Enhancing incident response through forensic, memory analysis and malware sandboxing techniques

Wylie Shanks

Enhancing incident response through forensic, memory analysis and malware sandboxing techniques

GIAC (GCFA) Gold Certification

Author: Wylie Shanks, giac@infosecmatters.com
Advisor: Richard Carbone

Accepted: March 25, 2014

Abstract

This paper examines the important role of digital forensics, memory analysis and malware sandboxing in enhancing incident response practices. Methods for successful detection, eradication and recovery efforts will be explored through forensic and malware analysis techniques using Mandiant Redline, Volatility and Cuckoo Sandbox.

1. Introduction

Almost daily, there are reports of successful data breaches and new threat vectors including compromised systems or vulnerable software. While patching applications and operating systems provides moderate improvement in the overall security posture, adversaries require only one successfully exploited weakness to obtain their goal. According to (Cole, E., 2012), "A system that is 100% secure has 0% functionality". Meeting the needs of businesses and clients requires systems to be readily available for ease of use. As a result, networked and interconnected systems are inherently less than 100% secure. Therefore, incidents or system compromise becomes more likely to occur. Consequently, it is important to develop an incident response strategy to understand, cope with and rectify issues as they arise.

Understanding the nature of the issue before the appropriate response strategy is executed is of paramount importance. According to (Cichonski, Millar, Grance & Scarfone, 2012), an event is something that is observed in a system or network and may be benign such as sending an email or a connection blocked by a firewall. Adverse events have negative consequences such as unauthorized access to data or a system crash. Security incidents are the focus of most incident response plans. These incidents involve the violation or impending violation of organizational security policies or practices (Cichonski, Millar, Grance & Scarfone, 2012). Incident response plans should be executed when an incident is detected.

Through assessment and analysis of event logs and other artifacts, damage can be assessed and appropriate response and recovery strategies implemented. Without a strategy, incidents may take longer to resolve, increase the business' financial cost and impact the name, brand and reputation of the firm.

Effective incident response plans include an understanding of threats affecting the business. Once threat vectors are known, plans can be made to mitigate or rectify the issues. Proactive steps can also be taken to reduce the impact, severity and duration of the incident through pre-planned processes.

Wylie Shanks, giac@infosecmatters.com

In this paper the author examines various aspects of incident response along with methods to improve incident response practices. In the first section of this paper numerous facets of incidents response will be discussed. Next, the intrusion kill chain is discussed and the role it plays in locating and effectively responding to system intrusions. Important phases are outlined with the crucial role each phase plays in the response plan. The third and fourth sections examine methods of forensic and memory analysis within the context of discovering malware or system compromise. Automated malware analysis using Cuckoo Sandbox is discussed in section five. The paper concludes by analyzing malware samples using Cuckoo Sandbox, Volatility and Mandiant Redline before briefly mentioning eradication and recovery efforts.

The phases of incident response will now be discussed.

1.1. Preparation

Establishing incident response capability within an organization as a means of preventing incidents is the focus of this phase. Incident handling communication protocols including contact and escalation information for other team members, stakeholders and managers need to be documented and maintained. Vital to an incident response team's readiness is a secure facility within which to operate and access tools including laptops, blank media, evidence collection forms and equipment (e.g. write blockers, etc.). Incident handlers must be trained in their role including the detection, analysis, eradication of and recovery from incidents.

While incident response teams are generally not responsible for securing systems, they can act as advocates for generally accepted security practices. Such teams may also perform risk assessments, detect gaps and can make recommendations for remediation.

1.2. Identification

Identification of an incident regarding its scope and magnitude may be quite difficult for organizations. This is due to the volume of event data received and the training level and experience of the incident response team.

Incidents may be detected through a variety of means including third-party notification, intrusion detection systems (IDS), intrusion prevention systems (IPS), security information and event management systems (SIEMs), anti-virus products, file integrity checking products and log analysis.

Once the data has been analyzed a determination is made whether a security incident has occurred. Context Triggered Piecewise Hashing (CTPH) may be used to find files that are similar by calculating the hash value of portions of a file. Also known as fuzzy hashing, a percentage is calculated based on how similar the byte and sequence match is to the original file. Ssdeep is an application that implemented this technique (Kornblum, 2006).

1.3. Containment

Once an incident has been identified it is possible to implement containment procedures to limit spread of the issue and prevent further damage to the system. Considerations include: service availability requirements (e.g. a real-time system with high uptime requirements), whether evidence needs to be preserved, the efficacy of the solution (e.g. a temporary solution or workaround) and resource availability. Once the strategy has been implemented the incident should not spread or cause further system/network damage.

1.4. Eradication

The eradication process eliminates the cause of the unwanted incident. This process also implements mitigation controls, removing existing vulnerabilities in a prioritized, phased approach. Controls may include removing malware, changing user account permissions and passwords, adjusting firewall rules and patching exploited software vulnerabilities. In circumstances related to eliminating the incident, systems may need to be restored from clean backups or rebuilt using trusted media before current patches are applied. Otherwise, restoring backups or rebuilding systems from trusted media occurs in the recovery phase.

1.5. Recovery

Restoring the system to normal operation occurs during the recovery phase. Additional patching may be required in the recovery phase to eliminate future incidents. New firewall rules and the implementation of additional controls may also be required in this phase to ensure the newly recovered systems are not vulnerable to compromise. Additional controls may include the implementation of IDS, IPS, web content filter and anti-virus signatures designed to address the system weakness.

After recovery, the new system must be monitored to detect new attacks and to confirm the system is functioning correctly.

1.6. Lessons learned

Lessons learned provide a timely and useful means of updating security and incident handling practices and procedures with newly discovered insights. Valuable information surrounding improved awareness and detection of new threats may also be integrated. Questions that may be asked during a lessons learned meeting include:

- Were existing procedures adequate and were they followed?
- What inhibited the detection, containment, eradication and recovery efforts?
- How would the response be different in the future?
- What actions can be taken now to mitigate the risk of future incidents?

1.7. CSIRT

A computer security incident response team (CSIRT) provides incident handling capabilities within an organization. The objective of the CSIRT, its role, function and the services it provides to the organization are formalized. Multiple teams may be involved in the delivery of incident response services. For example, the team responsible for firewall administration may be responsible for investigating perimeter security systems, whereas system administrators would investigate host-based system security issues. An IT security team may partially or exclusively provide the role and function of a CSIRT (West-Brown et al., 2003).

A list of common CSIRT services includes:

- Incident handling and response including incident analysis and coordination.
- Vulnerability and security assessments.
- Forensic evidence collection and analysis.

2. What is the intrusion kill chain?

The kill chain is an integrated process comprised of locating adversary targets and their location, tracking and observing the target, engaging it and then assessing the results. This process is referred to as a "chain" because any disruption in the process affects the entire chain. This concept has been adapted to intrusions. This model includes breach of the trusted boundary, obtaining access to internal systems and obtaining the objective. Objectives may include moving within the internal environment or affecting the confidentiality, integrity or availability of the system (Hutchins, E. M., Cloppert, M. J., & Amin, R. M., n.d.).

Adversaries research and plan their attack in a phased approach. Understanding this methodology can aid the detection and prevention of current and future incidents. According to (Hutchins, E. M., Cloppert, M. J., & Amin, R. M., n.d.), "The intrusion kill chain is defined as reconnaissance, weaponization, delivery, exploitation, installation, command and control (C2) and actions on objectives."

The authors have defined each phase as follows:

1. **Reconnaissance** – Targets are identified and selected. Information may be obtained from websites and email address lists.
2. **Weaponization** – Automated tools are used to deliver an exploit within a payload. For example, Adobe PDF or Microsoft Office files may be updated with weaponized code.
3. **Delivery** – The transmission of the weaponized code to the selected target. Methods include website (e.g. watering hole), email (attachments or links) or removable media (e.g. USB device).
4. **Exploitation** – The code is executed. Typically, this exploits a software vulnerability or feature that allows other software to be installed.

5. **Installation** – On-going access to the target may be maintained through the installation of a backdoor or remote access software.
6. **Command and Control (C2)** – The compromised target establishes outbound communication with a command and control site. Once connected, the target receives further instructions.
7. **Actions on Objectives** – Depending on the adversary's objectives, data may be collected and exfiltrated. Alternatively, data or systems may be modified, encrypted or made unavailable. The target system may also be used to compromise additional systems within the environment.

2.1. Why is the intrusion kill chain important?

Understanding the adversary's attack methodology allows system owners to plan, budget for and implement appropriate detective and preventive controls. Table 1 illustrates common defensive and preventive capabilities, their applicability to intrusion kill chain phases and their ability to detect, deny, disrupt, degrade, deceive or destroy an attack.

Table 1: *Courses of Action Matrix*

Phase	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance	Web analytics	Firewall ACL				
Weaponization	NIDS	NIPS				
Delivery	Vigilant user	Proxy filter	In-line AV	Queuing		
Exploitation	HIDS	Patch	DEP			
Installation	HIDS	"chroot" jail	AV			
C2	NIDS	Firewall ACL	NIPS	Tarpit	DNS redirect	
Actions on Objectives	Audit log			Quality of Service	Honeypot	

Note. Adapted from *Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains*, p.5, by Hutchins, E. M., Cloppert, M. J., & Amin, R. M., (n.d.).

2.2. Kill chain analysis

Analysis of the full kill chain provides information that is relevant to the current incident. This data can be used to prevent future attacks. Without additional information from the earlier phases it is difficult to detect and prevent these incidents from occurring in the future. Unlike traditional incident response that occurs after the exploitation phase, the goal of kill chain analysis is detection of incidents much earlier on in the delivery phase.

3. Forensic Analysis

Relevant data needs to be identified, collected and analyzed in a manner that retains the data's integrity intact and maintains its chain of custody. Volatile data, which can change rapidly and is typically lost during a power outage, should be collected before non-volatile data (e.g. data written to a hard drive). (Lee, R. et al, 2012) provides a step-by-step process to find unknown malware:

1. Prep evidence / data reduction
2. Anti-virus check
3. Indicators of compromise search
4. Automated memory analysis
5. Evidence of persistence
6. Packing / entropy check (CTPH/ssdeep)
7. Review event logs
8. Super timeline examination
9. By-hand memory analysis
10. By-hand third party hash lookups
11. Master File Table (MFT) anomalies
12. File-time anomalies
13. Malware analysis

See Appendix C for links to other resources including software mentioned in this paper.

3.1. Memory imaging (Windows)

A memory image should be the first step taken in a suspected system compromise, as memory is the most volatile data within a system. There are many tools available to image memory including Mandiant Redline, MoonSols Windows Memory Toolkit Community Edition and commercial tools including EnCase and FTK. As with any forensic image acquisition, changing the data integrity of the artifact prior to acquisition should be avoided. According to (Carbone, R., Bean, C., Salois, M., 2011) the cold boot attack is a means of volatile data collection that may be considered in extreme circumstances, as it is not the method of choice.

Mandiant's Redline is a free Windows-based tool that can be used to collect memory images from Windows computers and analyze its contents for indicators of compromise (IOC). This tool is helpful during the triage process when determining if a system has been compromised.

MoonSols Windows Memory Toolkit Community Edition is another free memory imaging utility that creates raw memory images for use with many memory analysis tools.

EnCase Forensic v7 and FTK v5 provides multi-platform memory imaging capability. Once connected to the suspect machine the forensic analyst can create a memory image for later analysis.

3.2. File system imaging

Once memory has been imaged the file system should be imaged to separate storage media using forensically sound techniques. The first step when analyzing a file system image is reducing the amount of data to be analyzed. Hash values should be calculated for every file in the image and matched against the National Software Reference Library (NSRL) hash set to eliminate known files. Many advanced forensic analysis suites undertake these steps automatically. Unallocated space should be scanned for executable files and dynamic link libraries. Any files found should be extracted (e.g. data carving).

Mounting the image file as read-only allows for multiple anti-virus products to scan both the extracted files and image for known malware. If malware is found, files may also be scanned by ssdeep for similar malware.

3.3. Find and collect relevant artifacts

Several areas of the Windows operating system should be examined for relevant artifacts. These include:

Registry – Unknown or malicious files may run at startup (see run or runonce).

Services – Check the path, name and hash value of executable files.

Scheduled tasks – Malicious tasks may be run via the task scheduler.

Event logs – If sufficient auditing and logging was enabled there may be entries of interest to include in the overall incident timeline. Several events including logon, account creation and clearing of the event log are of interest during the review.

Volume Shadow Copy Service (VSS) – Data from the Volume Shadow Copy Services (VSS) can be extracted and reviewed using an advanced forensic analysis suite.

Prefetch – The same suite of tools can process the Windows prefetch directory.

3.4. Unknown hash matches

After filtering the files of interest list down to a reasonable number the hash values of the remaining files should be reviewed against the VirusTotal, Team Cymru and Bit9 FileAdvisor databases. One suggestion is to consider analyzing unmatched files in a malware sandbox environment in order to provide additional information.

4. Memory Analysis using Volatility

Programs in memory have fewer protection mechanisms against analysis and review. Relevant data is obtained by finding strings (human readable text), established network connections and through the analysis of running processes. Process injection, hiding processes, network and socket connections and the Windows registry are just a few of the many artifacts that can be reviewed or extracted.

Process injection involves inserting executable code into a running process. Executable program code may be typically inserted using the Windows application programming interface (API). The first steps involve connecting to the process then allocating memory within the process. The executable code is copied into the newly allocated space. Finally, the process executes the new code (Foundstone, 2013). Here are a few instructions that may be used in process injection:

Attach - OpenProcess()

Allocate Memory - VirtualAllocEx()

Copy executable - WriteProcessMemory()

Execute - CreateRemoteThread(), NtCreateThreadEx(), RtlCreateUserThread()

Process hiding is a technique found in many rootkits. Programs run in either user mode or kernel mode - the highest privilege. Applications typically run in user mode while operating system components run in kernel mode. A single virtual address space is used to run code in the kernel. Code in this space is not isolated from each other as it is in user mode. Running in kernel mode allows files, folders, registry entries, network ports or other information to be hidden using Direct Kernel Object Manipulation (DKOM).

Running processes use the EPROCESS kernel structure to point to the previous and next process. To hide a process, the previous and subsequent processes are linked together bypassing the intermediate process. This hides the process from products like Task Manager (Elia, 2005).

Network and socket connections are stored in memory. These artifacts may provide the source of an infection or simply indicate evidence of third-party communication. Unexpected connections may be benign or used for data exfiltration or command and control communication.

The Windows registry is loaded into memory upon the startup of Windows. It stores a vast amount of program and system data used by both applications and the operating system. Memory analysis of the registry can provide access to many artifacts including programs run at startup, the most recently used list (programs recently launched) and programs launched from the desktop, among others.

4.1. Indicators of Compromise (IOC)

Indicators of compromise are forensic artifacts that may be used to determine whether a system has been breached. Given the varied nature of malware and the approaches used to compromise systems, there are a wide variety of IOCs including registry items, browser cookies, event log entries, processes in memory, files on disk and file download history among others. These IOCs can be collected to create an IOC for a particular incident. Scanning systems for these IOCs is used to determine whether or not they have been compromised. Similarly, analysis of IOCs can be used to determine patterns and behaviors used by adversaries. This information may lead to improvements in detective and preventive capabilities in the future.

4.2. Locating malware

Volatility is a free memory analysis tool written in Python. It supports 32-bit and 64-bit versions of Windows, Linux 32-bit and 64-bit kernels (from 2.6.11 – 3.5.x), Mac OS X (10.5 – 10.9) and Android phone (32-bit and 64-bit) memory dumps (Volatility, 2013a). The focus of this paper is on Windows memory analysis.

Volatility has many plugins that are useful for identifying evidence of infection. A selection of these includes (Hale Ligh, Adair, Harstein & Richard, 2011):

- apihooks – detects hooks into user and kernel mode processes
- connections – a list of open connections
- connscan – a list of TCP connections
- dlllist – a list of loaded DLLs for each process
- malfind – detects hidden and injected code
- pslist – prints a list of loaded processes
- sockets - a list of open sockets
- sockscan – a list of open TCP sockets
- svcscan – detects hidden services

A sample memory image from the Malware Analyst's Cookbook was used for the following review. See Appendix C for the link to this memory file containing the Zeus malware memory sample.

The following commands were used to analyze the memory image (Malwareninja, 2011):

vol.py –f zeus.vmem imageinfo (result: WinXPSP2x86)

vol.py –f zeus.vmem connections (no results)

vol.py –f zeus.vmem connscan

Offset(P)	Local Address	Remote Address	Pid
0x02214988	172.16.176.143:1054	193.104.41.75:80	856
0x06015ab0	0.0.0.0:1056	193.104.41.75:80	856

A whois lookup was launched to determine the ownership and country of the IP address. The site http://www.malwareurl.com was consulted to determine whether the site has hosted malware. The detailed report shows that Trojan Zbot was hosted at this site. Next, the process ID (PID) was checked to see which process attempted the remote connection.

vol.py –f zeus.vmem pslist

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	...
0x80ff88d8	svchost.exe	856	676	29	336	0	...

The output shows svchost.exe attempted the connection and is indicative of suspicious activity. Web browsers would normally attempt an outbound connection on port 80. The malfind plugin is used to determine whether code has been injected into a process and can be limited to the suspicious process.

vol.py –f zeus.vmem malfind --pid=856

```
Process: svchost.exe Pid: 856 Address: 0xb70000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 38, MemCommit: 1, PrivateMemory: 1, Protection: 6
0x00b70000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
```

Two noteworthy items appear in the output. First, the process is executable yet its PrivateMemory is set to 1. This indicates that memory is not shared with other processes. The second noteworthy item is that its VadS protection is set to PAGE_EXECUTE_READWRITE. A scan of loaded DLLs indicates that there are no files loaded at address 0x00b70000. However, there appears to be a PE executable file header "MZ" at the start of the output (Volatility, 2013b). This code appears to have

been injected into svchost.exe. The suspected malicious code can be dumped for later inspection through this command:

vol.py –f zeus.vmem malfind --pid=856 --dump-dir=/tmp

md5sum /tmp/process.0x80ff88d8.0xb70000.dmp

Result: 59f1993ae96c0108f0fa224609f51a2f

VirusTotal reports that this file is identified at Zbot (malware).

5. Automated Malware Analysis – Cuckoo Sandbox

Cuckoo Sandbox is an automated malware sandboxing tool used to perform dynamic analysis against binary samples. An isolated Windows guest virtual machine is used to run the sample and analyze the results. As such, executable files, DLLs, URLs, PDF and Microsoft Office files among others can be analyzed in this environment. Cuckoo Sandbox has many features including the tracking of any files created, deleted or downloaded during execution, capturing a memory dump from the machine and capturing any network traffic created (Oktavianto, D., & Muhardianto, I., 2013).

The isolated environment allows the sample to run without adversely affecting the host or guest system. A guest Windows virtual machine is built and configured within supported virtual machine (VM) software such as VirtualBox. Once the desired system state is achieved a system snapshot is taken. This snapshot can be used to return the system to a known, clean state after the sample has been analyzed.

A command-line interface is used to perform all tasks within Cuckoo Sandbox. However, use of the web form may speed up the analysis process from sample submission through to report reviewing. Data from the report may be exported for use in other tools or reports and archived using other forensic artifacts.

5.1. Static analysis

Static analysis requires a strong understanding of programming code including low-level assembly language. Programs must be translated from binary code to human readable form through disassembly or decompilation. Once translated, the analyst reviews the numerous lines of code looking for relevant artifacts. This can

be a time consuming effort/undertaking and may be hampered by anti-debugging code that is designed to thwart code analysis (Oktavianto, D., & Muhardianto, I., 2013).

5.2. Dynamic analysis

Dynamic analysis involves executing the sample and monitoring its behavior. Changes to the file system, network and registry are recorded. The sample may require specific input or values in order to run correctly. It is imperative that these inputs are in place when executing the sample. In addition, some malware modifies its behavior if it detects that it is running in a virtual environment. This is a feature designed to thwart analysis of the sample. See Appendix B for more information regarding VirtualBox system hardening instructions.

5.3. Integration with Volatility

Cuckoo Sandbox integration with Volatility is optional. However, the reduction in time required to analyze a sample and the increase in potential indicators of compromise are worth the additional steps required to integrate the products. See Appendix A for installation instructions.

Cuckoo Sandbox's memory.conf configuration file is used to configure which Volatility options are to be used. Two additional configuration changes are required to enable this integration. The first change is to set *memory_dump = on* in file cuckoo.conf while the second change is to set *enabled=yes* in the *memory* section of file processing.conf.

6. Analysis of a malware sample

Samples may be submitted to the Cuckoo Sandbox once the environment has been configured. Malware samples used in the analysis below were obtained from https://zeustracker.abuse.ch.

A test environment was created as follows:

- Ubuntu 12.04.4 LTS server running VirtualBox and a Windows XP SP3 guest.
- Windows 7 Enterprise running Mandiant Redline and Volatility.

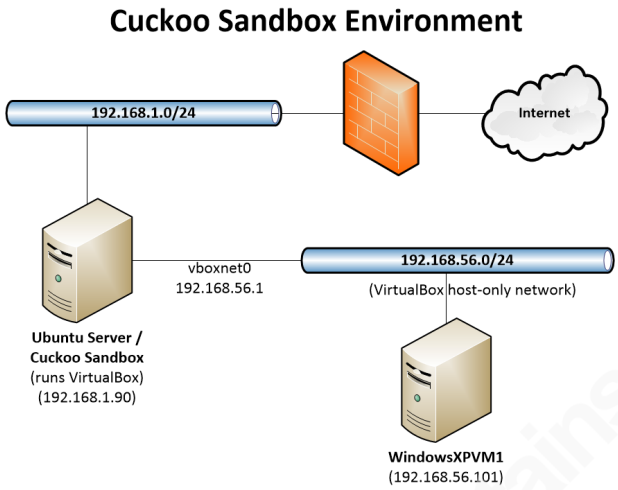

Figure 1 – Cuckoo Sandbox Environment

Before running Cuckoo Sandbox the Windows XP SP3 guest VM (WindowsXPVM1) was restored to Snapshot1 to ensure the environment was clean using this command:

vboxmanage snapshot "WindowsXPVM1" restore Snapshot1

The user ID sandboxuser was used to launch two separate terminal windows to start Cuckoo Sandbox and to enable the web interface. The following commands were used:

cd /opt/cuckoo

python /opt/cuckoo/cuckoo.py

python /opt/cuckoo/utils/web.py

If additional information is required the cuckoo sandbox process can be run in debug mode by appending --debug to cuckoo.py on the command line. The web form can be reached at http://192.168.1.90:8080 as shown below.

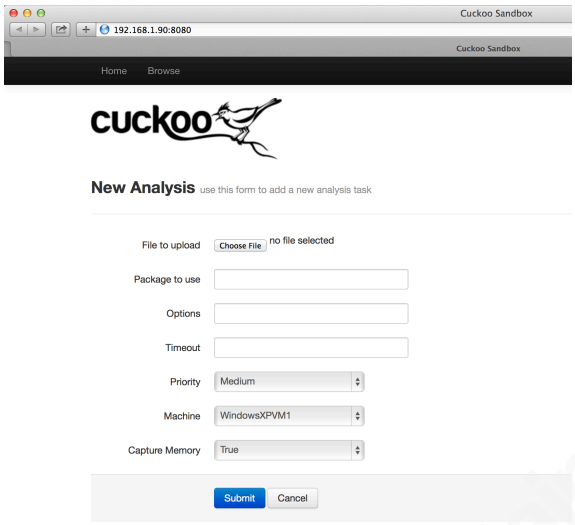

Figure 2 – Cuckoo Sandbox new analysis form

Samples may be submitted through the web interface or via a third terminal window:

python /opt/cuckoo/utils/submit.py <filename>

where <filename> is the name of the sample to be analyzed. For example:

python /opt/cuckoo/utils/submit.py /home/sandboxuser/samples/virussign.com_8f172d5dedfc25063878cf7ac2a6ed50.vir

Cuckoo Sandbox reports consist of multiple subsections that are discussed below through the use of the aforementioned virussign.com malware sample.

6.1. File Details

This section of the report provides basic file information such as file name, size and various hash values. These values may be useful in identifying this file in the environment. In addition, the ssdeep value may be used to find similar malware in the environment based on the similarity of its code. The existence of a packed executable file (e.g. UPX, ASPack, Themida, etc.) may indicate the file is malicious. Packers are used to compress files and have the added benefit of obfuscating human readable strings which makes string analysis much more difficult. The following information was reported for the sample file:

File name	virussign.com_8f172d5dedfc25063878cf7ac2a6ed50.vir
File size	223008 bytes
File type	PE32 executable (GUI) Intel 80386, for MS Windows
CRC32	D32D67D6

MD5	8f172d5dedfc25063878cf7ac2a6ed50
SHA1	76e9e285ccc11c68d72c48f12d45ba5d74947e0c
SHA256	0222ae7a94b2296471a1ff8e3a90049a86b84828ff3eda0b4c151ae70e52c59a
Ssdeep	3072:L2aACAMfVxHsjqUwkMejsRkudvR0FlgHIRXmUa9ITTYIQA3x/EFyWQIKxJ6:L2dMQRIR0FZXpTTYIQA3xA1IE
PEiD	None matched

6.2. Imports

DLLs imported (used) in the submitted program are listed in this section. Analyzing the functions used by these DLLs can give an indication of the type of activity performed by this program. A select list of imported DLLs and their function is shown below:

Library ADVAPI32.dll:
0x30001004 - RegEnumKeyExW
0x30001008 - RegCreateKeyExW
0x3000100c - RegOpenKeyExW
0x30001010 - RegSetValueExW
0x3000101c - RegQueryValueExW
0x30001024 - SetServiceStatus
0x30001030 - RegDeleteValueW
0x30001034 - RegisterServiceCtrlHandlerW
0x30001038 - CryptDestroyHash
0x3000103c - CryptCreateHash
0x30001040 - CryptGetHashParam
0x30001044 - CryptHashData
0x30001060 - RegDeleteKeyW
0x30001064 - SetFileSecurityW

Library WININET.dll:
0x300012a0 - InternetSetFilePointer
0x300012a4 - InternetOpenUrlW
0x300012a8 - InternetCloseHandle
0x300012ac - InternetReadFile
0x300012b0 - InternetOpenW
0x300012b4 - InternetGetConnectedState
0x300012b8 - InternetCanonicalizeUrlW
0x300012bc - HttpQueryInfoW

Functions that access, use and manipulate the registry are found in advapi32.dll along with hash creation and file security. Wininet.dll provides higher-level network functionality such as Internet access.

6.3. Dropped files

Seven files were dropped onto the file system as the malware sample executed. A list of these files, their hashes, ssdeep values and other details can be found in Appendix D. Of note, are the dropped files with the same name as those used by VirtualBox. Files VBoxDrvInst.exe and VBoxTray.exe are two of the seven dropped files.

6.4. Behavior Summary

As mentioned in the imports section, this malware sample appears to have the ability to perform a variety of capabilities. A sample of various file access attempts is:

```
C:\WINDOWS\system32\drivers\hmjkhn.sys
C:\DOCUME~1\cuckoo\LOCALS~1\Temp\dixobo.exe
R:\14eeef, S:\14f2f6, T:\14f708
C:\Python27\pythonw.exe
C:\DOCUMENTS AND SETTINGS\ALL USERS\APPLICATION DATA\MICROSOFT\CRYPTO\RSA\MACHINEKEYS\*
C:\DOCUMENTS AND SETTINGS\ALL USERS\APPLICATION DATA\MICROSOFT\CRYPTO\DSS\MACHINEKEYS\*
C:\Program Files\Oracle\VirtualBox Guest Additions\VBoxDrvInst.exe
C:\WINDOWS\system32\VBoxTray.exe
```

Several noteworthy items include the location of dropped files (the first two above-listed entries); drive letters that do not exist in the guest operating system, access to Python (used to connect with Cuckoo Sandbox), RSA/DSS crypto machine keys and the VirtualBox guest additions. As noted in Appendix D this malware sample may be virtual machine-aware.

6.5. Mutexes

The way that multithreaded applications limit access to shared resources is through the use of mutual exclusion (mutex). This ensures that only one thread has control of the resource at a time. Thus, mutexes are common in multithreaded applications. However, malware authors may want to limit the number of running instances of their application on a computer (so they do not re-infect an already compromised system) or perform their own multithreaded communication. Unique mutexes may be used to identify the malware. A sample of the mutexes discovered in the malware sample is:

```
C:\0p1mutx9
C:\smss.exeM_372_
C:\csrss.exeM_620_
C:\winlogon.exeM_644_
C:\services.exeM_688_
C:\lsass.exeM_700_
C:\vboxservice.exeM_856_
C:\svchost.exeM_908_
C:\svchost.exeM_984_
C:\svchost.exeM_1068_
C:\svchost.exeM_1124_
C:\svchost.exeM_1188_
C:\spoolsv.exeM_1476_
C:\explorer.exeM_1604_
C:\vboxtray.exeM_1732_
C:\pythonw.exeM_1748_
C:\pythonw.exeM_600_
C:\virussign.com_8f172d5dedfc25063878cf7ac2a6ed50.virM_1696_
```

An Internet search of "Op1mutx9" reported that this mutex has been used in other malware. The remaining mutexes append M_<process ID>_" to the file where <process ID> is the process ID of the running process. This will be discussed further in the process section.

6.6. Registry changes

The following sample of registry keys were noted in the Cuckoo Sandbox report:

HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\Authorized Applications\List\
HKLM\SOFTWARE\Microsoft\Cryptography\Defaults\Provider\
HKCU\System\CurrentControlSet\Control\SafeBoot\
HKLM\System\CurrentControlSet\Control\SafeBoot\
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\

Reading the firewall policy (authorized application list), default cryptography providers (e.g. smart cards), safeboot and the registry entry used for persistence (run) are further indicators of this malware's potential capabilities.

6.7. Processes

Each of the identified processes in the Cuckoo Sandbox report, when clicked, opens to display activity conducted. The timestamp, thread, function, arguments, status and return code are displayed. The processes discovered were:

virussign.com_8f172d5dedfc25063878cf7ac2a6ed50.vir PID: 1696, Parent PID: 600
netsh.exe PID: 1032, Parent PID: 1696
Explorer.EXE PID: 1604, Parent PID: 1584
VBoxTray.exe PID: 1732, Parent PID: 1604
wscntfy.exe PID: 1988, Parent PID: 1068
netsh.exe PID: 464, Parent PID: 1604

This section provides a wealth of information as to the activities conducted by the sample. For example, a search for the term firewall in the virussign.com_8f172d5dedfc25063878cf7ac2a6ed50.vir process shows two successful registry changes to Firewall Disable Notify (which disables the pop-up alert in Windows XP) and FirewallOverride. The Windows firewall is then disabled.

23:02:44,334	1724	RegSetValueExA	Handle => 0x000000c4 Buffer => 1 ValueName => FirewallDisableNotify Type => 4	SUCCESS	0x00000000
23:02:44,334	1724	RegSetValueExA	Handle => 0x000000c4 Buffer => 1 ValueName => FirewallOverride Type => 4	SUCCESS	0x00000000
23:02:44,495	1724	CreateProcessInternalW	ApplicationName => ProcessId => 1032 CommandLine => netsh firewall set opmode disable ThreadHandle => 0x000000d8 ProcessHandle => 0x000000d4 ThreadId => 1756 CreationFlags => 0x00000000	SUCCESS	0x00000001

The processes list includes the process ID and parent ID. The parent ID denotes the process that spawned the child process. Interestingly, VBoxTray.exe is spawned by explorer.exe. By examining data reported on from the integration with Volatility additional indicators of compromise are obtained.

6.8. Volatility

Using Volatility plugins additional artifacts were located that provided indication of malicious activity.

6.8.1. Malfind

The malfind plugin detects injected processes. The twelve processes shown below were identified as possibly injected. The processes will be interrogated in a later step to confirm whether or not they are injected. As noted earlier, the PID and process name appears in the mutexes previously discovered.

PID	Process name	VAD start	VAD tag
620	csrss.exe	0x7f6f0000	Vad
644	winlogon.exe	0xb790000	VadS
644	winlogon.exe	0x19c0000	VadS
644	winlogon.exe	0x475f0000	VadS
644	winlogon.exe	0x3bc30000	VadS
644	winlogon.exe	0x15410000	VadS
644	winlogon.exe	0xfd40000	VadS
644	winlogon.exe	0x3ce40000	VadS
644	winlogon.exe	0x66990000	VadS
644	winlogon.exe	0x6b740000	VadS
1748	pythonw.exe	0x11a0000	VadS
1748	pythonw.exe	0x12b0000	VadS

6.8.2. PSXView

This plugin assists with finding hidden processes. Several processes in the excerpted list below have a value of "false" including PIDs 600 (pythonw.exe), 1604 (explorer.exe) and 464 (netsh.exe). A value of false in a column indicates that the process is missing from that source of process listings. For example, netsh.exe does not appear in a list generated using the pslist plugin. Therefore, these processes may be hidden and will be analyzed.

PID	Process name	In pslist	In psscan	In thrdproc	In pspcid	In csrss	In session	In deskthrd
448	wuauclt.exe	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
464	netsh.exe	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
600	pythonw.exe	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE
620	csrss.exe	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE
644	winlogon.exe	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
700	lsass.exe	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
1604	explorer.exe	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE
1748	pythonw.exe	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
1836	wscntfy.exe	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

Two of the four processes of interest PID 600 (pythonw.exe) and PID 1604 (explorer.exe) could not be dumped as they were paged to disk:

Command: *vol.py -f memory.dmp proexedump -p 600 --dump-dir=/tmp*

Result: Error: PEB at 0x7ffde000 is paged

Command: vol.py -f memory.dmp procexedump -p 1604 --dump-dir=/tmp

Result: Error: PEB at 0x7ffd4000 is paged

The next process (netsh.exe – PID 464) could not be dumped via neither the process ID or offset (0x063f8538). The offset was obtained by running the psxview command from the command line, as it did not appear on the Cuckoo Sandbox report.

Command: vol.py -f memory.dmp vaddump --offset=0x063f8538 --dump-dir=/tmp

Result: Unable to get process AS for -. [This indicates that volatility was unable to get the process address space for that file and, therefore, was unable to dump the process.]

The final process PID 620 is analyzed in section 6.9.

6.9. Analysis of dropped and extracted files

The dropped files from section 6.3 were submitted to virustotal.com for analysis. Known malware was found in most of the submitted samples.

Filename	Result
npeai.exe	Win32/Sality
dixobo.exe	probably safe
hmkjhn.sys	Win32/Sality
VBoxTray.exe	Win32/Sality
VBoxDrvInst.exe	Win32/Sality

An analysis of the PIDs listed in section 6.8.1 indicates malicious code was injected into running processes. The following commands were used to dump the memory of various processes:

vol.py -f memory.dmp malfind --pid=1748 --dump-dir=/tmp/7

vol.py -f memory.dmp malfind --pid=644 --dump-dir=/tmp/7

vol.py -f memory.dmp malfind --pid=620 --dump-dir=/tmp/7

The mutex (pythonw.exeM_1748) appears in the hex dump of the malfind output from PID 1748. A review of the dumped file names indicates the process was at 0x864cc410.0x12b0000.

The MD5 hash values were calculated for the dumped processes using the command *md5sum /tmp/7*. The calculated hash values were submitted to virustotal.com to see if they had been previously analyzed. Files were submitted to virustotal.com where no results were received from the hash value searches. An excerpt of the dumped files, hash values and virustotal.com results are show below:

PID 1748 (pythonw.exe) – Detected as Sality by VirusTotal.
25def264c460916a18318dea8002dd90 /tmp/7/process.0x864cc410.0x11a0000.dmp
410c6ad62fd24bbd6024471a6a6db452 /tmp/7/process.0x864cc410.0x12b0000.dmp

PID 644 (winlogon.exe) – VirusTotal did not detect malware in the extracted files.
c5265eae501e49fb83cda1efde6e3323 /tmp/7/process.0x86508b28.0x15410000.dmp
0699a7324afe1ad2c17c870abe2fcfb7 /tmp/7/process.0x86508b28.0x19c0000.dmp

PID 620 (csrss.exe) – VirusTotal did not detect malware in the extracted files.
d320f3e7b378c50d3122ac0361aabff7 /tmp/7/process.0x86559ce0.0x7f6f0000.dmp

6.10. Mandiant Redline

Mandiant Redline analyzed the Zeus memory file mentioned in section 4. 2 and reported the presence of malware in the memory image. The first indication of malware was reported by the Injected Memory Sections filter.

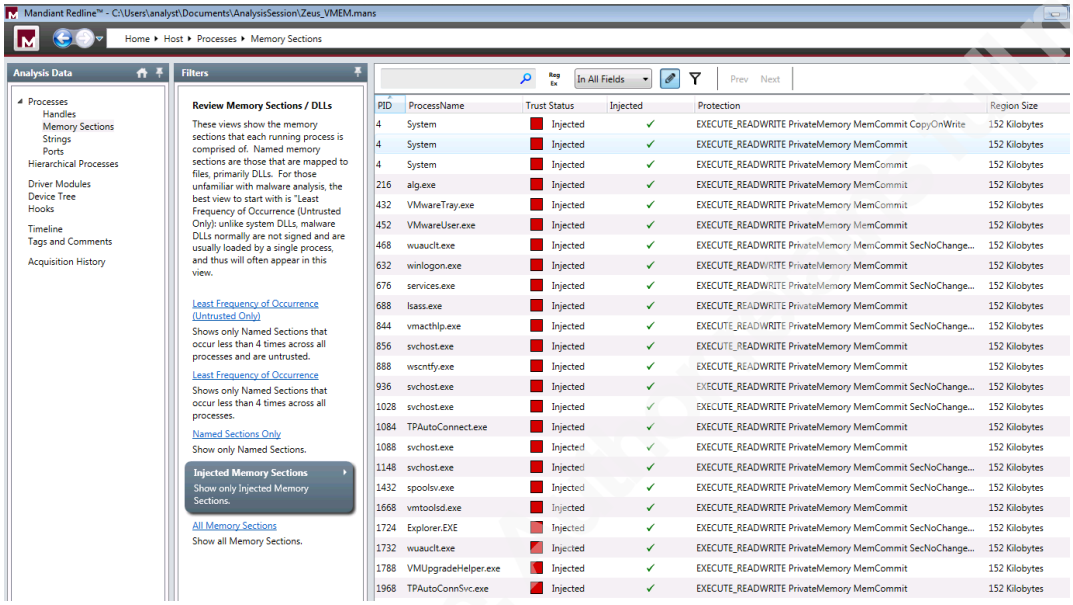

Figure 3 – Injected Memory Sections

The trust status column clearly indicates which processes have been injected. An additional indication of process injection is listed in the protection column. The value of EXECUTE_READWRITE_PrivateMemory indicates that the process has executable private (non-shared) memory allocated to the process.

The injected processes were carved out of the memory image using the "Acquire this Process Address Space" command.

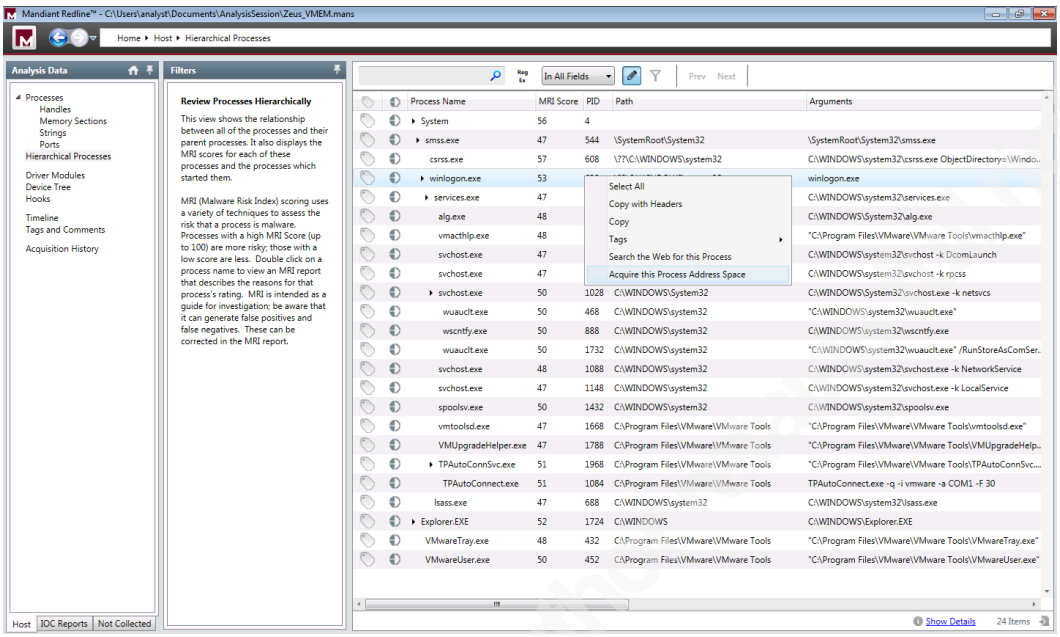

Figure 4 – Acquire this Process Address Space

Of the eight injected files the author attempted to acquire from process address space Symantec Norton 360 Premier Edition blocked them both. The acquired files were verified to be malicious. These files could have been submitted to VirusTotal for further analysis had they not been detected as malicious.

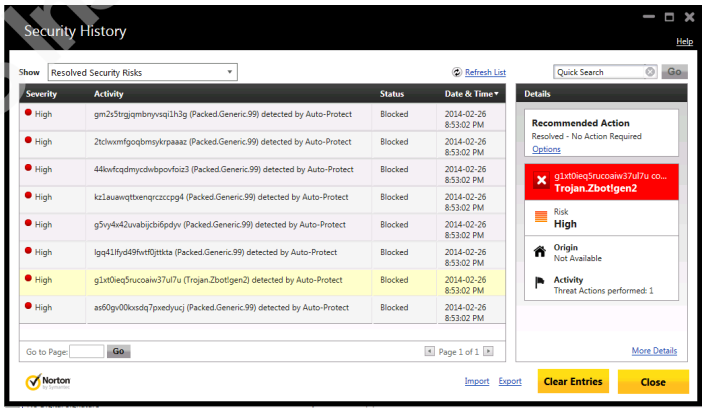

Figure 5 – Security History – Blocked process acquisition

Wylie Shanks, giac@infosecmatters.com

7. Eradication and recovery efforts

Having discovered malware and additional indicators of compromise it is now possible to eradicate the problem and recover the system. Eradication can involve the creation of an anti-virus signature by your anti-virus software vendor. This may be especially helpful if the infection is wide spread within the organization as the anti-virus software can be used to scan systems and automatically remove the infection.

Submission of the malware sample to VirusTotal (or other providers listed in the appendix) may result in the creation of new malware signatures for your anti-virus product. In addition, many anti-virus vendors have a malware submission option via their website. There may be security, confidentiality, or legal considerations when submitting a sample for analysis. Prior approval should be obtained before submitting any file.

Restoring the system from a trusted backup may provide higher assurance that the infection has been removed. If it is not possible to restore from a trusted backup then the infected system may need to be rebuilt from trusted media and updated to current patch levels.

8. Conclusion

Incident response practices can be enhanced through forensics, memory image analysis and malware sandboxing techniques. Forensic techniques can be used to locate indicators of compromise and find evidence of persistence. Analysis of memory images may provide additional indicators of comprise including hooked processes and modified registry keys (e.g. processes to run at startup). Similarly, malware sandboxing techniques provide behavioral analysis of the sample and provide additional information including dropped files, changed registry keys and the disabling of services among many others.

Armed with a means of detecting malicious software, the eradication and recovery efforts can be completed effectively in order to restore the system to a known state. In addition, information gathered during intrusion kill chain analysis

can help protect the organization from future attacks through modification of the internal defense systems (e.g. patching, HIDS, NIDS, NIPS, anti-virus, firewall etc.).

References

Carbone, R., Bean, C., Salois, M. (2011). *An in-depth analysis of the cold boot attack – Can it be used for sound forensic acquisition*. Retrieved from website: http://cradpdf.drdc-rddc.gc.ca/PDFS/unc108/p534323_A1b.pdf

Cichonski, P., Millar, T., Grance, T., & Scarfone, K. National Institute of Standards and Technology, (2012). *Computer security incident handling guide* (Special Publication 800-61 Revision 2). Retrieved from website: http://csrc.nist.gov/publications/nistpubs/800-61rev2/SP800-61rev2.pdf

Cole, E. (2012). *Advanced persistent threat: Understanding the danger and how to protect your organization*. Syngress.

Elia, F. (2005). *When malware meets rootkits*. Retrieved February 17, 2014 from https://www.symantec.com/avcenter/reference/when.malware.meets.rootkits.pdf

Foundstone (2013). *Windows dll injection basics*. Retrieved on February 17, 2014 from http://blog.opensecurityresearch.com/2013/01/windows-dll-injection-basics.html

Hale Ligh, M., Adair, S., Harstein, B., & Richard, M. (2011). *Malware analyst's cookbook and dvd: Tools and techniques for fighting malicious code*. Indianapolis, Indiana: Wiley Publishing Inc.

Hutchins, E. M., Cloppert, M. J., & Amin, R. M. (n.d.). *Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains*. Retrieved from http://www.lockheedmartin.ca/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf

Kornblum, J. (2006). *Identifying almost identical files using context triggered piecewise hashing.* Retrieved February 14, 2014 from http://dfrws.org/2006/proceedings/12-Kornblum.pdf

Lee, R. et al (2012). *Digital forensics and incident response poster: finding unknown malware - step-by-step*. Retrieved February 8, 2014 from https://blogs.sans.org/computer-forensics/files/2012/06/SANS-Digital-Forensics-and-Incident-Response-Poster-2012.pdf

Malwareninja. (2011). *Zeus analysis in volatility 2.0*. Retrieved February 17, 2014 from http://malwarereversing.wordpress.com/2011/09/23/zeus-analysis-in-volatility-2-0/

Oktavianto, D., & Muhardianto, I. (2013). *Cuckoo Malware Analysis*. Packt Publishing.

Sikorski, M., & Honig, A. (2012). *Practical malware analysis: The hands-on guide to dissecting malicious software*. No Starch Press.

Special Report - The Department of Energy's July 2013 Cyber Security Breach. (2013, December). Retrieved February 8, 2014, from http://energy.gov/sites/prod/files/2013/12/f5/IG-0900.pdf

Volatility (2013a). *The volatility framework*. Retrieved February 9, 2014 from https://code.google.com/p/volatility/

Volatility (2013b). *Volatility framework-commandreferencemal23*. Retrieved February 14, 2014 from https://code.google.com/p/volatility/wiki/CommandReferenceMal23

West-Brown, M. J., Stikvoort, D., Kossakowski, K., Killcrece, G., Ruefle, R., & Zajicek, M. (2003, April). *Handbook for computer security incident response teams (CSIRTs)*. Retrieved February 2, 2014, from http://www.cert.org/archive/pdf/csirt-handbook.pdf

Appendix A: Deploy Cuckoo Sandbox on Ubuntu 12.04

References:
http://docs.cuckoosandbox.org/en/latest/
http://santi-bassett.blogspot.ca/2013/01/installing-cuckoo-sandbox-on-virtualbox.html
http://santi-bassett.blogspot.ca/2013/01/setting-up-windows-guest-on-virtualbox.html
https://media.readthedocs.org/pdf/cuckoo/latest/cuckoo.pdf
https://www.virtualbox.org/wiki/Linux_Downloads

Download the latest the long-term support (LTS) version of Ubuntu 12.04:
http://www.ubuntu.com/download/server

Install Ubuntu 12.04 and allow SSH login (if remote console access is desired)

Login then switch to root access to install the software. Open terminal if console access is not already present.

To login as root: sudo su –
apt-get update
apt-get dist-upgrade
(install the XFCE4 or other GUI environment if desired: apt-get install xfce4)
apt-get install unzip

Install VirtualBox
Add the following line to /etc/apt/sources.list:
deb http://download.virtualbox.org/virtualbox/debian precise contrib
wget -q http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc -O- | sudo apt-key add –
apt-get update
apt-get install virtualbox-4.3
/etc/init.d/vboxdrv setup

Set static IP address (example):

vi /etc/network/interfaces

```
# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.1.90
        netmask 255.255.255.0
        gateway 192.168.1.1
        dns-nameservers 192.168.1.1 8.8.8.8

auto vboxnet0
iface vboxnet0 inet dhcp
```

Wylie Shanks, giac@infosecmatters.com

Install VirtualBox Extension Pak

wget
http://dlc.sun.com.edgesuite.net/virtualbox/4.3.6/Oracle_VM_VirtualBox_Extension_Pack-4.3.6-91406.vbox-extpack
vboxmanage extpack install Oracle_VM_VirtualBox_Extension_Pack-4.3.6-91406.vbox-extpack

Install Python

apt-get install python python-dpkt python-jinja2 python-magic python-pymongo mongodb python-libvirt python-bottle python-pefile python-mako python-sqlalchemy
apt-get install ssdeep python-pip dwarfdump
apt-get install build-essential git libpcre3 libpcre3-dev libpcre++-dev

cd /opt
apt-get install subversion python-pyrex libfuzzy-dev libcap2-bin tcpdump
svn checkout http://pyssdeep.googlecode.com/svn/trunk pyssdeep
cd pyssdeep
python setup.py build
python setup.py install
pip install pydeep

Configure tcpdump

setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump
getcap /usr/sbin/tcpdump

Install Yara

cd /opt
git clone https://github.com/plusvic/yara.git yara
cd yara
apt-get install libtool automake
chmod +x build.sh
./build.sh

Install Volatility: https://code.google.com/p/volatility/wiki/LinuxMemoryForensics

cd /opt
wget https://volatility.googlecode.com/files/volatility-2.3.1.tar.gz
tar -zxvf volatility-2.3.1.tar.gz
rm volatility-2.3.1.tar.gz

Cuckoo Sandbox must run as the same user that creates the virtual machines. New user "cuckoo" was created. Since we are using VirtualBox the new user must be assigned to

the "vboxusers" group. This is the group used to run VirtualBox.

Create a new user

adduser cuckoo
sudo adduser cuckoo sudo
usermod –G vboxusers cuckoo
usermod –G libvirtd cuckoo (if KVM or any libvirt based module is used)

Install Cuckoo Sandbox

cd /opt
git clone git://github.com/cuckoobox/cuckoo.git

Set Cuckoo Sandbox configuration

vi /opt/cuckoo/conf/virtualbox.conf
[virtualbox]
mode = **headless**
path = /usr/bin/VBoxManage
machines = **WindowsXPVM1**
[WindowsXPVM1]
label = **WindowsXPVM1**
platform = windows
ip = **192.168.56.101**

Configuring integration with Volatility

vi /opt/cuckoo/conf/processing.conf
[memory]
enabled = **yes**

Change memory disabled (no) to enabled (yes)

vi conf/memory.conf
Enable or disable settings as desired.

vi conf/cuckoo.conf
memory_dump = **on**
To enable dumping of memory to a file for analysis

Create a directory to store the Windows XP install media then change ownership to cuckoo ID

mkdir /opt/cuckoo/ISO
chown –R cuckoo /opt/cuckoo

To enable the IP address configured earlier:

ifdown eth0
ifup eth0

Transfer the Windows XP installation media (ISO) into the Ubuntu host. sFTP may be used if openssh was installed earlier. Otherwise, the media may be installed from a USB device. The ISO file was stored as follows: /opt/cuckoo/ISO/windowsxp.iso.

Installing the Windows XP guest

Switch user to cuckoo before continuing with these steps. Type:
sudo su cuckoo

Create the Windows XP VirtualBox guest:
vboxmanage createvm --name "WindowsXPVM1" --ostype WindowsXP --register
vboxmanage modifyvm "WindowsXPVM1" --memory 1000 --acpi on --boot1 dvd --nic1 nat
vboxmanage createhd --filename "WinXP.vdi" --size 20000
vboxmanage storagectl "WindowsXPVM1" --name "IDE Controller" --add ide --controller PIIX4
vboxmanage storageattach "WindowsXPVM1" --storagectl "IDE Controller" --port 0 --device 0 --type hdd --medium "WinXP.vdi"
vboxmanage storageattach "WindowsXPVM1" --storagectl "IDE Controller" --port 0 --device 1 --type dvddrive --medium /opt/cuckoo/ISO/windowsxp.iso

Remote Desktop must be used to connect to 192.168.1.90 from another machine if the virtual machine is started "headless". To start the headless virtual machine and install Windows XP type:
VBoxHeadless --startvm "WindowsXPVM1"

vboxmanage controlvm "WindowsXPVM1" poweroff
mkdir -p /home/cuckoo/shares/WindowsXPVM1
vboxmanage sharedfolder add "WindowsXPVM1" --name "WindowsXPVM1" --hostpath /home/cuckoo/shares/WindowsXPVM1 --automount
vboxmanage sharedfolder add "WindowsXPVM1" --name setup --hostpath /home/cuckoo/shares/setup --automount --readonly
vboxmanage modifyvm "WindowsXPVM1" --nictrace1 on --nictracefile1 /home/cuckoo/shares/WindowsXPVM1/dump.pcap
vboxheadless --startvm "WindowsXPVM1"

lsmod | grep vboxnetadp # module needed to add a new host-only interface at the host
vboxmanage list hostonlyifs # checks host-only interfaces at the host
vboxmanage hostonlyif create # leaving default IP 192.168.56.1/24
vboxmanage list dhcpservers # checks dhcp servers
vboxmanage list vms # checks virtual machines

```
vboxmanage showvminfo "WindowsXPVM1" # checks NICs information
vboxmanage controlvm "WindowsXPVM1" poweroff
vboxmanage modifyvm "WindowsXPVM1" --nic1 hostonly
vboxmanage modifyvm "WindowsXPVM1" --hostonlyadapter1 vboxnet0
vboxheadless --startvm WindowsXPVM1
```

vi /etc/rc.local file

```
iptables -A FORWARD -o eth0 -i vboxnet0 -s 192.168.56.0/24 -m conntrack --ctstate
NEW -j ACCEPT
iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A POSTROUTING -t nat -j MASQUERADE
sysctl -w net.ipv4.ip_forward=1
```

Windows XP guest configuration can be found here:
http://santi-bassett.blogspot.ca/2013/01/installing-cuckoo-sandbox-on-virtualbox.html

Ensure the following are completed:

- Disable Windows Firewall.
- Disable Automatic Updates.
- Install Python 2.7.6 (download and copy to shared folder)
- Install PIL (for Python 2.7 – download and copy to shared folder)
- Copy /opt/cuckoo/agent/agent.py to the WindowsXPVM1 shared folder. From the Windows XP guest copy agent.py to a startup folder. Rename the file agent.pyw.
- Activate Windows. It may be necessary to temporarily enable Internet access in order to do so. Ensure Internet access is disabled after Windows activation.
- Enter the static IP address of the guest OS as 192.168.56.101 and 192.168.56.1 as the gateway. Enter your DNS address or 8.8.8.8.
- Ping 192.168.56.1 to confirm that network is working as intended.
- Enable Autologon.
- Reboot the Windows XP guest.
- Ensure agent.pyw begins at startup and is running before taking the virtual machine snapshot.

Appendix B: Hardening Cuckoo Sandbox (VirtualBox)

Some malware will check to see if it is running in a virtual environment and may change how it runs as a result. To avoid detection by virtual machine-aware malware running in a VirtualBox environment please see the following resources:

Read the instructions found at this site:
http://www.alienvault.com/open-threat-exchange/blog/hardening-cuckoo-sandbox-against-vm-aware-malware

Implement all file changes (in green) listed here:
https://github.com/jaimeblasco/AlienvaultLabs/blob/master/cuckoomon_hardened/cuckoomon_vbox_hardened.patch

Additional hardening instructions are located here:
http://0xmalware.blogspot.ca/2013/10/cuckoo-sandbox-hardening-virtualbox.html

The cuckoomon.dll file can be compiled using the instructions found here:
http://blog.michaelboman.org/2013/05/cross-compiling-cuckoomon-under-linux.html

Appendix C: Other resources

The Honeynet Project: http://honeynet.org/project
VirusSign (Malware research & data center): http://www.virussign.com
Clearing house for incident handling tools:
http://www.enisa.europa.eu/activities/cert/support/chiht
Request tracker for incident response: http://www.bestpractical.com/rtir/index.html
Mandiant Redline: https://www.mandiant.com/resources/download/redline
MoonSols Windows Memory Toolkit Community Edition:
http://www.moonsols.com/windows-memory-toolkit/

URL review / categorization:

BlueCoat WebPulse Site Review: http://sitereview.bluecoat.com/sitereview.jsp
Symantec Safe Web: http://safeweb.norton.com/
Websense CSI: ACE Insight: http://csi.websense.com
FortiGuard Center: http://www.fortiguard.com/ip_rep.php
The Anti-Abuse Project: http://www.anti-abuse.org/multi-rbl-check
Spamhaus (SPAM block list): http://www.spamhaus.org/sbl
NoVirusThanks: http://www.novirusthanks.org/services/
MalwareURL: http://www.malwareurl.com

Online file/URL scanners:

Anubis (Online malware scanner): https://anubis.iseclab.org
Online Cuckoo Sandbox: https://malwr.com/
Metascan online: https://www.metascan-online.com/en
VirusTotal: www.virustotal.com
Wepawet: http://wepawet.iseclab.org
Threat Expert: http://www.threatexpert.com
Free URL scanner: https://urlquery.net/

File / hash value verification:

Bit9 FileAdvisor: http://fileadvisor.bit9.com/services/search.aspx
Malware Hash Registry: http://www.team-cymru.org/Services/MHR/
National Software Reference Library (NSRL): http://www.nsrl.nist.gov/Downloads.htm
WinMHR (beta): http://www.team-cymru.org/Services/MHR/WinMHR

Indicators of Compromise (IOCs):

OpenIOC: http://openioc.org/

Malware Analysts Cookbook (Zeus memory sample):

https://malwarecookbook.googlecode.com/svn-history/r4/trunk/17/1/zeus.vmem.zip

Wylie Shanks, giac@infosecmatters.com

Appendix D: Analyzed malware sample (dropped files)

File name	hmjkhn.sys
File size	5669 bytes
File type	PE32 executable (native) Intel 80386, for MS Windows
MD5	8ac1e580cf274b3ca98124580e790706
Ssdeep	96:eYtNn0TXtPVSDHawANDfq4bV1f7fn/33dMg7D:eYD0TXNVCLANT/b7n9Mg/

File name	dixobo.exe
File size	2188928 bytes
File type	PE32 executable (native) Intel 80386, for MS Windows
MD5	0c89243c7c3ee199b96fcc16990e0679
Ssdeep	24576:S0IYOl102l/BjEhDhgc8AR7CLVu/xheKyPSuKV94IJ1H4c9jXHb4MYc7P5e9E4Dc:St109YvwDpjk85AT2EYNRqZjWwH+P
Yara	• shellcode (Matched shellcode byte patterns)

File name	SYSTEM.INI
File size	267 bytes
File type	ASCII text, with CRLF line terminators
MD5	70f7a2124f43bad7d78553154d4eb6ab
Ssdeep	6:aQ44VvYIie0xTqFbqSQpH3BYf1fyBcfjfcUG:F4Yv1wqFeSQN3OUBq0UG

File name	VBoxTray.exe
File size	1381648 bytes
File type	PE32 executable (GUI) Intel 80386, for MS Windows
MD5	a827dc1e052878ed19f3f06a228be306
Ssdeep	24576:Ej/81Q/hUanJPOyYVia1kRVtrldx5XeKN1xQIb:Ej/8e5UantOyYViNVRldrXeKN1p
Yara	• shellcode (Matched shellcode byte patterns) • vmdetect (Possibly employs anti-virtualization techniques)

File name	npeai.exe
File size	146944 bytes
File type	PE32 executable (GUI) Intel 80386, for MS Windows
MD5	9609b9868c42aed0f5487bfc70efb5a6
Ssdeep	3072:lNQKPWDyXI0fFJltZrpReFX3YPVcjFQiJFjzdHKmFU:lNSDyXIkFthpb+FQezdHKmq
Yara	• shellcode (Matched shellcode byte patterns)

File name	VBoxDrvInst.exe
File size	214288 bytes
File type	PE32 executable (console) Intel 80386, for MS Windows
MD5	a57d551e146e8d4e9d9440d88286f478
Ssdeep	6144:yvDkagYNgSUKEaun/NSIwzvKArT4mqIjoj:sDlNgbKEau1r2vrf4mqL
Yara	• shellcode (Matched shellcode byte patterns)