0	0x36902000	2014-05-24 01:01:25 UTC+0000
0xf6b958d0	indicator-apple	2186	0	0	0x36b69000	2014-05-24 01:01:25 UTC+0000
0xf6b96580	clock-applet	2187	0	0	0x36b5c000	2014-05-24 01:01:25 UTC+0000
0xf6b97230	indicator-apple	2188	0	0	0x351e4000	2014-05-24 01:01:25 UTC+0000
0xf51f8000	notification-ar	2189	0	0	0x351e5000	2014-05-24 01:01:25 UTC+0000
0xf51fd8d0	indicator-sound	2196	0	0	0x35207000	2014-05-24 01:01:25 UTC+0000
0xf51ff230	indicator-messa	2199	0	0	0x35238000	2014-05-24 01:01:25 UTC+0000
0xf6b94c20	indicator-appli	2200	0	0	0x3523d000	2014-05-24 01:01:25 UTC+0000
0xf51fcc20	indicator-sessi	2205	0	0	0x35261000	2014-05-24 01:01:26 UTC+0000
0xf5280000	indicator-me-se	2208	0	0	0x3526d000	2014-05-24 01:01:26 UTC+0000
0xf5281960	sh	2210	0	0	0x354ed000	2014-05-24 01:01:26 UTC+0000
0xf5282610	gtk-window-deco	2211	0	0	0x35174000	2014-05-24 01:01:26 UTC+0000
0xf5284c20	gvfsd-burn	2215	0	0	0x35279000	2014-05-24 01:01:26 UTC+0000
0xf5286580	gnome-screensav	2218	0	0	0x352a0000	2014-05-24 01:01:29 UTC+0000
0xf52858d0	gnome-terminal	2220	0	0	0x36b9d000	2014-05-24 01:01:30 UTC+0000
0xf6b90000	gnome-pty-helpe	2223	0	0	0x352bd000	2014-05-24 01:01:30 UTC+0000
0xf6b93f70	bash	2224	0	0	0x352bf000	2014-05-24 01:01:30 UTC+0000
0xf6bf1960	gdu-notificatio	2237	0	0	0x352ad000	2014-05-24 01:01:34 UTC+0000
0xf6878cb0	applet.py	2240	0	0	0x352dc000	2014-05-24 01:01:54 UTC+0000

B.4 Output for plugin linux_pstree

The following output was generated by the Volatility *linux_pslist* plugin, as found in Section 3.3.4:

Table B.3: Plugin output for linux_pstree.

Name	Pid	Uid
init	1	0
.upstart-udev-br	251	0
.udevd	256	0
..udevd	373	0
..udevd	406	0
.rsyslogd	587	101
.dbus-daemon	620	102
.NetworkManager	640	0
..dhclient	658	0
.avahi-daemon	648	104
..avahi-daemon	650	104
.modem-manager	647	0
.wpa_supplicant	663	0
.getty	707	0
.getty	713	0
.getty	724	0
.getty	726	0
.getty	730	0
.acpid	733	0
.cron	734	0
.atd	736	0
.libvirtd	746	0
.VBoxService	844	0
.console-kit-dae	862	0
.dnsmasq	945	65534
.login	1077	0
..bash	1819	0
...startx	1842	0
....xinit	1859	0
.....Xorg	1860	0
.....ck-launch-sessi	1863	0
......ssh-agent	1927	0
......x-session-manag	1939	0
.......gnome-power-man	1949	0

Name	Pid	Uid
.......bluetooth-apple	1987	0
.......compiz	1990	0
........sh	2210	0
.........gtk-window-deco	2211	0
.......evolution-alarm	1993	0
.......nautilus	1996	0
.......gnome-panel	1999	0
.......nm-applet	2002	0
.......polkit-gnome-au	2009	0
.......gdu-notificatio	2237	0
.......applet.py	2240	0
.gdm-binary	1096	0
.cupsd	1106	0
.VBoxClient	1906	0
.VBoxClient	1917	0
.VBoxClient	1922	0
.VBoxClient	1928	0
.dbus-daemon	1943	0
.dbus-launch	1942	0
.gconfd-2	1948	0
.gnome-keyring-d	1955	0
.upowerd	1964	0
.gvfsd	1972	0
.gnome-settings-	1960	0
.gvfs-fuse-daemo	1982	0
.polkitd	2019	0
.udisks-daemon	2028	0
..udisks-daemon	2031	0
.gvfs-gdu-volume	2013	0
.gvfs-gphoto2-vo	2069	0
.gvfs-afc-volume	2083	0
.gvfsd-trash	2092	0
.bonobo-activati	2145	0
.wnck-applet	2167	0
.trashapplet	2168	0
.indicator-apple	2186	0
.clock-applet	2187	0
.indicator-apple	2188	0
.notification-ar	2189	0
.indicator-sound	2196	0

Name	Pid	Uid
.indicator-messa	2199	0
.indicator-appli	2200	0
.indicator-sessi	2205	0
.indicator-me-se	2208	0
.gvfsd-burn	2215	0
.gnome-screensav	2218	0
.gnome-terminal	2220	0
..gnome-pty-helpe	2223	0
..bash	2224	0
[kthreadd]	2	0
.[ksoftirqd/0]	3	0
.[migration/0]	4	0
.[watchdog/0]	5	0
.[events/0]	6	0
.[cpuset]	7	0
.[khelper]	8	0
.[netns]	9	0
.[async/mgr]	10	0
.[pm]	11	0
.[sync_supers]	12	0
.[bdi-default]	13	0
.[kintegrityd/0]	14	0
.[kblockd/0]	15	0
.[kacpid]	16	0
.[kacpi_notify]	17	0
.[kacpi_hotplug]	18	0
.[ata_aux]	19	0
.[ata_sff/0]	20	0
.[khubd]	21	0
.[kseriod]	22	0
.[kmmcd]	23	0
.[khungtaskd]	25	0
.[kswapd0]	26	0
.[ksmd]	27	0
.[aio/0]	28	0
.[ecryptfs-kthrea]	29	0
.[crypto/0]	30	0
.[scsi_eh_0]	36	0
.[scsi_eh_1]	38	0
.[kstriped]	40	0

Name	Pid	Uid
.[kmpathd/0]	41	0
.[kmpath_handlerd]	42	0
.[ksnapd]	43	0
.[kondemand/0]	44	0
.[kconservative/0]	45	0
.[scsi_eh_2]	180	0
.[usbhid_resumer]	186	0
.[jbd2/sda1-8]	200	0
.[ext4-dio-unwrit]	201	0
.[kpsmoused]	435	0
.[iprt/0]	494	0
.[flush-1:0]	1018	0
.[flush-1:1]	1019	0
.[flush-1:2]	1020	0
.[flush-1:3]	1021	0
.[flush-1:4]	1022	0
.[flush-1:5]	1023	0
.[flush-1:6]	1024	0
.[flush-1:7]	1025	0
.[flush-1:8]	1026	0
.[flush-1:9]	1027	0
.[flush-1:10]	1028	0
.[flush-1:11]	1029	0
.[flush-1:12]	1030	0
.[flush-1:13]	1031	0
.[flush-1:14]	1032	0
.[flush-1:15]	1033	0
.[flush-7:0]	1034	0
.[flush-7:1]	1035	0
.[flush-7:2]	1036	0
.[flush-7:3]	1037	0
.[flush-7:4]	1038	0
.[flush-7:5]	1039	0
.[flush-7:6]	1040	0
.[flush-7:7]	1041	0
.[flush-8:0]	1042	0

B.5 Output for plugin linux_pidhashtable

The following output was generated by the Volatility *linux_pidhashtable* plugin, as found in Section 3.3.5:

Table B.4: Plugin output for linux_pidhashtable (sorted by PID).

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf7070000	init	1	0	0	0x371ec000	2014-05-24 01:00:54 UTC+0000
0xf7070cb0	kthreadd	2	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7071960	ksoftirqd/0	3	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7072610	migration/0	4	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70732c0	watchdog/0	5	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7073f70	events/0	6	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7074c20	cpuset	7	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70758d0	khelper	8	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7076580	netns	9	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7077230	async/mgr	10	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7098000	pm	11	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7098cb0	sync_supers	12	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf7099960	bdi-default	13	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709a610	kintegrityd/0	14	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709b2c0	kblockd/0	15	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709bf70	kacpid	16	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709cc20	kacpi_notify	17	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709d8d0	kacpi_hotplug	18	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709e580	ata_aux	19	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf709f230	ata_sff/0	20	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70f8000	khubd	21	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70f8cb0	kseriod	22	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70f9960	kmmcd	23	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70fb2c0	khungtaskd	25	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70fbf70	kswapd0	26	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70fcc20	ksmd	27	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70fd8d0	aio/0	28	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70fe580	ecryptfs-kthrea	29	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf70ff230	crypto/0	30	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf73fbf70	scsi_eh_0	36	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf73fd8d0	scsi_eh_1	38	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf73fa610	kstriped	40	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf73f8000	kmpathd/0	41	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf73f8cb0	kmpath_handlerd	42	0	0	----------	2014-05-24 01:00:54 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf73fe580	ksnapd	43	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf73ff230	kondemand/0	44	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf6878000	kconservative/0	45	0	0	----------	2014-05-24 01:00:54 UTC+0000
0xf69f0cb0	scsi_eh_2	180	0	0	----------	2014-05-24 01:00:55 UTC+0000
0xf69f32c0	usbhid_resumer	186	0	0	----------	2014-05-24 01:00:55 UTC+0000
0xf69f7230	jbd2/sda1-8	200	0	0	----------	2014-05-24 01:00:55 UTC+0000
0xf69f3f70	ext4-dio-unwrit	201	0	0	----------	2014-05-24 01:00:55 UTC+0000
0xf69b1960	upstart-udev-br	251	0	0	0x36a4f000	2014-05-24 01:00:55 UTC+0000
0xf69b0cb0	udevd	256	0	0	0x3696e000	2014-05-24 01:00:55 UTC+0000
0xf54f2610	udevd	373	0	0	0x354fc000	2014-05-24 01:00:55 UTC+0000
0xf55b4c20	udevd	406	0	0	0x355bd000	2014-05-24 01:00:55 UTC+0000
0xf54f58d0	kpsmoused	435	0	0	----------	2014-05-24 01:00:56 UTC+0000
0xf55f8000	iprt/0	494	0	0	----------	2014-05-24 01:00:56 UTC+0000
0xf5430000	rsyslogd	587	101	103	0x3738d000	2014-05-24 01:00:56 UTC+0000
0xf55f9960	dbus-daemon	620	102	105	0x36904000	2014-05-24 01:00:56 UTC+0000
0xf55fd8d0	rsyslogd	624	101	103	0x3738d000	2014-05-24 01:00:56 UTC+0000
0xf55fb2c0	rsyslogd	625	101	103	0x3738d000	2014-05-24 01:00:56 UTC+0000
0xf5433f70	NetworkManager	640	0	0	0x3554e000	2014-05-24 01:00:56 UTC+0000
0xf5470000	modem-manager	647	0	0	0x35593000	2014-05-24 01:00:56 UTC+0000
0xf5470cb0	avahi-daemon	648	104	109	0x36b1a000	2014-05-24 01:00:56 UTC+0000
0xf54f32c0	avahi-daemon	650	104	109	0x3544b000	2014-05-24 01:00:56 UTC+0000
0xf55b6580	dhclient	658	0	0	0x35416000	2014-05-24 01:00:56 UTC+0000
0xf55b1960	wpa_supplicant	663	0	0	0x35450000	2014-05-24 01:00:56 UTC+0000
0xf55b3f70	NetworkManager	664	0	0	0x3554e000	2014-05-24 01:00:56 UTC+0000
0xf73fb2c0	getty	707	0	0	0x36a72000	2014-05-24 01:00:56 UTC+0000
0xf54f6580	getty	713	0	0	0x3558f000	2014-05-24 01:00:56 UTC+0000
0xf55fcc20	getty	724	0	0	0x36a71000	2014-05-24 01:00:56 UTC+0000
0xf55fa610	getty	726	0	0	0x35610000	2014-05-24 01:00:56 UTC+0000
0xf687cc20	getty	730	0	0	0x35514000	2014-05-24 01:00:56 UTC+0000
0xf69b58d0	acpid	733	0	0	0x354ff000	2014-05-24 01:00:56 UTC+0000
0xf54332c0	cron	734	0	0	0x35457000	2014-05-24 01:00:56 UTC+0000
0xf5431960	atd	736	0	0	0x3555a000	2014-05-24 01:00:56 UTC+0000
0xf54f7230	libvirtd	746	0	0	0x3544e000	2014-05-24 01:00:56 UTC+0000
0xf6b34c20	libvirtd	749	0	0	0x3544e000	2014-05-24 01:00:56 UTC+0000
0xf55b32c0	libvirtd	752	0	0	0x3544e000	2014-05-24 01:00:56 UTC+0000
0xf55b0cb0	libvirtd	753	0	0	0x3544e000	2014-05-24 01:00:56 UTC+0000
0xf55b58d0	libvirtd	754	0	0	0x3544e000	2014-05-24 01:00:56 UTC+0000
0xf55b0000	libvirtd	755	0	0	0x3544e000	2014-05-24 01:00:56 UTC+0000
0xf55b2610	libvirtd	756	0	0	0x3544e000	2014-05-24 01:00:56 UTC+0000
0xf55fe580	VBoxService	844	0	0	0x3545d000	2014-05-24 01:00:56 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf6bf6580	control	847	0	0	0x3545d000	2014-05-24 01:00:56 UTC+0000
0xf5434c20	timesync	849	0	0	0x3545d000	2014-05-24 01:00:56 UTC+0000
0xf5432610	vminfo	852	0	0	0x3545d000	2014-05-24 01:00:56 UTC+0000
0xf55b7230	cpuhotplug	855	0	0	0x3545d000	2014-05-24 01:00:56 UTC+0000
0xf6b332c0	memballoon	858	0	0	0x3545d000	2014-05-24 01:00:56 UTC+0000
0xf54f3f70	vmstats	861	0	0	0x3545d000	2014-05-24 01:00:56 UTC+0000
0xf70fa610	console-kit-dae	862	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf55ff230	automount	864	0	0	0x3545d000	2014-05-24 01:00:56 UTC+0000
0xf54358d0	console-kit-dae	868	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5436580	console-kit-dae	869	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf6bf7230	console-kit-dae	870	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf69b2610	console-kit-dae	871	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf687bf70	console-kit-dae	872	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf687d8d0	console-kit-dae	873	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf54f4c20	console-kit-dae	875	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf73f9960	console-kit-dae	876	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5471960	console-kit-dae	877	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf55fbf70	console-kit-dae	878	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5008000	console-kit-dae	879	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5008cb0	console-kit-dae	880	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5009960	console-kit-dae	881	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf500a610	console-kit-dae	882	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf500b2c0	console-kit-dae	883	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf500bf70	console-kit-dae	884	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf500cc20	console-kit-dae	885	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf500d8d0	console-kit-dae	886	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf500e580	console-kit-dae	887	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf500f230	console-kit-dae	888	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5028000	console-kit-dae	889	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5028cb0	console-kit-dae	890	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5029960	console-kit-dae	891	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf502a610	console-kit-dae	892	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf502b2c0	console-kit-dae	893	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf502bf70	console-kit-dae	894	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf502cc20	console-kit-dae	895	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf502d8d0	console-kit-dae	896	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf502e580	console-kit-dae	897	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf502f230	console-kit-dae	898	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5040000	console-kit-dae	899	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5040cb0	console-kit-dae	900	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf5041960	console-kit-dae	901	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5042610	console-kit-dae	902	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf50432c0	console-kit-dae	903	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5043f70	console-kit-dae	904	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5044c20	console-kit-dae	905	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf50458d0	console-kit-dae	906	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5046580	console-kit-dae	907	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5047230	console-kit-dae	908	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5060000	console-kit-dae	909	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5060cb0	console-kit-dae	910	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5061960	console-kit-dae	911	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5062610	console-kit-dae	912	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf50632c0	console-kit-dae	913	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5063f70	console-kit-dae	914	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5064c20	console-kit-dae	915	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf50658d0	console-kit-dae	916	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5066580	console-kit-dae	917	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5067230	console-kit-dae	918	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5078000	console-kit-dae	919	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5078cb0	console-kit-dae	920	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5079960	console-kit-dae	921	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf507a610	console-kit-dae	922	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf507b2c0	console-kit-dae	923	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf507bf70	console-kit-dae	924	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf507cc20	console-kit-dae	925	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf507d8d0	console-kit-dae	926	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf507e580	console-kit-dae	927	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf507f230	console-kit-dae	928	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5098000	console-kit-dae	929	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf509b2c0	console-kit-dae	933	0	0	0x35512000	2014-05-24 01:00:56 UTC+0000
0xf5437230	dnsmasq	945	65534	30	0x35540000	2014-05-24 01:00:57 UTC+0000
0xf50b7230	flush-1:0	1018	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50b6580	flush-1:1	1019	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50b1960	flush-1:2	1020	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50b32c0	flush-1:3	1021	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50b4c20	flush-1:4	1022	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50b58d0	flush-1:5	1023	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf509a610	flush-1:6	1024	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf5099960	flush-1:7	1025	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf509cc20	flush-1:8	1026	0	0	----------	2014-05-24 01:00:57 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf509f230	flush-1:9	1027	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf509bf70	flush-1:10	1028	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf509d8d0	flush-1:11	1029	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf5430cb0	flush-1:12	1030	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d0000	flush-1:13	1031	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d0cb0	flush-1:14	1032	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d1960	flush-1:15	1033	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d2610	flush-7:0	1034	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d32c0	flush-7:1	1035	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d3f70	flush-7:2	1036	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d4c20	flush-7:3	1037	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d58d0	flush-7:4	1038	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d6580	flush-7:5	1039	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50d7230	flush-7:6	1040	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50e8000	flush-7:7	1041	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50e8cb0	flush-8:0	1042	0	0	----------	2014-05-24 01:00:57 UTC+0000
0xf50ecc20	login	1077	0	0	0x351a9000	2014-05-24 01:00:57 UTC+0000
0xf50b3f70	gdm-binary	1096	0	0	0x355d0000	2014-05-24 01:00:57 UTC+0000
0xf512cc20	cupsd	1106	0	0	0x35103000	2014-05-24 01:00:57 UTC+0000
0xf5128cb0	gdm-binary	1107	0	0	0x355d0000	2014-05-24 01:00:57 UTC+0000
0xf54732c0	bash	1819	0	0	0x35542000	2014-05-24 01:01:20 UTC+0000
0xf54f0000	startx	1842	0	0	0x36bc5000	2014-05-24 01:01:22 UTC+0000
0xf5129960	xinit	1859	0	0	0x35108000	2014-05-24 01:01:22 UTC+0000
0xf512bf70	Xorg	1860	0	0	0x35590000	2014-05-24 01:01:22 UTC+0000
0xf512d8d0	console-kit-dae	1861	0	0	0x35512000	2014-05-24 01:01:22 UTC+0000
0xf512a610	ck-launch-sessi	1863	0	0	0x3510b000	2014-05-24 01:01:23 UTC+0000
0xf5186580	VBoxClient	1906	0	0	0x36bff000	2014-05-24 01:01:23 UTC+0000
0xf5187230	SHCLIP	1907	0	0	0x36bff000	2014-05-24 01:01:23 UTC+0000
0xf50ea610	VBoxClient	1917	0	0	0x35588000	2014-05-24 01:01:23 UTC+0000
0xf50e9960	X11 monitor	1919	0	0	0x35588000	2014-05-24 01:01:23 UTC+0000
0xf50b2610	VBoxClient	1922	0	0	0x351a2000	2014-05-24 01:01:23 UTC+0000
0xf50b0000	Host events	1924	0	0	0x351a2000	2014-05-24 01:01:23 UTC+0000
0xf509e580	ssh-agent	1927	0	0	0x36952000	2014-05-24 01:01:23 UTC+0000
0xf69b0000	VBoxClient	1928	0	0	0x35143000	2014-05-24 01:01:23 UTC+0000
0xf69b4c20	HGCM-NOTIFY	1929	0	0	0x35143000	2014-05-24 01:01:23 UTC+0000
0xf69b3f70	X11-NOTIFY	1930	0	0	0x35143000	2014-05-24 01:01:23 UTC+0000
0xf6b37230	x-session-manag	1939	0	0	0x35100000	2014-05-24 01:01:23 UTC+0000
0xf6b36580	dbus-launch	1942	0	0	0x354c6000	2014-05-24 01:01:23 UTC+0000
0xf6b31960	dbus-daemon	1943	0	0	0x35163000	2014-05-24 01:01:23 UTC+0000
0xf6b30000	x-session-manag	1946	0	0	0x35100000	2014-05-24 01:01:23 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf6b358d0	gconfd-2	1948	0	0	0x36bd4000	2014-05-24 01:01:23 UTC+0000
0xf6b33f70	gnome-power-man	1949	0	0	0x36b84000	2014-05-24 01:01:23 UTC+0000
0xf6b32610	x-session-manag	1950	0	0	0x35100000	2014-05-24 01:01:23 UTC+0000
0xf687b2c0	gnome-keyring-d	1955	0	0	0x354cd000	2014-05-24 01:01:24 UTC+0000
0xf687f230	gnome-keyring-d	1957	0	0	0x354cd000	2014-05-24 01:01:24 UTC+0000
0xf6bf4c20	gnome-keyring-d	1958	0	0	0x354cd000	2014-05-24 01:01:24 UTC+0000
0xf6bf58d0	gnome-settings-	1960	0	0	0x354c4000	2014-05-24 01:01:24 UTC+0000
0xf6bf32c0	gnome-power-man	1961	0	0	0x36b84000	2014-05-24 01:01:24 UTC+0000
0xf6bf0cb0	gnome-settings-	1963	0	0	0x354c4000	2014-05-24 01:01:24 UTC+0000
0xf6bf0000	upowerd	1964	0	0	0x35494000	2014-05-24 01:01:24 UTC+0000
0xf5474c20	gvfsd	1972	0	0	0x35486000	2014-05-24 01:01:24 UTC+0000
0xf69f1960	gvfs-fuse-daemo	1982	0	0	0x35401000	2014-05-24 01:01:24 UTC+0000
0xf69f58d0	gvfs-fuse-daemo	1985	0	0	0x35401000	2014-05-24 01:01:24 UTC+0000
0xf5182610	bluetooth-apple	1987	0	0	0x36bd7000	2014-05-24 01:01:24 UTC+0000
0xf5181960	gvfs-fuse-daemo	1988	0	0	0x35401000	2014-05-24 01:01:24 UTC+0000
0xf5180cb0	compiz	1990	0	0	0x351af000	2014-05-24 01:01:24 UTC+0000
0xf5180000	gvfs-fuse-daemo	1991	0	0	0x35401000	2014-05-24 01:01:24 UTC+0000
0xf50ed8d0	evolution-alarm	1993	0	0	0x36bc2000	2014-05-24 01:01:24 UTC+0000
0xf50eb2c0	nautilus	1996	0	0	0x36909000	2014-05-24 01:01:24 UTC+0000
0xf50b0cb0	gnome-panel	1999	0	0	0x35402000	2014-05-24 01:01:24 UTC+0000
0xf6b30cb0	nm-applet	2002	0	0	0x36991000	2014-05-24 01:01:24 UTC+0000
0xf6bf2610	polkit-gnome-au	2009	0	0	0x35188000	2014-05-24 01:01:24 UTC+0000
0xf5472610	gvfs-gdu-volume	2013	0	0	0x36b74000	2014-05-24 01:01:24 UTC+0000
0xf512f230	polkitd	2019	0	0	0x35196000	2014-05-24 01:01:24 UTC+0000
0xf51832c0	bluetooth-apple	2025	0	0	0x36bd7000	2014-05-24 01:01:24 UTC+0000
0xf73fcc20	udisks-daemon	2028	0	0	0x3519e000	2014-05-24 01:01:24 UTC+0000
0xf69b6580	udisks-daemon	2031	0	0	0x3519f000	2014-05-24 01:01:24 UTC+0000
0xf687a610	nautilus	2035	0	0	0x36909000	2014-05-24 01:01:24 UTC+0000
0xf5473f70	gnome-panel	2038	0	0	0x35402000	2014-05-24 01:01:24 UTC+0000
0xf5476580	evolution-alarm	2040	0	0	0x36bc2000	2014-05-24 01:01:24 UTC+0000
0xf69f2610	nm-applet	2049	0	0	0x36991000	2014-05-24 01:01:24 UTC+0000
0xf69f6580	gvfs-gphoto2-vo	2069	0	0	0x36bd9000	2014-05-24 01:01:24 UTC+0000
0xf69b7230	gvfs-afc-volume	2083	0	0	0x35169000	2014-05-24 01:01:24 UTC+0000
0xf54758d0	gvfs-afc-volume	2089	0	0	0x35169000	2014-05-24 01:01:24 UTC+0000
0xf6bf3f70	gvfsd-trash	2092	0	0	0x3516b000	2014-05-24 01:01:24 UTC+0000
0xf5128000	bonobo-activati	2145	0	0	0x3516d000	2014-05-24 01:01:24 UTC+0000
0xf5098cb0	bonobo-activati	2147	0	0	0x3516d000	2014-05-24 01:01:24 UTC+0000
0xf687e580	bonobo-activati	2161	0	0	0x3516d000	2014-05-24 01:01:24 UTC+0000
0xf54f1960	wnck-applet	2167	0	0	0x35162000	2014-05-24 01:01:25 UTC+0000
0xf69f4c20	trashapplet	2168	0	0	0x36902000	2014-05-24 01:01:25 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf5477230	compiz	2173	0	0	0x351af000	2014-05-24 01:01:25 UTC+0000
0xf6b90cb0	wnck-applet	2180	0	0	0x35162000	2014-05-24 01:01:25 UTC+0000
0xf6b91960	trashapplet	2181	0	0	0x36902000	2014-05-24 01:01:25 UTC+0000
0xf6b958d0	indicator-apple	2186	0	0	0x36b69000	2014-05-24 01:01:25 UTC+0000
0xf6b96580	clock-applet	2187	0	0	0x36b5c000	2014-05-24 01:01:25 UTC+0000
0xf6b97230	indicator-apple	2188	0	0	0x351e4000	2014-05-24 01:01:25 UTC+0000
0xf51f8000	notification-ar	2189	0	0	0x351e5000	2014-05-24 01:01:25 UTC+0000
0xf51f8cb0	notification-ar	2190	0	0	0x351e5000	2014-05-24 01:01:25 UTC+0000
0xf51f9960	indicator-apple	2191	0	0	0x36b69000	2014-05-24 01:01:25 UTC+0000
0xf51fa610	indicator-apple	2192	0	0	0x351e4000	2014-05-24 01:01:25 UTC+0000
0xf51fb2c0	clock-applet	2193	0	0	0x36b5c000	2014-05-24 01:01:25 UTC+0000
0xf51fd8d0	indicator-sound	2196	0	0	0x35207000	2014-05-24 01:01:25 UTC+0000
0xf51ff230	indicator-messa	2199	0	0	0x35238000	2014-05-24 01:01:25 UTC+0000
0xf6b94c20	indicator-appli	2200	0	0	0x3523d000	2014-05-24 01:01:25 UTC+0000
0xf51fcc20	indicator-sessi	2205	0	0	0x35261000	2014-05-24 01:01:26 UTC+0000
0xf51fe580	indicator-sessi	2206	0	0	0x35261000	2014-05-24 01:01:26 UTC+0000
0xf5280000	indicator-me-se	2208	0	0	0x3526d000	2014-05-24 01:01:26 UTC+0000
0xf5280cb0	indicator-me-se	2209	0	0	0x3526d000	2014-05-24 01:01:26 UTC+0000
0xf5281960	sh	2210	0	0	0x354ed000	2014-05-24 01:01:26 UTC+0000
0xf5282610	gtk-window-deco	2211	0	0	0x35174000	2014-05-24 01:01:26 UTC+0000
0xf52832c0	gtk-window-deco	2213	0	0	0x35174000	2014-05-24 01:01:26 UTC+0000
0xf5284c20	gvfsd-burn	2215	0	0	0x35279000	2014-05-24 01:01:26 UTC+0000
0xf5286580	gnome-screensav	2218	0	0	0x352a0000	2014-05-24 01:01:29 UTC+0000
0xf52858d0	gnome-terminal	2220	0	0	0x36b9d000	2014-05-24 01:01:30 UTC+0000
0xf5287230	gnome-terminal	2222	0	0	0x36b9d000	2014-05-24 01:01:30 UTC+0000
0xf6b90000	gnome-pty-helpe	2223	0	0	0x352bd000	2014-05-24 01:01:30 UTC+0000
0xf6b93f70	bash	2224	0	0	0x352bf000	2014-05-24 01:01:30 UTC+0000
0xf6879960	gnome-terminal	2225	0	0	0x36b9d000	2014-05-24 01:01:30 UTC+0000
0xf6bf1960	gdu-notificatio	2237	0	0	0x352ad000	2014-05-24 01:01:34 UTC+0000
0xf6878cb0	applet.py	2240	0	0	0x352dc000	2014-05-24 01:01:54 UTC+0000

B.6 Output for plugin linux_psxview

The following output was generated by the Volatility *linux_psxview* plugin, as found in Section 3.3.6:

Table B.5: Plugin output for linux_psxview (sorted by PID).

Offset(V)	Name	PID	Pslist	Pid_hash	Kmem_cache	Parents	Leaders
0xc07c76e0	swapper	0	FALSE	FALSE	FALSE	TRUE	FALSE
0xf7070000	init	1	TRUE	TRUE	FALSE	TRUE	TRUE
0xf7070cb0	kthreadd	2	TRUE	TRUE	FALSE	TRUE	TRUE
0xf7071960	ksoftirqd/0	3	TRUE	TRUE	FALSE	FALSE	TRUE
0xf7072610	migration/0	4	TRUE	TRUE	FALSE	FALSE	TRUE
0xf70732c0	watchdog/0	5	TRUE	TRUE	FALSE	FALSE	TRUE
0xf7073f70	events/0	6	TRUE	TRUE	FALSE	FALSE	TRUE
0xf7074c20	cpuset	7	TRUE	TRUE	FALSE	FALSE	TRUE
0xf70758d0	khelper	8	TRUE	TRUE	FALSE	FALSE	TRUE
0xf7076580	netns	9	TRUE	TRUE	FALSE	FALSE	TRUE
0xf7077230	async/mgr	10	TRUE	TRUE	FALSE	FALSE	TRUE
0xf7098000	pm	11	TRUE	TRUE	FALSE	FALSE	TRUE
0xf7098cb0	sync_supers	12	TRUE	TRUE	FALSE	FALSE	TRUE
0xf7099960	bdi-default	13	TRUE	TRUE	FALSE	FALSE	TRUE
0xf709a610	kintegrityd/0	14	TRUE	TRUE	FALSE	FALSE	TRUE
0xf709b2c0	kblockd/0	15	TRUE	TRUE	FALSE	FALSE	TRUE
0xf709bf70	kacpid	16	TRUE	TRUE	FALSE	FALSE	TRUE
0xf709cc20	kacpi_notify	17	TRUE	TRUE	FALSE	FALSE	TRUE
0xf709d8d0	kacpi_hotplug	18	TRUE	TRUE	FALSE	FALSE	TRUE
0xf709e580	ata_aux	19	TRUE	TRUE	FALSE	FALSE	TRUE
0xf709f230	ata_sff/0	20	TRUE	TRUE	FALSE	FALSE	TRUE
0xf70f8000	khubd	21	TRUE	TRUE	FALSE	FALSE	TRUE
0xf70f8cb0	kseriod	22	TRUE	TRUE	FALSE	FALSE	TRUE
0xf70f9960	kmmcd	23	TRUE	TRUE	FALSE	FALSE	TRUE
0xf70fb2c0	khungtaskd	25	TRUE	TRUE	FALSE	FALSE	TRUE
0xf70fbf70	kswapd0	26	TRUE	TRUE	FALSE	FALSE	TRUE
0xf70fcc20	ksmd	27	TRUE	TRUE	FALSE	FALSE	TRUE
0xf70fd8d0	aio/0	28	TRUE	TRUE	FALSE	FALSE	TRUE
0xf70fe580	ecryptfs-kthrea	29	TRUE	TRUE	FALSE	FALSE	TRUE
0xf70ff230	crypto/0	30	TRUE	TRUE	FALSE	FALSE	TRUE
0xf73fbf70	scsi_eh_0	36	TRUE	TRUE	FALSE	FALSE	TRUE
0xf73fd8d0	scsi_eh_1	38	TRUE	TRUE	FALSE	FALSE	TRUE
0xf73fa610	kstriped	40	TRUE	TRUE	FALSE	FALSE	TRUE
0xf73f8000	kmpathd/0	41	TRUE	TRUE	FALSE	FALSE	TRUE

Offset(V)	Name	PID	Pslist	Pid_hash	Kmem_cache	Parents	Leaders
0xf73f8cb0	kmpath_handlerd	42	TRUE	TRUE	FALSE	FALSE	TRUE
0xf73fe580	ksnapd	43	TRUE	TRUE	FALSE	FALSE	TRUE
0xf73ff230	kondemand/0	44	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6878000	kconservative/0	45	TRUE	TRUE	FALSE	FALSE	TRUE
0xf69f0cb0	scsi_eh_2	180	TRUE	TRUE	FALSE	FALSE	TRUE
0xf69f32c0	usbhid_resumer	186	TRUE	TRUE	FALSE	FALSE	TRUE
0xf69f7230	jbd2/sda1-8	200	TRUE	TRUE	FALSE	FALSE	TRUE
0xf69f3f70	ext4-dio-unwrit	201	TRUE	TRUE	FALSE	FALSE	TRUE
0xf69b1960	upstart-udev-br	251	TRUE	TRUE	FALSE	FALSE	TRUE
0xf69b0cb0	udevd	256	TRUE	TRUE	FALSE	TRUE	TRUE
0xf54f2610	udevd	373	TRUE	TRUE	FALSE	FALSE	TRUE
0xf55b4c20	udevd	406	TRUE	TRUE	FALSE	FALSE	TRUE
0xf54f58d0	kpsmoused	435	TRUE	TRUE	FALSE	FALSE	TRUE
0xf55f8000	iprt/0	494	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5430000	rsyslogd	587	TRUE	TRUE	FALSE	FALSE	TRUE
0xf55f9960	dbus-daemon	620	TRUE	TRUE	FALSE	FALSE	TRUE
0xf55fd8d0	rsyslogd	624	FALSE	TRUE	FALSE	FALSE	FALSE
0xf55fb2c0	rsyslogd	625	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5433f70	NetworkManager	640	TRUE	TRUE	FALSE	TRUE	TRUE
0xf5470000	modem-manager	647	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5470cb0	avahi-daemon	648	TRUE	TRUE	FALSE	TRUE	TRUE
0xf54f32c0	avahi-daemon	650	TRUE	TRUE	FALSE	FALSE	TRUE
0xf55b6580	dhclient	658	TRUE	TRUE	FALSE	FALSE	TRUE
0xf55b1960	wpa_supplicant	663	TRUE	TRUE	FALSE	FALSE	TRUE
0xf55b3f70	NetworkManager	664	FALSE	TRUE	FALSE	FALSE	FALSE
0xf73fb2c0	getty	707	TRUE	TRUE	FALSE	FALSE	TRUE
0xf54f6580	getty	713	TRUE	TRUE	FALSE	FALSE	TRUE
0xf55fcc20	getty	724	TRUE	TRUE	FALSE	FALSE	TRUE
0xf55fa610	getty	726	TRUE	TRUE	FALSE	FALSE	TRUE
0xf687cc20	getty	730	TRUE	TRUE	FALSE	FALSE	TRUE
0xf69b58d0	acpid	733	TRUE	TRUE	FALSE	FALSE	TRUE
0xf54332c0	cron	734	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5431960	atd	736	TRUE	TRUE	FALSE	FALSE	TRUE
0xf54f7230	libvirtd	746	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6b34c20	libvirtd	749	FALSE	TRUE	FALSE	FALSE	FALSE
0xf55b32c0	libvirtd	752	FALSE	TRUE	FALSE	FALSE	FALSE
0xf55b0cb0	libvirtd	753	FALSE	TRUE	FALSE	FALSE	FALSE
0xf55b58d0	libvirtd	754	FALSE	TRUE	FALSE	FALSE	FALSE
0xf55b0000	libvirtd	755	FALSE	TRUE	FALSE	FALSE	FALSE
0xf55b2610	libvirtd	756	FALSE	TRUE	FALSE	FALSE	FALSE

Offset(V)	Name	PID	Pslist	Pid_hash	Kmem_cache	Parents	Leaders
0xf55fe580	VBoxService	844	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6bf6580	control	847	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5434c20	timesync	849	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5432610	vminfo	852	FALSE	TRUE	FALSE	FALSE	FALSE
0xf55b7230	cpuhotplug	855	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6b332c0	memballoon	858	FALSE	TRUE	FALSE	FALSE	FALSE
0xf54f3f70	vmstats	861	FALSE	TRUE	FALSE	FALSE	FALSE
0xf70fa610	console-kit-dae	862	TRUE	TRUE	FALSE	FALSE	TRUE
0xf55ff230	automount	864	FALSE	TRUE	FALSE	FALSE	FALSE
0xf54358d0	console-kit-dae	868	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5436580	console-kit-dae	869	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6bf7230	console-kit-dae	870	FALSE	TRUE	FALSE	FALSE	FALSE
0xf69b2610	console-kit-dae	871	FALSE	TRUE	FALSE	FALSE	FALSE
0xf687bf70	console-kit-dae	872	FALSE	TRUE	FALSE	FALSE	FALSE
0xf687d8d0	console-kit-dae	873	FALSE	TRUE	FALSE	FALSE	FALSE
0xf54f4c20	console-kit-dae	875	FALSE	TRUE	FALSE	FALSE	FALSE
0xf73f9960	console-kit-dae	876	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5471960	console-kit-dae	877	FALSE	TRUE	FALSE	FALSE	FALSE
0xf55fbf70	console-kit-dae	878	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5008000	console-kit-dae	879	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5008cb0	console-kit-dae	880	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5009960	console-kit-dae	881	FALSE	TRUE	FALSE	FALSE	FALSE
0xf500a610	console-kit-dae	882	FALSE	TRUE	FALSE	FALSE	FALSE
0xf500b2c0	console-kit-dae	883	FALSE	TRUE	FALSE	FALSE	FALSE
0xf500bf70	console-kit-dae	884	FALSE	TRUE	FALSE	FALSE	FALSE
0xf500cc20	console-kit-dae	885	FALSE	TRUE	FALSE	FALSE	FALSE
0xf500d8d0	console-kit-dae	886	FALSE	TRUE	FALSE	FALSE	FALSE
0xf500e580	console-kit-dae	887	FALSE	TRUE	FALSE	FALSE	FALSE
0xf500f230	console-kit-dae	888	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5028000	console-kit-dae	889	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5028cb0	console-kit-dae	890	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5029960	console-kit-dae	891	FALSE	TRUE	FALSE	FALSE	FALSE
0xf502a610	console-kit-dae	892	FALSE	TRUE	FALSE	FALSE	FALSE
0xf502b2c0	console-kit-dae	893	FALSE	TRUE	FALSE	FALSE	FALSE
0xf502bf70	console-kit-dae	894	FALSE	TRUE	FALSE	FALSE	FALSE
0xf502cc20	console-kit-dae	895	FALSE	TRUE	FALSE	FALSE	FALSE
0xf502d8d0	console-kit-dae	896	FALSE	TRUE	FALSE	FALSE	FALSE
0xf502e580	console-kit-dae	897	FALSE	TRUE	FALSE	FALSE	FALSE
0xf502f230	console-kit-dae	898	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5040000	console-kit-dae	899	FALSE	TRUE	FALSE	FALSE	FALSE

Offset(V)	Name	PID	Pslist	Pid_hash	Kmem_cache	Parents	Leaders
0xf5040cb0	console-kit-dae	900	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5041960	console-kit-dae	901	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5042610	console-kit-dae	902	FALSE	TRUE	FALSE	FALSE	FALSE
0xf50432c0	console-kit-dae	903	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5043f70	console-kit-dae	904	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5044c20	console-kit-dae	905	FALSE	TRUE	FALSE	FALSE	FALSE
0xf50458d0	console-kit-dae	906	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5046580	console-kit-dae	907	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5047230	console-kit-dae	908	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5060000	console-kit-dae	909	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5060cb0	console-kit-dae	910	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5061960	console-kit-dae	911	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5062610	console-kit-dae	912	FALSE	TRUE	FALSE	FALSE	FALSE
0xf50632c0	console-kit-dae	913	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5063f70	console-kit-dae	914	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5064c20	console-kit-dae	915	FALSE	TRUE	FALSE	FALSE	FALSE
0xf50658d0	console-kit-dae	916	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5066580	console-kit-dae	917	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5067230	console-kit-dae	918	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5078000	console-kit-dae	919	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5078cb0	console-kit-dae	920	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5079960	console-kit-dae	921	FALSE	TRUE	FALSE	FALSE	FALSE
0xf507a610	console-kit-dae	922	FALSE	TRUE	FALSE	FALSE	FALSE
0xf507b2c0	console-kit-dae	923	FALSE	TRUE	FALSE	FALSE	FALSE
0xf507bf70	console-kit-dae	924	FALSE	TRUE	FALSE	FALSE	FALSE
0xf507cc20	console-kit-dae	925	FALSE	TRUE	FALSE	FALSE	FALSE
0xf507d8d0	console-kit-dae	926	FALSE	TRUE	FALSE	FALSE	FALSE
0xf507e580	console-kit-dae	927	FALSE	TRUE	FALSE	FALSE	FALSE
0xf507f230	console-kit-dae	928	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5098000	console-kit-dae	929	FALSE	TRUE	FALSE	FALSE	FALSE
0xf509b2c0	console-kit-dae	933	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5437230	dnsmasq	945	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50b7230	flush-1:0	1018	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50b6580	flush-1:1	1019	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50b1960	flush-1:2	1020	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50b32c0	flush-1:3	1021	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50b4c20	flush-1:4	1022	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50b58d0	flush-1:5	1023	TRUE	TRUE	FALSE	FALSE	TRUE
0xf509a610	flush-1:6	1024	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5099960	flush-1:7	1025	TRUE	TRUE	FALSE	FALSE	TRUE

Offset(V)	Name	PID	Pslist	Pid_hash	Kmem_cache	Parents	Leaders
0xf509cc20	flush-1:8	1026	TRUE	TRUE	FALSE	FALSE	TRUE
0xf509f230	flush-1:9	1027	TRUE	TRUE	FALSE	FALSE	TRUE
0xf509bf70	flush-1:10	1028	TRUE	TRUE	FALSE	FALSE	TRUE
0xf509d8d0	flush-1:11	1029	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5430cb0	flush-1:12	1030	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50d0000	flush-1:13	1031	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50d0cb0	flush-1:14	1032	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50d1960	flush-1:15	1033	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50d2610	flush-7:0	1034	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50d32c0	flush-7:1	1035	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50d3f70	flush-7:2	1036	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50d4c20	flush-7:3	1037	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50d58d0	flush-7:4	1038	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50d6580	flush-7:5	1039	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50d7230	flush-7:6	1040	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50e8000	flush-7:7	1041	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50e8cb0	flush-8:0	1042	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50ecc20	login	1077	TRUE	TRUE	FALSE	TRUE	TRUE
0xf50b3f70	gdm-binary	1096	TRUE	TRUE	FALSE	FALSE	TRUE
0xf512cc20	cupsd	1106	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5128cb0	gdm-binary	1107	FALSE	TRUE	FALSE	FALSE	FALSE
0xf54732c0	bash	1819	TRUE	TRUE	FALSE	TRUE	TRUE
0xf54f0000	startx	1842	TRUE	TRUE	FALSE	TRUE	TRUE
0xf5129960	xinit	1859	TRUE	TRUE	FALSE	TRUE	TRUE
0xf512bf70	Xorg	1860	TRUE	TRUE	FALSE	FALSE	TRUE
0xf512d8d0	console-kit-dae	1861	FALSE	TRUE	FALSE	FALSE	FALSE
0xf512a610	ck-launch-sessi	1863	TRUE	TRUE	FALSE	TRUE	TRUE
0xf5186580	VBoxClient	1906	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5187230	SHCLIP	1907	FALSE	TRUE	FALSE	FALSE	FALSE
0xf50ea610	VBoxClient	1917	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50e9960	X11 monitor	1919	FALSE	TRUE	FALSE	FALSE	FALSE
0xf50b2610	VBoxClient	1922	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50b0000	Host events	1924	FALSE	TRUE	FALSE	FALSE	FALSE
0xf509e580	ssh-agent	1927	TRUE	TRUE	FALSE	FALSE	TRUE
0xf69b0000	VBoxClient	1928	TRUE	TRUE	FALSE	FALSE	TRUE
0xf69b4c20	HGCM-NOTIFY	1929	FALSE	TRUE	FALSE	FALSE	FALSE
0xf69b3f70	X11-NOTIFY	1930	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6b37230	x-session-manag	1939	TRUE	TRUE	FALSE	TRUE	TRUE
0xf6b36580	dbus-launch	1942	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6b31960	dbus-daemon	1943	TRUE	TRUE	FALSE	FALSE	TRUE

Offset(V)	Name	PID	Pslist	Pid_hash	Kmem_cache	Parents	Leaders
0xf6b30000	x-session-manag	1946	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6b358d0	gconfd-2	1948	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6b33f70	gnome-power-man	1949	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6b32610	x-session-manag	1950	FALSE	TRUE	FALSE	FALSE	FALSE
0xf687b2c0	gnome-keyring-d	1955	TRUE	TRUE	FALSE	FALSE	TRUE
0xf687f230	gnome-keyring-d	1957	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6bf4c20	gnome-keyring-d	1958	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6bf58d0	gnome-settings-	1960	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6bf32c0	gnome-power-man	1961	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6bf0cb0	gnome-settings-	1963	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6bf0000	upowerd	1964	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5474c20	gvfsd	1972	TRUE	TRUE	FALSE	FALSE	TRUE
0xf69f1960	gvfs-fuse-daemo	1982	TRUE	TRUE	FALSE	FALSE	TRUE
0xf69f58d0	gvfs-fuse-daemo	1985	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5182610	bluetooth-apple	1987	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5181960	gvfs-fuse-daemo	1988	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5180cb0	compiz	1990	TRUE	TRUE	FALSE	TRUE	TRUE
0xf5180000	gvfs-fuse-daemo	1991	FALSE	TRUE	FALSE	FALSE	FALSE
0xf50ed8d0	evolution-alarm	1993	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50eb2c0	nautilus	1996	TRUE	TRUE	FALSE	FALSE	TRUE
0xf50b0cb0	gnome-panel	1999	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6b30cb0	nm-applet	2002	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6bf2610	polkit-gnome-au	2009	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5472610	gvfs-gdu-volume	2013	TRUE	TRUE	FALSE	FALSE	TRUE
0xf512f230	polkitd	2019	TRUE	TRUE	FALSE	FALSE	TRUE
0xf51832c0	bluetooth-apple	2025	FALSE	TRUE	FALSE	FALSE	FALSE
0xf73fcc20	udisks-daemon	2028	TRUE	TRUE	FALSE	TRUE	TRUE
0xf69b6580	udisks-daemon	2031	TRUE	TRUE	FALSE	FALSE	TRUE
0xf687a610	nautilus	2035	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5473f70	gnome-panel	2038	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5476580	evolution-alarm	2040	FALSE	TRUE	FALSE	FALSE	FALSE
0xf69f2610	nm-applet	2049	FALSE	TRUE	FALSE	FALSE	FALSE
0xf69f6580	gvfs-gphoto2-vo	2069	TRUE	TRUE	FALSE	FALSE	TRUE
0xf69b7230	gvfs-afc-volume	2083	TRUE	TRUE	FALSE	FALSE	TRUE
0xf54758d0	gvfs-afc-volume	2089	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6bf3f70	gvfsd-trash	2092	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5128000	bonobo-activati	2145	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5098cb0	bonobo-activati	2147	FALSE	TRUE	FALSE	FALSE	FALSE
0xf687e580	bonobo-activati	2161	FALSE	TRUE	FALSE	FALSE	FALSE
0xf54f1960	wnck-applet	2167	TRUE	TRUE	FALSE	FALSE	TRUE

Offset(V)	Name	PID	Pslist	Pid_hash	Kmem_cache	Parents	Leaders
0xf69f4c20	trashapplet	2168	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5477230	compiz	2173	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6b90cb0	wnck-applet	2180	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6b91960	trashapplet	2181	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6b958d0	indicator-apple	2186	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6b96580	clock-applet	2187	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6b97230	indicator-apple	2188	TRUE	TRUE	FALSE	FALSE	TRUE
0xf51f8000	notification-ar	2189	TRUE	TRUE	FALSE	FALSE	TRUE
0xf51f8cb0	notification-ar	2190	FALSE	TRUE	FALSE	FALSE	FALSE
0xf51f9960	indicator-apple	2191	FALSE	TRUE	FALSE	FALSE	FALSE
0xf51fa610	indicator-apple	2192	FALSE	TRUE	FALSE	FALSE	FALSE
0xf51fb2c0	clock-applet	2193	FALSE	TRUE	FALSE	FALSE	FALSE
0xf51fd8d0	indicator-sound	2196	TRUE	TRUE	FALSE	FALSE	TRUE
0xf51ff230	indicator-messa	2199	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6b94c20	indicator-appli	2200	TRUE	TRUE	FALSE	FALSE	TRUE
0xf51fcc20	indicator-sessi	2205	TRUE	TRUE	FALSE	FALSE	TRUE
0xf51fe580	indicator-sessi	2206	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5280000	indicator-me-se	2208	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5280cb0	indicator-me-se	2209	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5281960	sh	2210	TRUE	TRUE	FALSE	TRUE	TRUE
0xf5282610	gtk-window-deco	2211	TRUE	TRUE	FALSE	FALSE	TRUE
0xf52832c0	gtk-window-deco	2213	FALSE	TRUE	FALSE	FALSE	FALSE
0xf5284c20	gvfsd-burn	2215	TRUE	TRUE	FALSE	FALSE	TRUE
0xf5286580	gnome-screensav	2218	TRUE	TRUE	FALSE	FALSE	TRUE
0xf52858d0	gnome-terminal	2220	TRUE	TRUE	FALSE	TRUE	TRUE
0xf5287230	gnome-terminal	2222	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6b90000	gnome-pty-helpe	2223	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6b93f70	bash	2224	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6879960	gnome-terminal	2225	FALSE	TRUE	FALSE	FALSE	FALSE
0xf6bf1960	gdu-notificatio	2237	TRUE	TRUE	FALSE	FALSE	TRUE
0xf6878cb0	applet.py	2240	TRUE	TRUE	FALSE	FALSE	TRUE

B.7 Output for plugin linux_lsmod

The following output was generated by the Volatility *linux_lsmod* plugin, as found in Section 3.6.1:

Table B.6: Plugin output for linux_lsmod (sorted by module name).

Base Address	Kernel Module	Size in Memory (in bytes)
f8070120	ac97_bus	1,014
f80e3480	agpgart	32,011
f80a6460	ahci	19,013
f98fc4e0	binfmt_misc	6,599
f82e95e0	bridge	68,008
f8251b40	drm	168,054
f80631a0	e1000	97,525
f80d99c0	hid	67,742
f8040d20	i2c_piix4	8,635
f80f21e0	ip6table_filter	1,275
f801d440	ip6_tables	11,764
f827e200	iptable_filter	1,302
f98caa60	iptable_nat	3,752
f811e0a0	ip_tables	10,460
f98d72a0	ipt_MASQUERADE	1,419
f9829580	ipt_REJECT	2,004
f803bca0	joydev	8,735
f800ef40	libahci	21,667
f80c83c0	lp	7,342
f98767c0	nf_conntrack	63,258
f98a4ea0	nf_conntrack_ipv4	10,783
f9895180	nf_defrag_ipv4	1,117
f98bdc80	nf_nat	16,289
f81a9000	nls_cp437	4,931
f802e9e0	parport	31,492
f8049660	parport_pc	26,058
f806d1a0	ppdev	5,556
f8116b40	psmouse	59,033
f8105b40	serio_raw	4,022
f829b0e0	snd	49,006
f820b600	snd_ac97_codec	99,227
f82260c0	snd_intel8x0	25,632
f8069380	snd_page_alloc	7,120

Base Address	Kernel Module	Size in Memory (in bytes)
f80bcc20	snd_pcm	71,475
f82ae1a0	snd_rawmidi	17,783
f828b9c0	snd_seq	47,174
f8108da0	snd_seq_device	5,744
f82bcca0	snd_seq_midi	4,588
f80a9da0	snd_seq_midi_event	6,047
f80368c0	snd_timer	19,067
f80260c0	soundcore	880
f801f300	stp	1,667
f80fbb00	usbhid	36,882
f81e3a40	vboxguest	211,970
f97ffa60	vboxsf	38,249
f80782a0	vboxvideo	1,279
f8075720	x_tables	15,921
f988b1a0	xt_state	1,014
f82b2520	xt_tcpudp	1,927

This page intentionally left blank.

Annex C VirusTotal submission and analysis for compiled rootkit sample

The following are copies of the two compiled rootkit associated libraries, *jynx2.so* and *reality.so*, respectively.

C.1 Library jynx2.so

The following is a copy of the report provided by VirusTotal upon analysing library file *jynx2.so* (rootkit file). The report provides clear indication that not a single scanner was able to detect the rootkit as malicious or infected.

virustotal

SHA256:	bd9ed228d63f1f380c9cebd158d047fd2c37fe1f6911a0b39e4ac72f32caf661
File name:	jynx2.so
Detection ratio:	0 / 53
Analysis date:	2014-10-16 17:49:20 UTC (0 minutes ago)

Antivirus	Result	Update
AVG	✔	20141016
AVware	✔	20141016
Ad-Aware	✔	20141016
AegisLab	✔	20141016
Agnitum	✔	20141015
AhnLab-V3	✔	20141016
Antiy-AVL	✔	20141016

Antivirus	Result	Update
Avast	✔	20141016
Avira	✔	20141016
Baidu-International	✔	20141016
BitDefender	✔	20141016
Bkav	✔	20141015
ByteHero	✔	20141016
CAT-QuickHeal	✔	20141016
CMC	✔	20141016
ClamAV	✔	20141016
Comodo	✔	20141016
Cyren	✔	20141016
DrWeb	✔	20141016
ESET-NOD32	✔	20141016
Emsisoft	✔	20141016
F-Prot	✔	20141016
F-Secure	✔	20141016
Fortinet	✔	20141016
GData	✔	20141016
Ikarus	✔	20141016
Jiangmin	✔	20141015
K7AntiVirus	✔	20141016
K7GW	✔	20141016
Kaspersky	✔	20141016
Kingsoft	✔	20141016

Antivirus	Result	Update
Malwarebytes	✔	20141016
McAfee	✔	20141016
McAfee-GW-Edition	✔	20141015
MicroWorld-eScan	✔	20141016
Microsoft	✔	20141016
NANO-Antivirus	✔	20141016
Norman	✔	20141016
Qihoo-360	✔	20141016
Rising	✔	20141016
SUPERAntiSpyware	✔	20141016
Sophos	✔	20141016
Symantec	✔	20141016
Tencent	✔	20141016
TheHacker	✔	20141013
TotalDefense	✔	20141016
TrendMicro-HouseCall	✔	20141016
VBA32	✔	20141016
VIPRE	✔	20141016
ViRobot	✔	20141016
Zillya	✔	20141016
Zoner	✔	20141014
nProtect	✔	20141016

C.2 Library reality.so

The following is a copy of the report provided by VirusTotal upon analysing library file *reality.so*. The report provides clear indication that not a single scanner was able detect it as malicious, infected or associated to the Jynx2 rootkit.

virustotal

SHA256:	9e7491bdad9b4fadb22157331ae216076bb40a108dac7235d8de04dfe4175a94
File name:	reality.so
Detection ratio:	0 / 53
Analysis date:	2014-10-16 17:49:56 UTC (0 minutes ago)

Antivirus	Result	Update
AVG	✔	20141016
AVware	✔	20141016
Ad-Aware	✔	20141016
AegisLab	✔	20141016
Agnitum	✔	20141015
AhnLab-V3	✔	20141016
Antiy-AVL	✔	20141016
Avast	✔	20141016
Avira	✔	20141016
Baidu-International	✔	20141016
BitDefender	✔	20141016

Antivirus	Result	Update
Bkav	✔	20141015
ByteHero	✔	20141016
CAT-QuickHeal	✔	20141016
CMC	✔	20141016
ClamAV	✔	20141016
Comodo	✔	20141016
Cyren	✔	20141016
DrWeb	✔	20141016
ESET-NOD32	✔	20141016
Emsisoft	✔	20141016
F-Prot	✔	20141016
F-Secure	✔	20141016
Fortinet	✔	20141016
GData	✔	20141016
Ikarus	✔	20141016
Jiangmin	✔	20141015
K7AntiVirus	✔	20141016
K7GW	✔	20141016
Kaspersky	✔	20141016
Kingsoft	✔	20141016
Malwarebytes	✔	20141016
McAfee	✔	20141016

Antivirus	Result	Update
McAfee-GW-Edition	✔	20141015
MicroWorld-eScan	✔	20141016
Microsoft	✔	20141016
NANO-Antivirus	✔	20141016
Norman	✔	20141016
Qihoo-360	✔	20141016
Rising	✔	20141016
SUPERAntiSpyware	✔	20141016
Sophos	✔	20141016
Symantec	✔	20141016
Tencent	✔	20141016
TheHacker	✔	20141013
TotalDefense	✔	20141016
TrendMicro-HouseCall	✔	20141016
VBA32	✔	20141016
VIPRE	✔	20141016
ViRobot	✔	20141016
Zillya	✔	20141016
Zoner	✔	20141014
nProtect	✔	20141016

Bibliography

Carbone, R. 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. TM 2009-161. Technical memorandum. Defence R&D Canada – Valcartier. January 2013.

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

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

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

Carbone, R. Malware memory analysis for non-specialists: Investigating publicly available memory image for the Stuxnet worm. DRDC-RDDC-2013-R1. Scientific report. DRDC – Valcartier Research Centre. November 2013.

Carbone, R. Malware memory analysis for non-specialists: Investigating publicly available memory image for the Tigger Trojan horse. DRDC-RDDC-2014-R28. Scientific report. DRDC – Valcartier Research Centre. June 2014.

Carbone, Richard. Malware memory analysis of the KBeast rootkit: Investigating publicly available Linux rootkits using the Volatility memory analysis framework. Scientific Report. DRDC-RDDC-2015-RXXX (FINAL DRAFT). DRDC – Valcartier Research Centre. September 2014.

Hale Ligh, Michael; Case, Andrew et al. The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Max Memory. Book. John Wiley & Sons. July 2014.

List of symbols/abbreviations/acronyms/initialisms

API	Application Programming Interface
AV	Anti-virus or antivirus
BIOS	Basic Input/Output System
CAF	Canadian Armed Forces
CFNOC	Canadian Forces Network Operation Centre
CPU	Central Processing Unit
DNS	Domain Name Service / Domain Name Server
DRDC	Defence Research and Development Canada
DTB	Directory Table Base
DVD	Digital Video Disc or Digital Versatile Disc
DVD +/- RW	Digital Video Disc +/- Read/Write
ECL	Export Control List
ELF	Executable and Linkable Format
FAC	Forces armées canadiennes
GB	Gigabyte ($1x10^9$)
GCC	GNU C Compiler
GHz	Gigahertz
GiB	Gibibyte (2^{30} bytes)
GID	Group ID
ID	Identification
IDT	Interrupt Descriptor Table
IGMP	Internet Group Management Protocol
IT	Information Technology
ITCU	Integrated Technological Crime Unit
KiB	Kibibyte (2^{10} bytes)
LKM	Loadable Kernel Module
MD5	Message-Digest Algorithm 5
NSRL	National Software Reference Library
PAE	Physical Address Extension

PAM	Pluggable Authentication Module
PC	Personal Computer
PCI	Peripheral Component Interconnect
PID	Process ID
PO Box	Post-Office Box or Post Office Box
PPID	Parent Process ID
R&D	Research & Development
RAM	Random Access Memory
RCMP	Royal Canadian Mounted Police
SATA	Serial ATA or Serial AT Attachment or
SHA1	Secure Hash Algorithm-1
SMP	Symmetric Multiprocessing
Syscall	System Call
TCP	Transmission Control Protocol
TI	Technologie de l'information
TM	Technical Memorandum
UDP	User Datagram Protocol
UID	User ID
USB2/3	Universal Serial Bus 2/3
UTC	Coordinated Universal Time
VM	Virtual Machine
x64	64-bit PC architecture
x86	32-bit PC architecture

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.) Defence Research and Development Canada – Valcartier 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 of the Jynx2 Linux rootkit (Part 1) : Investigating a publicly available Linux rootkit using the Volatility memory analysis framework

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.) October 2014	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.) 104	6b. NO. OF REFS (Total cited in document.) 15

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

Scientific Report

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

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

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.) 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-R176	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 is the second in a series that will examine Linux Volatility-specific memory malware-based analysis techniques. Windows-based malware memory analysis techniques were analysed in a previous series. Unlike these Windows-based reports, some of the techniques described therein are not applicable to Linux-based analyses including data carving and anti-virus scanning. Thus, with minimal use of scanner-based technologies, the author will demonstrate what to look for while conducting Linux-specific Volatility-based investigations. Each investigation consists of an infected memory image and its accompanying Volatility memory profile that will be used to examine a different open source rootkit. Some of the rootkits are user-land while others are kernel-based. Rootkits were chosen over Trojans, worms and viruses as rootkits tend to be more sophisticated. This specific investigation examines the Jynx2 rootkit. However, this analysis is broken into two parts. The first examines a system infected with Jynx2 but which has not yet loaded any new processes with the infected library/rootkit while the second examines a system completely infected by Jynx2. It is hoped that through the proper application of various Volatility plugins combined with an in-depth knowledge of the Linux operating system, these case studies will provide guidance to other investigators in their own analyses.

Ce rapport est le second d'une série examinant les techniques spécifiques d'analyse de logiciels malveillants en mémoire sous Linux à l'aide de l'outil Volatility. Les techniques d'analyse de logiciels malveillants en mémoire pour Windows ont été décrites dans des rapports précédents. Cependant, certaines de ces techniques, telles que la récupération de données et le balayage d'antivirus ne s'appliquent pas aux analyses sous Linux. Par conséquent, avec une utilisation minimale des technologies de balayage, l'auteur démontrera ce qu'il faut rechercher lorsqu'on effectue des investigations spécifiques à Linux avec Volatility. Chaque investigation consiste en une image mémoire infectée, accompagnée de son profile mémoire Volatility, et examinera un programme malveillant furtif à code source ouvert différent. Certains seront en mode utilisateur tandis que d'autres seront en mode noyau. Les programmes malveillants furtifs ont été préférés aux chevaux de Troie, vers et virus, car ils ont tendance à être plus sophistiqués. La présente investigation examine spécifiquement le programme malveillant furtif Jynx2. Cependant, cette analyse est divisée en deux parties. La première examine un système infecté par Jynx2 mais qui n'a pas encore chargé de nouveau processus qui utilise la librairie infectée tandis que la seconde examine un système complètement infecté par Jynx2. Il est souhaité qu'avec une utilisation adéquate de différents plugiciels Volatility et d'une connaissance approfondie du système d'exploitation Linux, ces études de cas fourniront des conseils à d'autres enquêteurs pour leurs propres analyses.

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; Computer infection; Computer memory forensics; Digital forensics; Digital memory forensics; Forensics; Infection; Jynx2; Linux; Malware; Memory analysis; Memory forensics; Memory image; Rootkit; Scanners; Virus scanner; Volatility