Using IOC (Indicators of Compromise) in Malware Forensics

GIAC (GREM) Gold Certification

Author: Hun-Ya Lock, hylock@gmail.com
Advisor: Adam Kliarsky
Accepted: February 21st 2013

Abstract

Currently there is a multitude of information available on malware analysis. Much of it describes the tools and techniques used in the analysis but not in the reporting of the results. However in the combat of malware, the reporting of the results is as important as the results itself. If the results can be reported in a consistent, well-structured manner that is easily understood by man and machine, then it becomes possible to automate some of the processes in the detection, prevention and reporting of malware infections. This paper would study the benefits of using OpenIOC framework as a common syntax to describe the results of malware analysis.

1. Introduction

1.1. Enterprise Malware Management

In the IT operations of an enterprise, malware forensics is often used to support the investigations of incidents. This could be due to end-user ignorance and carelessness, like drive-by-downloads as a result of careless web access, mistakes and oversights by administrators and their tools (Leydon, 2012) as well as Advanced Persistence Threat (APT) attacks. The objective of incident handling is to manage and control faults and disruptions to IT services. It includes both reactive and proactive measure. Table 1 lists the 6-Step process in incident handling (Murray, 2012) as describe by SANS.

Incident Handling Step	Type of Measure
1. Preparation	Proactive measure
2. Identification	Reactive measure
3. Containment	Reactive measure
4. Eradication	Reactive measure
5. Recovery	Reactive measure
6. Lessons Learned	Proactive measure

Table 1: SANS 6-steps process in incident handling (Murray, 2012)

Malware forensics falls under step 6. In the event of a new variant of malware, malware forensics can also take place in steps 3 to 5. Aquilina et. al. describes the objectives of malware investigations as follows:

Malware Forensics Investigation Objectives	
1.	Discover nature and purpose of program.
2.	Determine the infection mechanism.
3.	Determine how program interact with the host system.
4.	Determine how program interact with network.
5.	Determine how the attacker interact with the program.

Malware Forensics Investigation Objectives	
6.	Determine the profile and sophistication level of the attack.
7.	Determine the extent of infection and compromise of the host machine and beyond.

Table 2: Malware Forensics Investigation Objectives (Aquilina, Malin & Casey, 2010)

The purpose of the investigation is to characterize malware in terms of its attributes (static) and behaviors (dynamic) (Kirillov, 2012). This leads to 2 broad approaches towards malware forensics investigation: static and dynamic analysis. By performing static and dynamic analysis, objectives 3 and 4 would be met respectively. These describe the most basic characteristics of a malware. The rest of the objectives (1, 2 and 5 to 7) can be derived from these low-level attributes.

1.2. Incident Handling & Malware Forensics

Many enterprises are profit-drive environment and will strive to streamline and simplify its incident handling process. Hence, malware forensics investigation objectives in Table 2 can be further simplified to the following:

Simplified Malware Forensics Investigation Objectives (SMFIO)		Malware Forensics Investigation Objectives
1.	Detecting possible infection.	3 & 4
2.	Preventing further infection.	2
3.	Profiling infection	1, 5 & 6

Table 3: Simplified Malware Investigation Objectives

In the process of malware forensics investigations, the specimen needs to be analyzed in a forensically sound manner that ensures authenticity of the evidence with an analysis process that is reliable and repeatable. The investigation must also be well-supported with documentation (Casey 2011). OpenIOC (Indicators of Compromise) is an open source framework developed by Mandiant[1] for sharing threat intelligence (Sophisticated indicators for the modern threat landscape: an instruction to OpenIOC, 2011). It can be used to improve the reliability and repeatability of the malware forensics

[1]http://www.mandiant.com/news/release/mandiant-releases-openioc-standard-for-sharing-threat-intelligence/

investigation process by providing a standard documentation syntax. The OpenIOC framework can be used in the investigation report. As the framework utilizes XML(eXtensible Markup Langu age) to describe threat information, the derived OpenIOC indicators can be used as input to various security controls as part of the "Lessons Learned" phase of SANS 6-step process in incident handling (Table1). This is because XML has the advantage of being both machine and human readable.

2. Malware Forensics

2.1. Clean Room Setup

When investigating a malware specimen, it is important to do so in an isolated, "clean room" environment. The machines and network used in the analysis have to be isolated away from the production environment to prevent any possibility of malware outbreak. The behavior of the specimen should be analyzed in a cleanly installed machine that is not connected to external networks. By setting up a such a baseline environment, any changes made to the machines' state can be attributed to the malware.

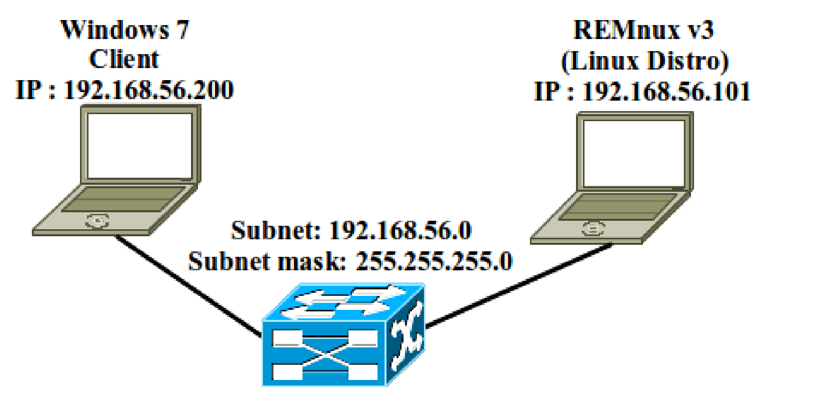

Diagram 1: Malware analysis logical setup

The setup used in this paper takes reference from SANS FOR610 (Reverse-Engineering Malware: Malware Analysis Tools and Techniques)[2] training. The diagram above shows the logical setup. In this setup, the malware will be executed in a Windows 7 SP1 machine. Various analysis tools are used to monitor and analyze its behavior. The

[2]http://computer-forensics.sans.org/training/course/reverse-engineering-malware-malware-analysis-tools-techniques#section_with_details_laptop_description

tools on Windows 7 and REMnux[3] machines are listed in Appendix 1.

The Windows 7 and REMnux machines are attached to the same subnet (192.168.56.0/24) to allow REMnux to monitor potential network traffic generated by the malicious specimen. In order to facilitate such a setup, all NIC (Network Interface Controller) cards on the machines and the switch need to be set to promiscuous mode. In addition, the Windows 7 client must be restored to its pristine state after each analysis or even during the analysis to ensure the reliability of the results obtained. The restoration of the Windows machine can be a time-consuming affair, so in practice this setup would be implemented in a virtual environment using VMware[4], VirtualBox[5] or even QEMU[6]. Besides using a single machine to host the setup, virtualization software has the advantage of supporting snapshots. Hence, the machine state at various stages of the investigations can be saved to facilitate rollbacks or the review of analysis results. For the paper, the visualization software used is VirtualBox version 4.2.4.

2.2. Static Analysis

In static analysis, the specimen's binary is examined without executing it. The tools commonly used for static analysis is documented in Appendix 1.

The first step in static analysis is file profiling which is done to obtain an initial assessment of the specimen's functionalities. Information such as strings, library dependencies, meta data and anti-virus signatures can be extracted from the executable file. The purpose of file profiling is reconnaissance, (Aquilina, Malin & Casey, 2010) in order to make an intelligent decision on the type of file and how to approach the analysis. It can also serve to fulfill step 1(detecting possible infection) of Simplified Malware Forensics Investigations Objectives (SMFIO)

The first step in file profiling is to obtain a cryptographic hash value of the specimen file, which is its digital fingerprint. This is easily obtained using Microsoft File Checksum Integrity Verifier (FCIV)[7]. Next, Linux `file` command would provide a

[3]http://zeltser.com/remnux/
[4]http://www.vmware.com/
[5]https://www.virtualbox.org/
[6]http://www.qemu.org/
[7]http://www.microsoft.com/en-us/download/details.aspx?id=11533

quick overview of the type of file (eg PE executable, DLL, kernel mode driver, documents, etc). The file's entropy is measured to determine the likelihood of it being packed and the export and import tables are viewed to get a sense of the functionalities of the specimen. There are many tools that can accomplish this, such as PEiD[8], xPELister[9] and PEBrowse[10].

2.3. Dynamic Analysis

In dynamic analysis, the behavior of the specimen is observed through its interaction with the host, as well as external system like web servers, IRC networks. There are a wide variety of tools available for dynamic analysis and the challenge is to decide on the most appropriate tools. Assuming that the malware specimen does not implement any anti-forensics measures, one of the most comprehensive tools to monitor behavior of a malware is SysInternal's Process Monitor[11]. A malware in its most basic form is essentially a Windows executable that, when run, would manifest as a Windows process, a child of Windows process or as a part of a process, in the case of code injection. This running process would interact with the host system in 5 main areas:

Main Areas of Interaction with Host System	
1.	Processes
2.	File system
3.	Registry
4.	Network activity
5.	API calls

Table 4: Main areas of interaction with host (Aquilina, Malin & Casey, 2010)

Process Monitor is able to monitor all of these interactions but often produces a very noisy set of data. In order to build to filters to remove unnecessary data from Process Monitor, RegShot[12] is used at the start of the investigations to sift through the noisy windows events and filter out potential malicious activities. RegShot is an open source Windows registry and file system comparison tool. Windows registry is a system-

[8]http://peid.has.it/
[9]http://tuts4you.com/request.php?426
[10]http://www.smidgeonsoft.prohosting.com/pebrowse-pro-file-viewer.html
[11]http://technet.microsoft.com/en-us/sysinternals/bb896645
[12]http://sourceforge.net/projects/regshot

defined database where applications and system components read and write configuration data. (Registry, 2012) Malware often uses the registry to find out the installed components and other capabilities of the target host as well as to store its own configuration. By comparing the registry before and after infection, evidence left by the malware can be used to build filters for Process Monitor.

Network activities also contain important information. If the malware attempts to "phone home", information of the remote attacker, as well as potential sources of malicious payload may be revealed. REMnux provides a variety of tools to emulate network services and wireshark[13] is available to monitor the network traffic.

2.4. Reporting

In digital forensics investigations, digital impression evidence and trace evidence are collected. Digital impression evidence are artifacts left in the physical memory, file system and registry as a result of the execution. Digital trace evidence are files and other artifacts that are typically introduced through the victim's online activity and are of a more temporary nature. (Aquilina, Malin & Casey, 2010)

When investigating malware infections, digital impression evidence are those that are associated with the infection and the self-preservation mechanisms and can be reproduced and observed in the "clean room" setup and compared with the victim's machine. These are classified as mandatory attributes. On the other hand, trace evidence depends on the environment that the malware is running in and the user's interaction with the infected system. Investigators may not always be able to reproduce them in the "clean room" setup and are classified as optional attributes.

When using OpenIOC framework to report the findings of the investigations, the mandatory and optional attributes can be expressed as *AND* and *OR* operators.

[13]http://www.wireshark.org/

3. OpenIOC Framework

3.1. Open IOC

Currently, there is no common language to describe the capabilities of malware. The hash value of the binary sample only identifies the specimen and little else. Furthermore, polymorphic and metamorphic codes (Paxson, 2011) result in multiple hash identities for the same class of malware. Hence there is a need to shift from identification of malware through its syntax (appearance of instructions) to its semantics (effect of instructions). OpenIOC is ideally suited for this purpose as the XML-based framework provides a flexible way of describing the complex semantics of a malware's behavior.

"Indicators of Compromise (IOCs) are forensic artifacts of an intrusion that can be identified on a host or network" (Sophisticated indicators for the modern threat landscape, 2012). It is similar to Mitre's CybOX's[14] (Cyber Observable eXpression) which uses XML schema for describing cyber observables. A cyber observable is a measurable event or stateful property in the cyber domain (Barnum, 2011). A standard manner of describing cyber observables, would allow for better communications amongst cyber security teams and potential interoperability of deployed tools and processes. According to Mandiant blog's[15], the CybOX team has included OpenIOC into its framework.

The motivations for developing OpenIOC, Mitre's CybOX and MAEC[16](Malware Attribute Enumeration and Characterization) are similar, which is to find a common language to describe malware infection and other cyber events. OpenIOC is focused on describing technical characteristics of a threat through an extensible XML schema. It has a comprehensive vocabulary for describing low level attributes which can be easily translated into machine-understandable formats. These can then be used as input to configure various IT security monitoring and detection tools like anti-virus, IDS (Intrusion Detection System), IPS (Intrusion Prevention System), firewalls, OS (Operating System) security controls and policies. Similarly, logs and other forms of

[14]http://cybox.mitre.org/
[15]https://blog.mandiant.com/archives/766
[16]http://maec.mitre.org/

outputs from these tools maybe translated into OpenIOC documents to be shared amongst other tools and systems. In this way, the intelligence gathered from an incident may be used to protect and prevent compromise of the entire environment. This would map to objectives 1 and 2 (detecting possible infection and preventing further infection) of SMFIO (Table 3). This method of malware investigation is illustrated by OpenIOC in the diagram below.

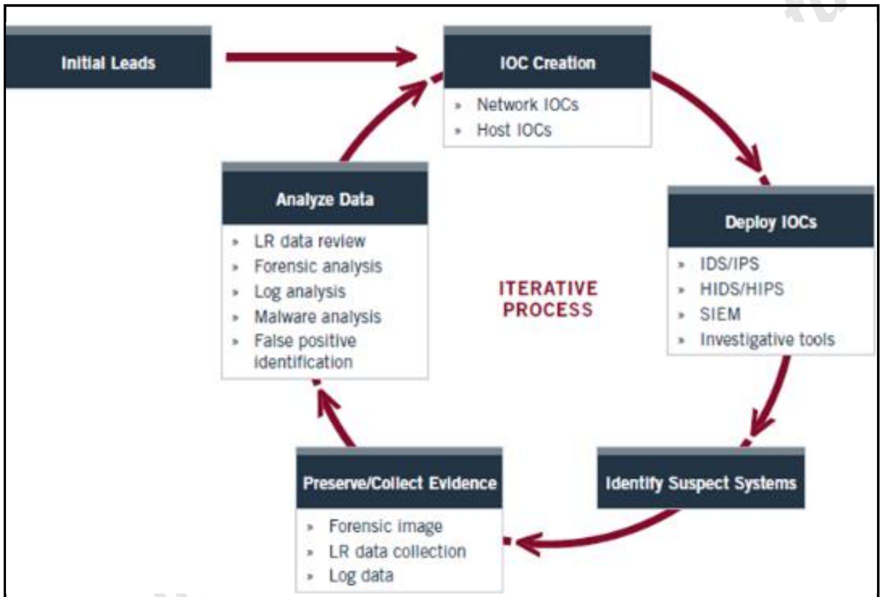

Diagram 2: Malware investigation lifecycle
taken from (Sophisticated indicators for the modern threat landscape, 2012)

Malware investigation is an iterative process. It begins with developing OpenIOC indicators based on the low-level attributes of the malware's interaction with the host and network. The OpenIOC can then be used as inputs to the enterprise monitoring tools and used for further analysis. The table below applies OpenIOC framework to an enterprise incident handling process, more specifically the proposed SMFIO (Simplified Malware Forensics Investigation Objective).

Simplified Objective	SANS Incident Handling Step	Explanation
-	Step 1: Preparation	This step takes place prior to an incident and does not take OpenIOC into account.
(1) Detecting possible infection.	Step 2: Identification	OpenIOC is used to describe the malware. It could be based on its file profile and network traffic signature.
(2) Preventing further infection.	Step 3: Containment Step 4: Eradication	OpenIOC is used to document the changes made to the infected host's file system and registry configurations; kernel and other program hooks; network protocols and ports. With this information, network and the host IPS could be configured for the purpose of containment and eradication.
-	Step 5: Correction	In the correction phases the IT system is placed back into production mode with all the business processes in place. This is beyond the scope of OpenIOC.
(3) Profiling infection.	Step 6: Lessons Learned	These consist of OpenIOCs that could describe the profile of the attacks in order to determine if it is a targeted attack. More robust containment and eradication steps would be required to prevent or at least reduce the damage from such attacks.

Table 5:OpenIOC for Incident Handling

3.2. Using IOC

A sample OpenIOC is shown below. It documents the low-level attributes that are observed when a host is infected by the Zeus virus. A full listing is presented in Appendix 3. The verbose nature of XML makes the IOC self-explanatory.

```xml
<?xml version="1.0" encoding="us-ascii"?>
<ioc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" id="6d2a1b03-b216-4cd8-9a9e-8827af6ebf93" last-modified="2011-10-28T19:28:20" xmlns="http://schemas.mandiant.com/2010/ioc">
  <short_description>Zeus</short_description>
  <description>Finds Zeus variants, twexts, sdra64, ntos</description>
  <keywords />
  <authored_by>Mandiant</authored_by>
  <authored_date>0001-01-01T00:00:00</authored_date>
  <links />
  <definition>
    <Indicator operator="OR" id="9c8df971-32a8-4ede-8a3a-c5cb2c1439c6">
      <Indicator operator="AND" id="0781258f-6960-4da5-97a0-ec35fb403cac">
        <IndicatorItem id="50455b63-35bf-4efa-9f06-aeba2980f80a" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/name" type="mir" />
          <Content type="string">winlogon.exe</Content>
        </IndicatorItem>
        <!---SNIP--->
      </Indicator>
      <Indicator operator="AND" id="9f7a5703-8a26-45cf-b801-1c13f0f15d40">
        <IndicatorItem id="cf77d82f-0ac9-4c81-af0b-d634f71525b5" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Type" type="mir" />
          <Content type="string">Mutant</Content>
        </IndicatorItem>
        <!---SNIP--->
        </Indicator>
      </Indicator>
    </Indicator>
  </definition>
</ioc>
```

AND Operator

AND Operator

OR Operator

Diagram 3: OpenIOC sample

(taken from http://openioc.org/iocs/6d2a1b03-b216-4cd8-9a9e-8827af6ebf93.ioc)

The malware's low-level behavior attribute is documented using **<IndicatorItem>** tag. Multiple **<IndicatorItem>** tags may be grouped together using **<Indicator>** tag. They may be grouped according to logical *AND* or *OR* operators as seen in the diagram above. A rule of thumb would be to group attributes associated with a behavior into using **<Indicator>** tag with *AND* attribute. Groups of behavior can then be associated with the *OR* operator. This set of indicators only describe the processes and handles that are created with the Zeus infection and can only achieve step 1 (detecting possible infection) of SMFIO.

The next section would analyze a malware specimen using SMFIO and use OpenIOC to document the results. It would then explore how to use the resulting OpenIOC in the management of the IT system in an enterprise.

4. Case Study

4.1. Background

A suspicious file would have one or more of the following characteristics, an unknown origin, located in system folders or unusual or hidden locations in the system, has unusual or misspelt names and contains obfuscated code. Suspicious files are often investigated to determine its damage potential and derive prevention mechanisms against it. This section examines a malware (hash value: aada169a1cbd822e1402991e6a9c9238) that was caught by a private honeypot. To facilitate the discussion, a random name of "ada.exe" was given to the specimen. The "clean room" set up discussed in section 2.1 was used.

4.2. File Profiling

Microsoft File Checksum Integrity Verifier was used to obtain the MD5 hash of the specimen. Linux `file` command, xPELister and PEBrowse were used in the initial assessment of the file type

From `file` command and PEiD, it was quite clear that this was a packed file with an high entropy level of 7.98. PECompact was the packer used.

```
[redacted]$ file ada.exe
ada.exe: PE32 executable (GUI) Intel 80386, for MS Windows, PECompact2 compressed
```

Diagram 4: Output from Linux file command

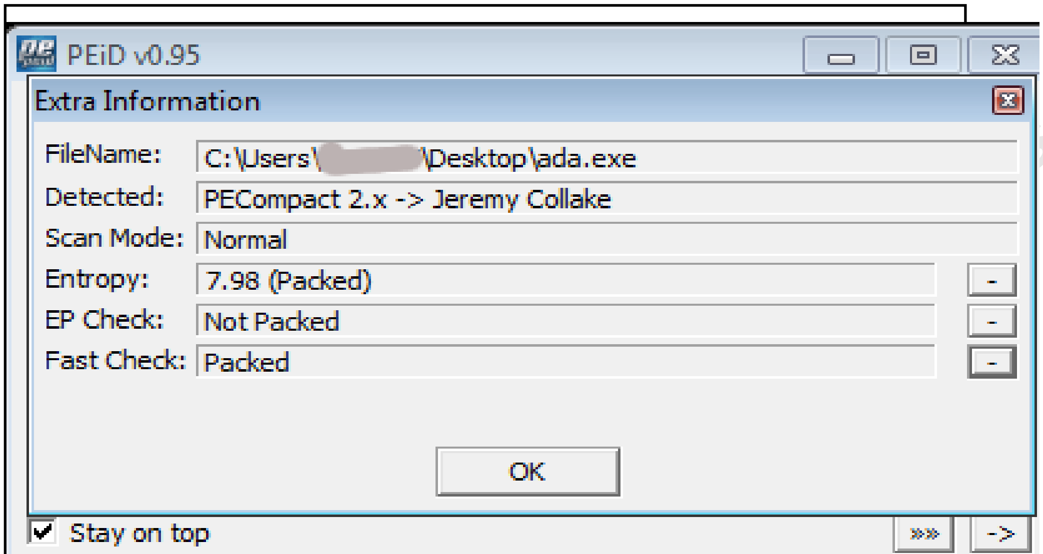

Diagram 5: Output from PEiD

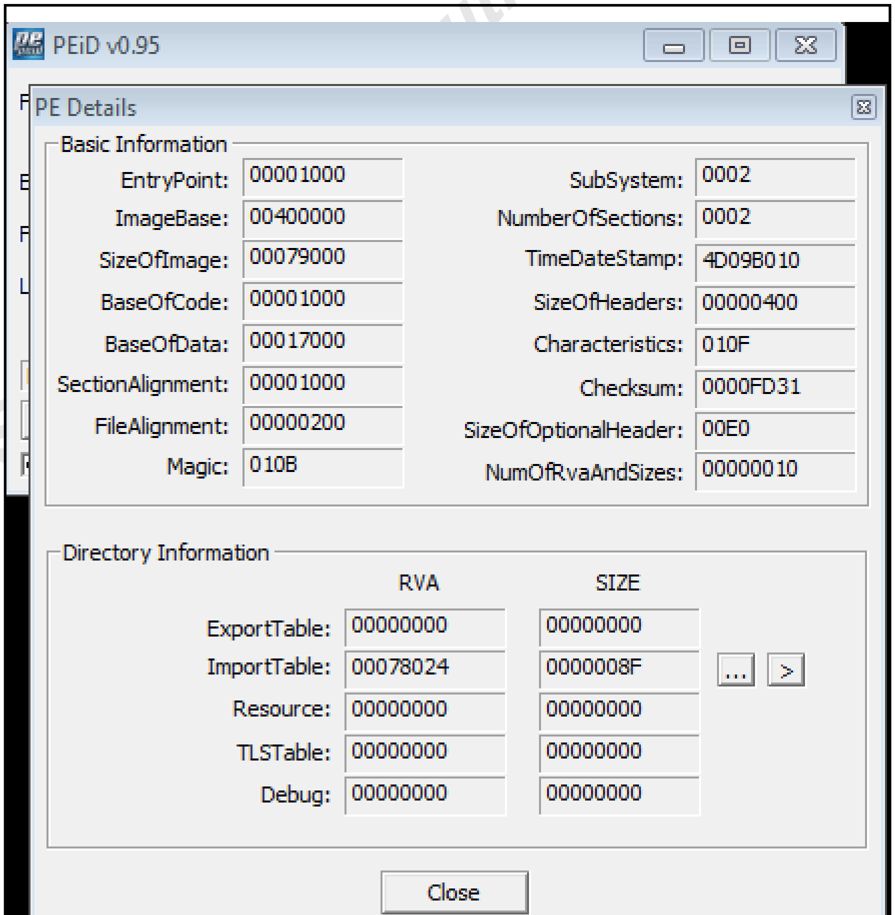

Diagram 6: PE header information of the obfuscated file

Given that this is a packed file, the information from its PE header such as section information, entry point and other file characteristics will be changed for the de-obfuscated malicious executable. Hence, the information gathered so far, was useful in identifying the infection, which is the obfuscated payload. Unfortunately with unlimited iterations of obfuscation, it would not be feasible to make use of this information to configure anti-virus scanners and IDS systems.

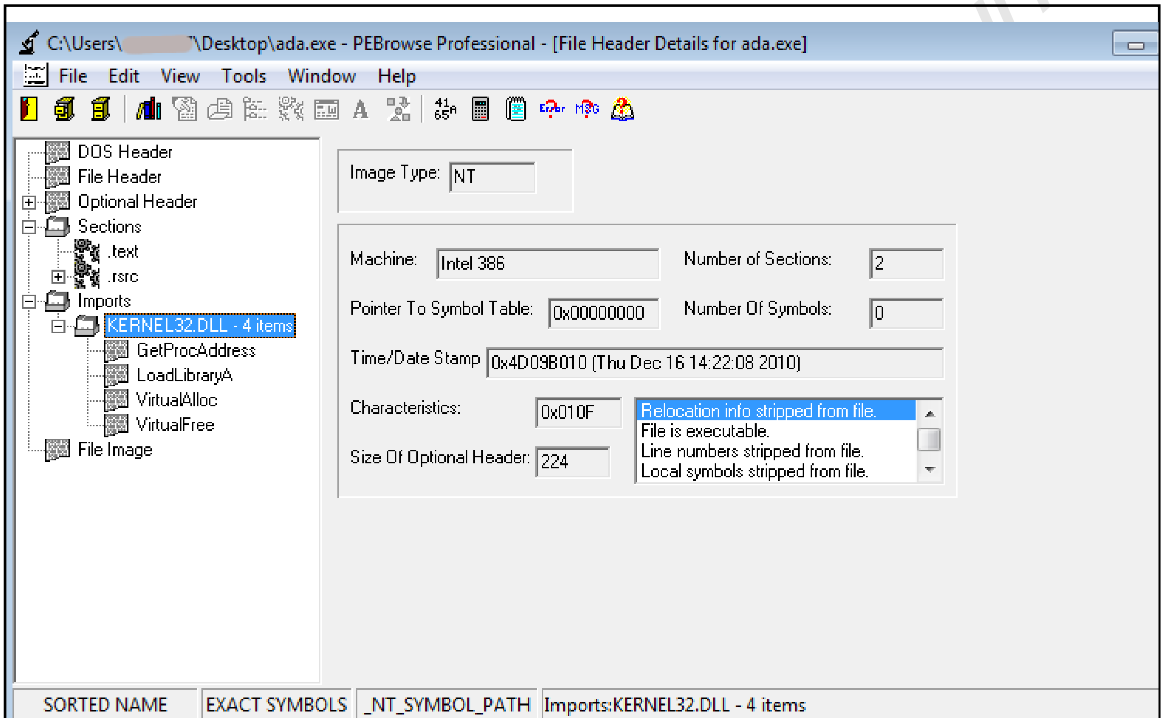

Diagram 7: Output from PEBrowse

PEBrowse interpreted the PE header and showed that it was a 32bit Windows executable. The import table only contained 4 functions from Kernel32.dll: GetProcAddress, LoadLibraryA, VirtualAlloc and VirtualFree, a characteristics of packed files. Without further information, it would be difficult to determine the damage potential of the file.

<u>SMFIO 1: Detecting possible infection</u>

The information obtained so far can be used for objective 1 (detecting possible infection) of SMFIO. The propose OpenIOC indicators are listed below.

```
AND
├─ File MD5 is aada169a1cbd822e1402991e6a9c9238
├─ File PE Type contains PE32
├─ File DetectedCharacteristics contains PECompact2
└─ OR
   └─ File Compile Time contains Dec 16 2010 14:22:08
```

Diagram 8: OpenIOC of file profile.

The approach taken is to list attributes associated with digital impression evidence using the *AND* operator and put trace evidence under the *OR* operator. Although, all the attributes listed in diagram 8 can be observed in an infection, file compile time is subjected to time zone configuration on the development and infected machines and requires careful handling. With other more reliable identifiers, file compile time is put in as an optional attribute.

4.3. Dynamic Analysis

Obtaining Snapshots of Changes using Regshot

Initially, RegShot was used to compare the registry and file system before and after infection. The most obvious indication of malware infection was the addition of a file named “serivces.exe” in “C:\Windows\System32” directory and as well as the deletion of the original malicious code. The file name “serivces” stood out as it was a misspelling of the word “services” which is a system component in Windows.

```
---------------------------------
Files added:20
---------------------------------
C:\ProgramData\Microsoft\Search\Data\Applications\Windows\Projects\SystemIndex\Indexer\CiFiles\00010002.ci
C:\ProgramData\Microsoft\Search\Data\Applications\Windows\Projects\SystemIndex\Indexer\CiFiles\00010002.dir
C:\ProgramData\Microsoft\Search\Data\Applications\Windows\Projects\SystemIndex\Indexer\CiFiles\00010002.wid
C:\ProgramData\Microsoft\Windows\WER\ReportQueue
\NonCritical_ada.exe_277f138b4479892cd1686a9b3a4c1293d2a154b8_cab_0c62fd2f\appcompat.txt
C:\ProgramData\Microsoft\Windows\WER\ReportQueue
\NonCritical_ada.exe_277f138b4479892cd1686a9b3a4c1293d2a154b8_cab_0c62fd2f\Report.wer
C:\Users\All Users\Microsoft\Search\Data\Applications\Windows\Projects\SystemIndex\Indexer\CiFiles\00010002.ci
C:\Users\All Users\Microsoft\Search\Data\Applications\Windows\Projects\SystemIndex\Indexer\CiFiles\00010002.dir
C:\Users\All Users\Microsoft\Search\Data\Applications\Windows\Projects\SystemIndex\Indexer\CiFiles\00010002.wid
C:\Users\All Users\Microsoft\Windows\WER\ReportQueue
\NonCritical_ada.exe_277f138b4479892cd1686a9b3a4c1293d2a154b8_cab_0c62fd2f\appcompat.txt
C:\Users\All Users\Microsoft\Windows\WER\ReportQueue
\NonCritical_ada.exe_277f138b4479892cd1686a9b3a4c1293d2a154b8_cab_0c62fd2f\Report.wer
C:\Users\remwin7\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations
\7e4dca80246863e3.automaticDestinations-ms
C:\Users\remwin7\AppData\Roaming\Microsoft\Windows\Recent\Documents.lnk
C:\Users\remwin7\AppData\Roaming\Microsoft\Windows\Recent\Network and Internet.lnk
C:\Users\remwin7\AppData\Roaming\Microsoft\Windows\Recent\regshot1.hiv.lnk
C:\Users\remwin7\Documents\adaprocmon-1.PML
C:\Users\remwin7\Documents\adaprocmon.PML
C:\Users\remwin7\Documents\ada_regshot1.hiv
C:\Windows\System32\LogFiles\Scm\baf1cbfd-e527-4a35-ac77-7500b81c5955
C:\Windows\System32\Tasks\{6C16B028-2612-42F7-A06D-0BEA239E95DD}
C:\Windows\System32\serivces.exe
```

Diagram 9: RegShot output showing addition of serivces.exe

```
---------------------------------
Files deleted:2
---------------------------------
C:\Users\remwin7\Desktop\ada.exe
C:\Windows\System32\LogFiles\Scm\2ee9a791-889f-4c9e-9dce-20fd814e27a5
```

Diagram 10: Regshot output showing malware deleted

Further analysis of RegShot's output showed that "services.exe" was installed as a Windows service with a seemingly legitimate service name "Plug and Play Manager". In reality, the Windows service that supported Plug and Play was called "PlugPlay".

```
---------------------------------
Keys added:131
---------------------------------
HKLM\SOFTWARE\Microsoft\Tracing\serivces_RASAPI32
    ....
HKLM\SYSTEM\CurrentControlSet\services\PlugPlayCM
HKLM\SYSTEM\CurrentControlSet\services\PROCMON23
```

Diagram 11: Regshot output showing suspicious keys.

```
----------------------------------
Values added:395
----------------------------------
HKLM\SOFTWARE\Microsoft\Tracing\serivces_RASAPI32\EnableFileTracing: 0x00000000
HKLM\SOFTWARE\Microsoft\Tracing\serivces_RASAPI32\EnableConsoleTracing: 0x00000000
HKLM\SOFTWARE\Microsoft\Tracing\serivces_RASAPI32\FileTracingMask: 0xFFFF0000
     ....
HKLM\SYSTEM\CurrentControlSet\services\PlugPlayCM\Type: 0x00000110
HKLM\SYSTEM\CurrentControlSet\services\PlugPlayCM\Start: 0x00000002
HKLM\SYSTEM\CurrentControlSet\services\PlugPlayCM\ErrorControl: 0x00000000
HKLM\SYSTEM\CurrentControlSet\services\PlugPlayCM\ImagePath: ""C:\Windows\system32\serivces.exe""
HKLM\SYSTEM\CurrentControlSet\services\PlugPlayCM\DisplayName: "Plug and Play Manager"
HKLM\SYSTEM\CurrentControlSet\services\PlugPlayCM\ObjectName: "LocalSystem"
HKLM\SYSTEM\CurrentControlSet\services\PlugPlayCM\FailureActions: 0A 00 00 00 00 00 00 00 00 00 00 00 01 00 0
00 14 00 00 00 01 00 00 00 B8 0B 00 00
HKLM\SYSTEM\CurrentControlSet\services\PlugPlayCM\Description: "Plug and Play Manager Support Service. If thi
service is stopped, protected content might not be down loaded to the device."
```

Diagram 12: Regshot output showing suspicious values added

The start key with value of 0x2 indicated that this service would start automatically. The type key with value of 0x110 indicated that this was a Win32 program that ran in a process by itself (JSI Tip 0324, 1997). This was the specimen's self-preservation mechanism.

Monitoring Interaction with Host System using Process Monitor & CaptureBat

After reverting back to its pristine stage, the system is reinfected and monitored by Process Monitor. Using the process names “ada.exe” and “serivces.exe” as a filter, here is the sequence of significant events that occurred:

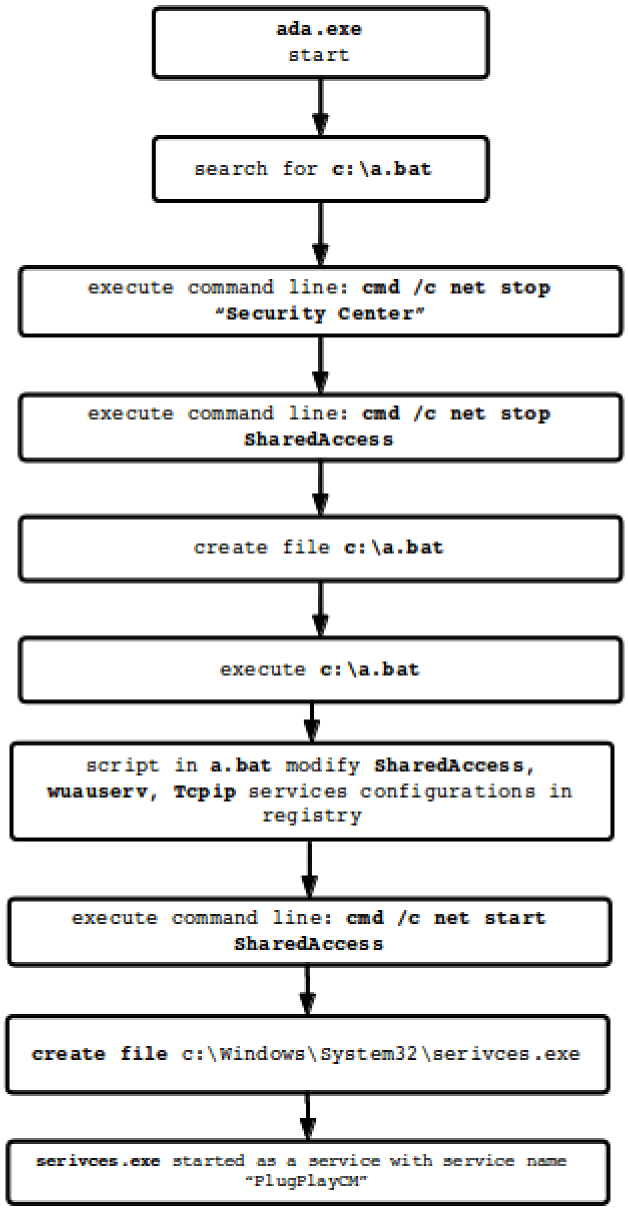

Diagram 13: Process flow of specimen

The malicious specimen stopped Windows Security Center[17] which then stopped alerts and notifications from several Windows security components including the firewall, anti-virus, Windows Update, Internet options. As a result, Windows SharedAccess service that controlled Internet-connection sharing[18], which included firewall configuration, was stopped. The file "a.bat" contained scripts that modified

[17] http://windows.microsoft.com/is-IS/windows-vista/Using-Windows-Security-Center
[18] http://technet.microsoft.com/en-us/library/cc766190%28v=ws.10%29.aspx

registry settings to disable firewall, Windows Automatic Update, Windows Security Centre services. It also contained entries that modified the TCP/IP parameters. As a result, when Windows SharedAccess was started, these services were no longer available. After modifying the registry settings "a.bat" was deleted. A copy of "a.bat" was recovered using CaptureBat[19] and documented in Appendix 4.

Finally, "serivces.exe" was created and was installed as a service with a service name "PlugPlayCM". After the service started, "ada.exe" was deleted.

Monitoring Interaction with Network using REMnux

REMnux was used to draw network traffic out from the malicious specimen. The first step was to use wireshark[20] to monitor network traffic from the malicious specimen, in order to determine the type of network services that it was seeking. The specimen initially sent TCP SYN requests to ip address 60.10.179.100, connecting to a range of ports which included 8684 – 8689, 9051, 137(WINS registry), 12032, 8680 – 8689, 1709 and 343. A snapshot of the SYN request is presented below.

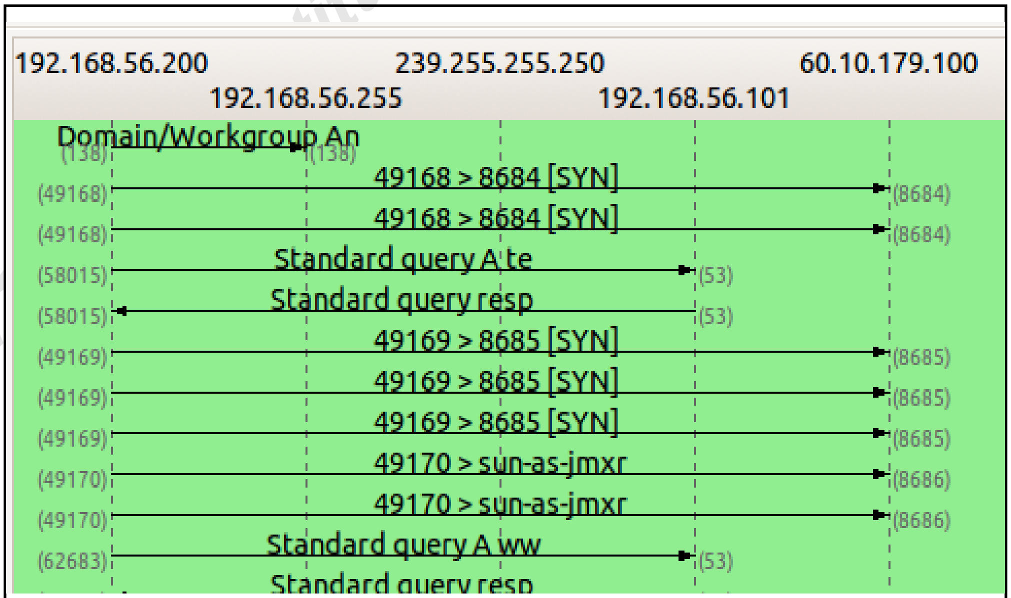

Diagram 14: Wireshark output of initial TCP SYN request

Of course, REMnux acting as the gateway to nowhere, was not able to connect to

[19]http://www.nz-honeynet.org/capture-standalone.html
[20]http://www.wireshark.org

IP address 60.10.179.100. A check with Robtex[21] reverse DNS service website, revealed that this ip address was blacklisted.

Next the specimen, made domain name resolution requests to ringc.strangled.net, checkip.dyndns.org and www.ip138.com. fakedns[22] was used to resolved all domain names requested by the malicious specimen to the REMnux machine. With the domain names resolved to REMnux machine's ip address, the specimen then sent HTTP Get requests to checkip.dyndns.org and www.ip138.com. Both sites were visited anonymously using the TOR[23] browser. It was found that they would both return the ip address of the requesting client so it can be assumed that the specimen was attempting to acquire the ip address of the host it had infected. A screenshot of www.ip138.com is displayed below:

Diagram 15: Screenshot of web site www.ip138.com

The most significant network requests were TCP SYN requests to port 8684. netcat[24] was used to start a port 8684 in listening state in REMnux. With this simple setup, the network requests to port 8684 was captured and examined. The output from netcat is shown below:

[21] http://www.robtex.com
[22] http://code.activestate.com/recipes/491264-mini-fake-dns-server/
[23] https://www.torproject.org/projects/torbrowser.html
[24] http://netcat.sourceforge.net/

```
remnux@remnux:~$ sudo nc -l -p 8684
NICK USA|WN7}|SP1|1|41200654
USER SP1-082 * 0 :WIN7_REM
```

Diagram 16: netcat output

It turned to be an IRC request. The specimen used details from the infected host to generate the user and nick login details. The host operating system was Windows 7 Service Pack 1 with system locale, keyboard and location set to USA. The Windows login username was Win7_REM. This might explain the phrases “USA”, “SP1” and WIN7_REM” in the IRC connection request.

To probe further, ircd service in REMnux was configured to listen to a range of ports 8684-8689, which included 8684. After sometime, it was observed in wireshark that the specimen joined IRC channel “#blue3”. The wireshark output is shown below:

```
:remnux. 375 USA|WN7}|SP1|1|54507277 :remnux. message of the day
:remnux. 372 USA|WN7}|SP1|1|54507277 :-
:remnux. 376 USA|WN7}|SP1|1|54507277 :End of message of the day.
:remnux. 251 USA|WN7}|SP1|1|54507277 :There are 1 users and 0 invisible on 1 server
:remnux. 255 USA|WN7}|SP1|1|54507277 :I have 1 clients and 0 servers
JOIN #blue3
:USA|WN7}|SP1|1|54507277!SP1-526@0::ffff:192.168.56.200 JOIN :#blue3
:remnux. 353 USA|WN7}|SP1|1|54507277 = #blue3 :@USA|WN7}|SP1|1|54507277
:remnux. 366 USA|WN7}|SP1|1|54507277 #blue3 :End of /NAMES list.
JOIN #blue3
JOIN #blue3
PING :remnux.
PONG remnux.
PING :remnux.
PONG remnux.
PING :remnux.
PONG remnux.
PING :remnux.
```

Diagram 17: Wireshark output for REMnux with ip of 60.10.169.100

From monitoring the behavior of the specimen, the following OpenIOC indicators are proposed.

SMFIO 1: Detecting possible infection

The changes to the host file system and registry are mandatory attributes (*AND* operator).

```
AND
 ├ File Full Path contains c:\Windows\System32
 ├ File Name contains serives.exe
 └ File MD5 is aada169a1cbd822e1402991e6a9c9238
```

Diagram 18: OpenIOC from dynamic analysis 1

SMFIO 2: Preventing further infection

These OpenIOC indicators describe the changes made to the host and the network traffic generated after an infection. These indicators suggest how the host system could be hardened in order to prevent further and future infections. For example, for this specimen a host firewall could be configured to prevent outgoing network traffic to IP address 60.10.179.100, connections to the port8680 to 8689 are put in as optional (*OR* operator) as they are dependent on the infected host's network settings.

```
OR
 ├ Network String General contains blue3
 ├ Network String URI is http://checkipdyndns.org
 ├ Network String URI is http://www.ip138.com
 └ AND
    ├ Port Remote IP contains 60.10.179.100
    └ Port remotePort contains 8680 to 8689
```

Diagram 19: OpenIOC from dynamic analysis 2

On the other hand, the OpenIOC indicators for the changes to the registry are mandatory (*AND* operator). This is because these indicators describe the infection and self-preservation mechanisms.

```
AND
  Registry KeyPath contains HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess
  Registry ValueName contains Start
  Registry Value contains 2
AND
  Registry KeyPath contains HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\
                                                   Parameters\FirewallPolicy\StandardProfile
  Registry ValueName contains EnableFirewall
  Registry Value contains 0
AND
  Registry Value contains Start
  Registry Value contains 4
  AND
    Registry KeyPath contains HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\wscsvc
    Registry KeyPath contains HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\wuauserv
AND
  Registry Path contains HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
  AND
    Registry ValueName contains MaxFreeTcbs
    Registry Value contains 0x7d0
  AND
    Registry ValueName contains MaxHashTableSize
    Registry Value contains 0x800
  AND
    Registry ValueName contains TcpTimedWaitDelay
    Registry Value contains 0x1e
  AND
    Registry ValueName contains MaxUserPort
    Registry Value contains 0xf618
```

Diagram 20: OpenIOC from dynamic analysis 3

SMFIO 3: Profiling Infection

Outbound network traffic sometimes provides identity of the attacker that can be used in developing the profile of the attack. In this case, remote IRC server that the specimen tried to connect to may provide a link to the attacker. As they are trace evidence, the *OR* operator is used.

```
OR
  Network DNS contains ringc.strangled.net
  Process remoteIP contains 60.10.179.100
```

Diagram 21: OpenIOC from dynamic analysis 4

4.4. Static Analysis

From the dynamic analysis, the infection and self-preservation mechanism was found. In addition, it was found that the specimen was most likely an IRC bot but without connecting the IRC bot to its C&C (Command and Control), it was difficult to determine it functionalities. Hence, static analysis was carried out to probe further into the specimen.

Gathering Strings

The first step into static analysis is to gather a list of strings from the binary executable. The Linux `strings` command was used. From file profiling done earlier, it was known that this was a packed specimen and the `strings` command would not produce many strings of interest. The results correspond to that from the file profiling stage.

```
[redacted]:~/[redacted]$ strings ada.exe
PECompact2
_r]k
RB/)M4
U2.3
 ...
kernel32.dll
LoadLibraryA
GetProcAddress
VirtualAlloc
VirtualFree
VWSU
kernl32.d
irtualFe
```

Diagram 22: List of strings from specimen

Debugging with OllyDbg

The specimen had various anti-forensics strategies which had to be overcome before the malicious code could be analyzed in OllyDbg[25]. As discovered earlier, the specimen was packed using PECompact version 2[26]. This was a common packer which would compress and in doing so obfuscate the code as well as the import table. When the executable was run, the decompression stub was loaded and it would restore the image of malicious code to an executable state that was loaded only onto the memory without writing to disk. The process of loading the de-obfuscated code in OllyDbg with the correct OEP (Original Entry Point) is well documented (Collake, 2005) and presented in Appendix 5.

The OEP of the malicious code is 0x415F64. Before running the specimen in OllyDbg, IDA[27] was used to generate the specimen's call flow.

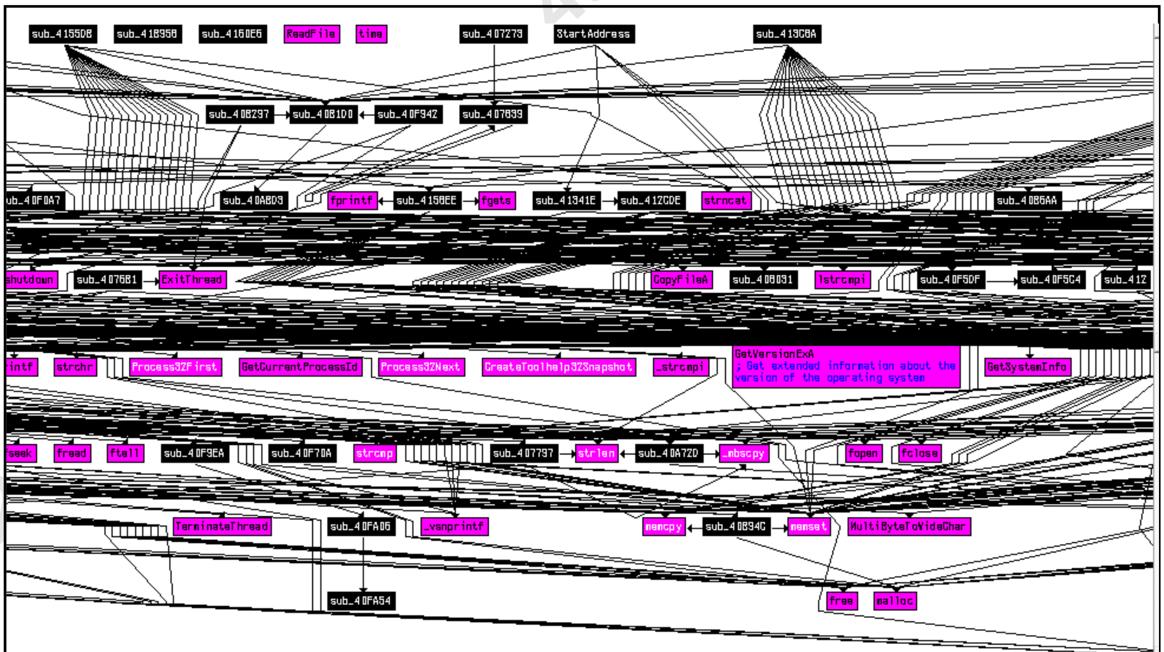

Diagram 23: Partial call flow of specimen

As can be seen from the call flow above, the code has a complex structure. To reverse it completely back to its original state might not be feasible for an enterprise IT department. However if the scope of the analysis is restricted to SMFIO, the

[25]http://www.ollydbg.de/
[26]http://bitsum.com/pecompact.php
[27]http://www.hex-rays.com/products/ida/index.shtml

investigation would be more feasible.

The initial portion of the code, when analyzed in OllyDbg, had numerous segments that checked if it was being monitored. On detection of a debugger, it would terminate prematurely. The flow charts of the anti-forensics are presented below.

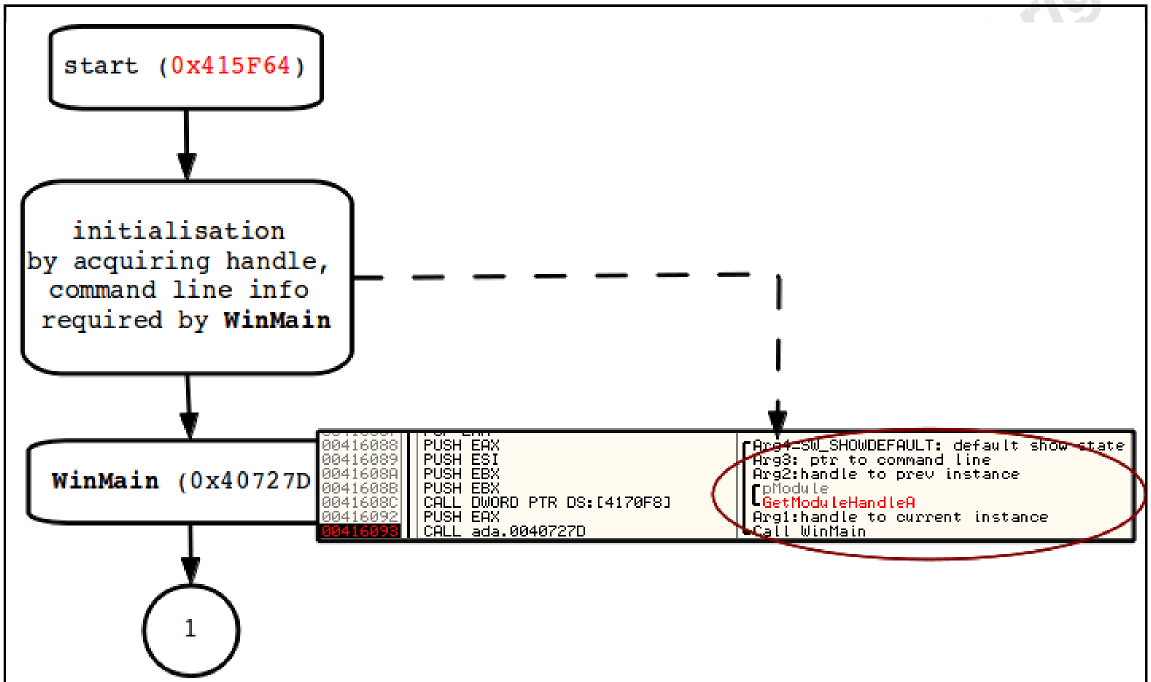

Diagram 24: Flow chart 1

Flow chart 1 shows the initialization phase of the specimen. At the start it gathered a handle to itself and the command line parameters and passed them to WinMain[28] function.

[28]http://msdn.microsoft.com/en-us/library/windows/desktop/ms633559%28v=vs.85%29.aspx

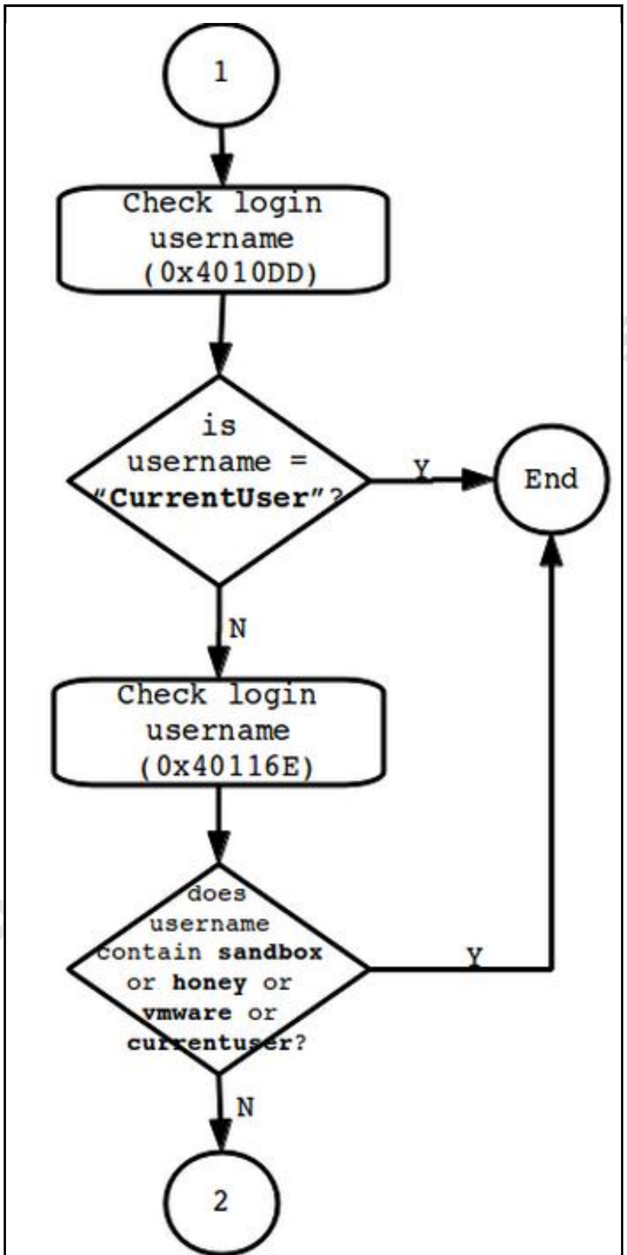

Diagram 25: Flow chart 2

In WinMain, the specimen launched into several anti-forensics strategies. Firstly, it checked if the username of the machine was "CurrentUser" or contained the strings "sandbox", "honey", "vmware" or "currentuser". These would be easily defeated by ensuring that the username of the machine did not contain these strings.

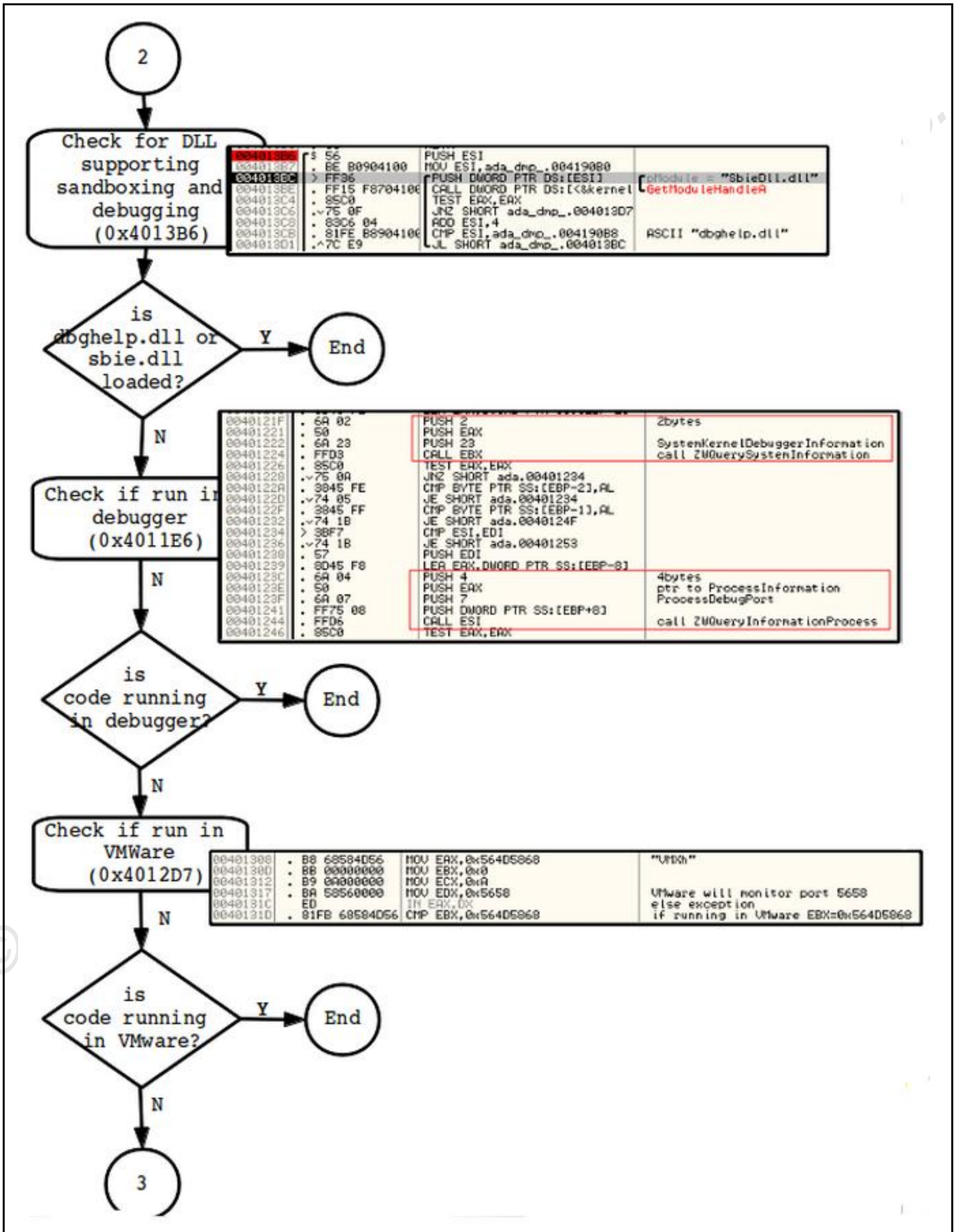

Diagram 26: Flow Chart 3

Next at 0x4013B6, it checked if “dbghelp.dll” and “sbie.dll” DLL (Dynamic Link

Library) were loaded. This was done through GetModuleHandleA[29] at address 0x4013BE. The presence of "sbie.dll" would indicate that Sandboxie[30] was running and it could potentially limit the specimen's malicious functions. "dbghelp.dll" was used by Microsoft DbgHelp library[31] and its presence would alert the specimen that it was debugged. If the specimen detected that these DLLs were loaded, it would end its process. Otherwise, it would continue on to function 0x4011E6 where it further checked for the presence of debugger through the use of ZwQuerySystemInformation[32] and ZwQueryInformationProcess[33].

Next at 0x401388, the specimen tested to see if it was ran in a VMware virtual machine. This was done through detecting the presence of the port 5658 (Liston, 2006). Liston and Skoudis had describe in their research that VMWare monitors port 5658 when EAX was set to the magic number 0x564D5868 ("VMXh"). If the specimen was ran in VMware, EBX would be set to 0x564D5868 after the command "IN EAX, EDX", else there would be an exception. In order to bypass this anti-forensics measure, "IN EAX, EDX" is modified to "NOP" and EBX is setup to the expected value.

After a series of anti-debugging steps, the specimen then installed itself as a Windows Service. In 0x407331 of WinMain, the specimen setup up the SERVICE_TABLE_ENTRY[34] which would contain the address to ServiceMain of a service. In this case, ServiceMain of the malicious service was at 0x40A8D3. At 0x40A970, CreateServiceA was called to install the service with service name "PlugPlayCM" and display name "Plug and Play Manger" which were seemingly legitimate service name. After StartServiceA was called at 0x40AA4C, it would call StartServiceCtrlDispatcherA[35] at 0x4073FF to connect the main thread of the malicious service to service control manager so that it would be the service control dispatcher thread for the calling process. When that specimen was run in a debugger, StartServiceCtrlDispatcherA would fail with error code of

[29]http://msdn.microsoft.com/en-us/library/windows/desktop/ms683199%28v=vs.85%29.aspx
[30]http://www.sandboxie.com/
[31]http://msdn.microsoft.com/en-us/library/windows/desktop/ms679294%28v=vs.85%29.aspx
[32]http://msdn.microsoft.com/en-us/library/windows/desktop/ms725506%28v=vs.85%29.aspx
[33]http://msdn.microsoft.com/en-us/library/windows/desktop/ms687420%28v=vs.85%29.aspx
[34]http://msdn.microsoft.com/en-us/library/windows/desktop/ms686001%28v=vs.85%29.aspx
[35]http://msdn.microsoft.com/en-us/library/windows/desktop/ms686324%28v=vs.85%29.aspx

ERROR_FAILED_SERVICE_CONTROLLER_CONNECT. This was because OllyDbg ran the specimen as a console program rather than a service. After calling StartServiceCrtlDispatcherA, the process would end. If the PlugPlayCM service had been started properly, the specimen would continue its activities in the service.

In order to continue debugging the specimen, it was executed without the use of a debugger. All the anti-forensics techniques used by the specimen would not be effective as it was not debugged and not ran in VMware. In this case, PlugPlayCM service could be started. At this point, OllyDbg could then be launched and attached to the running process “serivces.exe”.

In OllyDbg, the process would pause at ntdll.DbgBreakPoint. By reviewing the Threads window in OllyDbg, the following was observed.

Threads

Ident	Entry	Data block	Last error	Status	Priority	User time	System time
00000644	00000000	7FFDE000	ERROR_SUCCESS (00000000)	Active	32 + 0	0.0000 s	0.1201 s
00000730	0040740C	7FFD8000	ERROR_SUCCESS (00000000)	Active	32 + 0	0.0100 s	0.0100 s
0000072C	0040AA6D	7FFD9000	ERROR_SUCCESS (00000000)	Active	32 + 0	0.0000 s	0.0000 s
000007B4	00410246	7FFD7000	ERROR_SUCCESS (00000000)	Active	32 + 0	0.0100 s	0.0000 s
000007C4	73AF32FB	7FFD5000	ERROR_SUCCESS (00000000)	Active	32 + 0	0.0000 s	0.0000 s
000006FC	757C7587	7FFDA000	ERROR_SUCCESS (00000000)	Active	32 + 0	0.0000 s	0.0000 s
00000674	7730FD0F	7FFDD000	ERROR_SUCCESS (00000000)	Active	32 + 0	0.0000 s	0.0000 s
000006F8	773103E7	7FFDB000	ERROR_SUCCESS (00000000)	Active	32 + 0	0.0000 s	0.0000 s
00000658	7737F125	7FFDC000	ERROR_SUCCESS (00000000)	Active	32 + 0	0.0000 s	0.0000 s

Diagram 27: Threads spawned by serivces.exe

“serivces.exe” spawned 3 threads with starting addresses at 0x40740C, 0x40AA6D and 0x410246.

<u>Thread 0x4AA6D</u>

Thread 0x40AA6D was created in ServiceMain (0x40A8D3) at address 0x40A94E. The diagram below showed the code snippet. The create flag (dwCreateFlags) was set to 0 this would cause the thread to run immediately after creation. The register ESI is set to 0 previously at 0x40A8D8 with the command “XOR. ESI. ESI”.

```
.text:0040A944                 push    eax             ; lpThreadId
.text:0040A945                 push    esi             ; dwCreateFlags = 0
.text:0040A946                 push    esi             ; lpParameter = 0
.text:0040A947                 push    offset sub_40AA6D ; lpStartAddress
.text:0040A94C                 push    esi             ; dwStackSize = 0
.text:0040A94D                 push    esi             ; lpThreadAttributes = 0
.text:0040A94E                 call    CreateThread
```

Diagram 28: Start thread 0x40AA6D

The main function of thread 0x40AA6D was to create and start thread 0x40740C. Similarly to the ServiceMain, the create flags (dwCreateFlags) is set to 0 at 0x40AA74 and so the thread 0x40740C would run immediately after creation.

```
.text:0040AA6D sub_40AA6D      proc near               ; DATA XREF: .text:0
.text:0040AA6D
.text:0040AA6D ThreadId        = dword ptr -4
.text:0040AA6D
.text:0040AA6D                 push    ecx
.text:0040AA6E                 push    esi
.text:0040AA6F                 push    edi
.text:0040AA70                 lea     eax, [esp+0Ch+ThreadId]
.text:0040AA74                 xor     edi, edi
.text:0040AA76                 push    eax             ; lpThreadId
.text:0040AA77                 push    edi             ; dwCreationFlags
.text:0040AA78                 push    edi             ; lpParameter
.text:0040AA79                 push    offset sub_40740C ; lpStartAddress
.text:0040AA7E                 push    edi             ; dwStackSize
.text:0040AA7F                 push    edi             ; lpThreadAttributes
.text:0040AA80                 call    CreateThread
```

Diagram 29: Thread 0x40AA6D

Thread 0x40740C

Thread 0x40740C, contains several notable functions. First, a mutex "gregHDGHRTEfghRTHNNBMJKR!!EADSVXDFSWEdhstoio4io34o432m19" was created at 0x407418. If the mutex already existed], it would indicate that another instance of the malware was already running, the process would exit.

```
.text:00407418                 push    offset Name     ; "gregHDGHRTEfghRTHNNBMJKR!!E
.text:0040741D                 push    ebp             ; bInitialOwner
.text:0040741E                 push    ebp             ; lpMutexAttributes
.text:0040741F                 call    CreateMutexA
.text:00407425                 mov     hMutex, eax
.text:0040742A                 call    GetLastError
.text:00407430                 cmp     eax, 0B7h       ; ERROR_ALREADY_EXISTS
.text:00407435                 jnz     short loc_40743E
.text:00407437                 push    ebp             ; uExitCode
.text:00407438                 call    ExitProcess
```

Diagram 30: Thread 0x40740C

At 0x0x4074DD, the thread called WSAStartup requesting to use winsock version 2.2. If it failed, the process will exit.

```
.text:004074D1                 push    eax
.text:004074D2                 push    202h            ; char
.text:004074D7                 call    WSAStartup_0
.text:004074DD                 test    eax, eax
.text:004074DF                 jz      short loc_4074E9
.text:004074E1                 push    0FFFFFFFEh      ; uExitCode
.text:004074E3                 call    ExitProcess
```

Diagram 31: Thread 0x40740C

At 0x407519, thread 0410246 is created.

```
.text:0040750B                 push    eax             ; lpThreadId
.text:0040750C                 lea     eax, [esp+370h+Parameter]
.text:00407510                 push    ebp             ; dwCreationFlags
.text:00407511                 push    eax             ; lpParameter
.text:00407512                 push    offset sub_410246 ; lpStartAddress
.text:00407517                 push    ebp             ; dwStackSize
.text:00407518                 push    ebp             ; lpThreadAttributes
.text:00407519                 call    CreateThread
```

Diagram 32: Thread 0x40740C

Then it makes an interesting call to 0x40A391, where several IRC commands and IRC server numerics are listed. This further confirmed that the specimen is an IRCbot.

```
.text:0040A391 sub_40A391      proc near               ; CODE XREF: sub_4074
.text:0040A391
.text:0040A391 arg_0           = dword ptr  8
.text:0040A391
.text:0040A391                 push    esi
.text:0040A392                 mov     esi, [esp+arg_0]
.text:0040A396                 push    edi
.text:0040A397                 push    offset sub_4160E6 ; int
.text:0040A39C                 push    offset aError_0 ; "ERROR"
.text:0040A3A1                 mov     ecx, esi
.text:0040A3A3                 call    sub_4077F2
.text:0040A3A8                 push    offset sub_409D43 ; int
.text:0040A3AD                 push    offset aPrivmsg ; "PRIVMSG"
.text:0040A3B2                 mov     ecx, esi
.text:0040A3B4                 call    sub_4077F2
.text:0040A3B9                 push    offset sub_409FEB ; int
.text:0040A3BE                 push    offset aKick    ; "KICK"
.text:0040A3C3                 mov     ecx, esi
.text:0040A3C5                 call    sub_4077F2
.text:0040A3CA                 mov     edi, offset sub_40A08E
.text:0040A3CF                 mov     ecx, esi
.text:0040A3D1                 push    edi             ; int
.text:0040A3D2                 push    offset aTopic_0 ; "TOPIC"
.text:0040A3D7                 call    sub_4077F2
```

Diagram 33: Function @ 0x40A391

The IRC-related terms are listed in Appendix 6.

Function 0x40A391, is a simple function that calls 0x4077F2 multiple times, each with a different set of parameters. The parameters are a pointer to function and an IRC command. The functions associated with the commands “PRIVMSG” and “TOPIC” are 0x409D43 and 0x40A08E respectively. Both functions would make calls to 0x4014B0. The process flow from ServiceMain to 0x40AA6D to 0x40740C to 0x40A391 and finally down to 0x4014B0 is illustrated below.

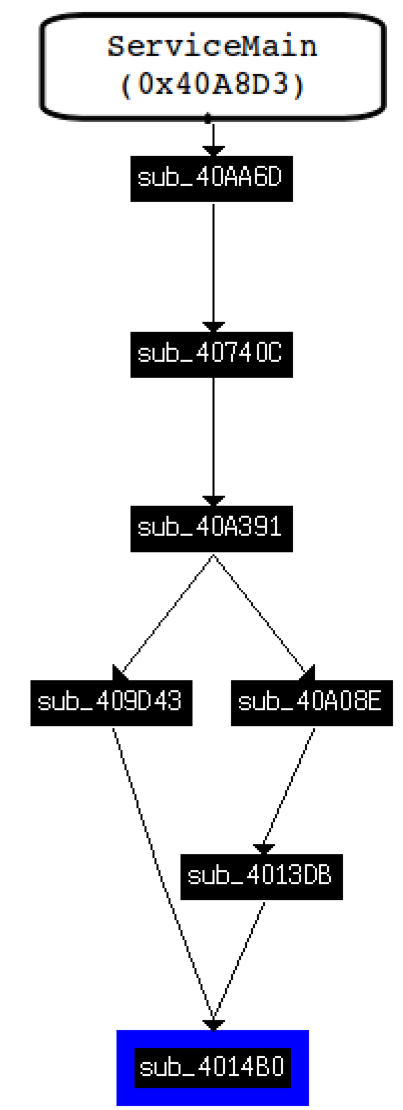

Diagram 34: Flow chart of main function calls

Function 0x4014B0 contained a list of IRC commands which is documented in Appendix 6. The list of commands suggests that the specimen had Denial of Service (DOS) capabilities. Each IRC command was paired with its own thread which would be launched when the command was issued.

Taking command “trollflood” as an example, it would launch thread 0x4153D3. The most significant function in this thread is a loop segment starting at 0x4154FC that would continuously open a socket and connect to a random location.

```
.text:004154FC loc_4154FC:                             ; CODE XREF: sub_4153D
.text:004154FC                 cmp     dword ptr [ebp+hostshort], 0
.text:00415503                 jnz     short loc_41551F
.text:00415505                 call    rand
.text:0041550B                 cdq
.text:0041550C                 mov     ecx, 0FFFFh
.text:00415511                 idiv    ecx
.text:00415513                 inc     edx
.text:00415514                 push    edx
.text:00415515                 call    ntohs
.text:0041551B                 mov     word ptr [ebp+name.sa_data], ax
.text:0041551F
.text:0041551F loc_41551F:                             ; CODE XREF: sub_4153D
.text:0041551F                 push    dword ptr [esi] ; s
.text:00415521                 call    closesocket
.text:00415527                 push    0               ; protocol
.text:00415529                 push    1               ; type
.text:0041552B                 push    2               ; af
.text:0041552D                 call    socket
.text:00415533                 lea     ecx, [ebp+name]
.text:00415536                 push    10h             ; namelen
.text:00415538                 push    ecx             ; name
.text:00415539                 push    eax             ; s
.text:0041553A                 mov     [esi], eax
.text:0041553C                 call    connect
.text:00415542                 inc     edi
.text:00415543                 add     esi, 4
.text:00415546                 cmp     edi, [ebp+var_D0]
.text:0041554C                 jl      short loc_4154FC
```

Diagram 35: trollflood function

SMFIO 3: Profiling infection

The bulk of the investigation effort was spent in static analysis. The mutex, strings and malware capabilities discovered would be useful for profiling the attack. From the analysis, the specimen is capable of :

- performing anti-forensics strategies.
- accessing sensitive system settings like registry, system folders.
- contacting an external C&C server.
- performing DOS attacks
- downloading external data

The proposed OpenIOC indicators are listed below.

```
AND
    Process Handle Name contains gregHDGHRTEfghRTHNNBMJKR!!EADSVXDFSWEdhstoio4io34o432m19
OR
    OR
        Process StringList contains PRIVMSG,KICK,TOPIC,001,005,332,366,376,422,433
        Process StringList contains l.in,log.in,l.out,lo,rmcc.die,rmcc.now,advscan,asc
        Process StringList contains threads,t,ipcc.wget,ipcc.download,r0flzcc.updt,r4wrcc.nb
        Process StringList contains trollflood,ccflood,ccgetflood,tcpsyn,visit,akicmp,patcher,opentem
        Process StringList contains tcp,tfn2ksyn,akudp,aksyn,sky,ddosstop,bandwidthflood,udpx,udp,ping
```

Diagram 36: OpenIOC for static analysis

These indicators only described what can be observed from an infected machine but not the capabilities and damage potential of the malware.

5. Conclusion

The Simplified Malware Forensics Investigation Objectives was used when performing malware analysis and the results were documented in OpenIOC. The result is presented in Appendix 7. This provides a reliable and consistent manner of reporting the infection. IT systems monitoring tools can be configured with the OpenIOC indicators. For example, a OpenIOC to yara[36] conversion might look like this.

```
AND
  Process Handle Name contains gregHDGHRTEfghRTHNNBMJKR!!EADSVXDFSWEdhstoio4io34o432m19
OR
  OR
    Process StringList contains PRIVMSG,KICK,TOPIC,001,005,332,366,376,422,433
    Process StringList contains l.in,log.in,l.out,lo,rmcc.die,rmcc.now,advscan,asc
    Process StringList contains threads,t,ipcc.wget,ipcc.download,r0flzcc.updt,r4wrcc.nb
    Process StringList contains trollflood,ccflood,ccgetflood,tcpsyn,visit,akicmp,patcher,opentem
    Process StringList contains tcp,tfn2ksyn,akudp,aksyn,sky,ddosstop,bandwidthflood,udpx,udp,ping
```

```
/* yara rule */
rule network_attacks
{
    meta:
        description = "Indicates network attacking capabilities"

    strings:
        $nwatt0 = "trollflood" nocase fullword
        $nwatt1 = "ccflood" nocase fullword
        $nwatt2 = "ccgetflood" nocase fullword
        $nwatt3 = "tcpsyn" nocase fullword
        $nwatt4 = "akicmp" nocase fullword
        $nwatt5 = "tcp" nocase fullword
        $nwatt6 = "tfn2ksyn" nocase fullword
        $nwatt7 = "akudp" nocase fullword
        $nwatt8 = "aksyn" nocase fullword
        $nwatt9 = "sky" nocase fullword
        $nwatt10 = "ddosstop" nocase fullword
        $nwatt11 = "bandwidthflood" nocase fullword
        $nwatt12 = "udpx" nocase fullword
        $nwatt13 = "udp" nocase fullword
        $nwatt14 = "ping" nocase fullword

    condition:
        $nwatt0 or $nwatt1 or $nwatt2 or $nwatt3 or $nwatt4 or
        $nwatt5 or $nwatt6 or $nwatt7 or $nwatt8 or $nwatt9 or
        $nwatt10 or $nwatt11 or $nwatt12 or $nwatt13 or $nwatt14
```

Diagram 37: OpenIOC to yara conversion

[36] http://code.google.com/p/yara-project/

A OpenIOC to snort[37] conversion might look like this.

```
OR
  Network String General contains blue3
  Network String URI is http://checkipdyndns.org
  Network String URI is http://www.ip138.com
  AND
    Port Remote IP contains 60.10.179.100
    Port remotePort contains 8680 to 8689
```

```
#snort rules

alert tcp $HOME_NET any -> 60.10.179.100 any (msg:"possible host infection"; sid:1000001;
rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET [80,443] (msg:"alert! possible host infection";
classtype:web-application-activity; reference:url,www.checkipdyns.org; sid:1000002; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET [80,443] (msg:"alert! possible host infection";
classtype:web-application-activity; reference:url,www.ip138.com; sid:1000002; rev:1;)

alert tcp $HOME_NET any -> any 8680:8689 (msg:"alert! possible host infection";
flow:to_server,established; content:"#blue3"; sid:1000003; rev:1;)
```

Diagram 38: OpenIOC to snort conversion

However the OpenIOC indicators proposed so far, only describes the low-level file, host and network attributes but lack the syntax to provide the semantics behind the attributes. A simple way to overcome this is to include an attribute for describing the objective of the set of indicators.

```
AND [identifying infection]
  Process Handle Name contains gregHDGHRTEfghRTHNNBMJKR!!EADSVXDFSWEdhstoio4io34o432m19
OR [profiling infection]
  OR
    Process StringList contains PRIVMSG,KICK,TOPIC,001,005,332,366,376,422,433
    Process StringList contains l.in,log.in,l.out,lo,rmcc.die,rmcc.now,advscan,asc
    Process StringList contains threads,t,ipcc.wget,ipcc.download,r0flzcc.updt,r4wrcc.nb
    Process StringList contains trollflood,ccflood,ccgetflood,tcpsyn,visit,akicmp,patcher,opentem
    Process StringList contains tcp,tfn2ksyn,akudp,aksyn,sky,ddosstop,bandwidthflood,udpx,udp,ping
```

Diagram 39: Modified OpenIOC indicators

To conclude, OpenIOC provides a simple and effective way of describing a malware infection. As its syntax is based on XML, it can be easily transformed to a format that can be used by IT monitoring tools like yara and snort. However, the current OpenIOC lacks the ability to provide semantics behind the attributes but this can be overcome by providing additional attributes to the XML syntax.

[37]http://www.snort.org

6. References

Aquilina, J. M., Malin, C. H., & Casey, E. (2010). Malware forensic field guide for windows systems, digital forensics field guides. New York: Syngress.

Barnum, S. (2011, Nov. 2). Cyber Observable eXpression (CybOX) Use Cases. Retrieved from http://cybox.mitre.org/documents/Cyber%20Observable%20eXpression%20(CybOX)%20Use%20Cases%20-%20(ITSAC%202011)%20-%20Sean%20Barnum.pdf

Casey, E. (2011). Handbook of digital forensics and investigation. Burlington: Academic Press.

JSI Tip 0324 - Registry entries for services (1997, Nov 24). Retrieved from http://www.windowsitpro.com/article/registry2/jsi-tip-0324-registry-entries-for-services-.

Kirillov, I. (2012, Febuary 08). An introduction to the malware attribute enumeration and characterization white paper. Retrieved from https://maec.mitre.org/about/docs/Introduction_to_MAEC_white_paper.pdf

Leydon, John. (2012, September 20). Sophos antivirus classifies its own update kit as malware. Retrieved from http://www.theregister.co.uk/2012/09/20/sophos_auto_immune_update_chaos/.

Liston, Tom. (2006). On the cutting edge: thwarting virtual machine detection. Retrieved from http://handlers.sans.org/tliston/ThwartingVMDetection_Liston_Skoudis.pdf

Murray, Jim. (2012, October 16). Analysis of the incident handling six-step process . Retrieved from http://www.giac.org/cissp-papers/17.pdf

Paxson, V. (2011, April 19). Viruses and worms. Retrieved from http://inst.eecs.berkeley.edu/~cs161/sp11/slides/4.19.virus-worms.pdf

Collake, J. (2005, April 25). PECompact v2.0 Anti-Virus Interoperability Technical Document. Retrieved from http://www.bitsum.com/pec2av.htm

Sophisticated indicators for the modern threat landscape: an instruction to OpenIOC (2011). Retrieved from http://openioc.org/resources/An_Introduction_to_OpenIOC.pdf

Appendix 1: Analysis Tools

Windows 7

- CaptureBat
- IDA
- OllyDbg
- PEBrowse
- PEiD
- Regshot
- Sysinternals Process Explorer
- Sysinternals Process Monitor

REMnux

- fakedns
- ircd server
- wireshark

Appendix 2: IOC Terms

The full list of IOC indicator terms retrieved on 10 Oct 2012 are listed below (http://openioc.org/terms/Current.iocterms):

Indicator Name
ArpEntryItem
CookieHistory
DiskItem
DnsEntryItem
DriverItem
Email
EventLogItem
FileDownloadHistoryItem
FileItem
FormHistoryItem
HiveItem
HookItem
ModuleItem
Network
PortItem
PrefetchItem

ProcessItem
RegistryItem
RouteEntryItem
ServiceItem
SystemInfoItem
SystemRestoreItem
TaskItem
UrlHistoryItem
UserItem
VolumnItem

Appendix 3: Zeus IOC


```
<?xml version="1.0" encoding="us-ascii"?>
<ioc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
id="6d2a1b03-b216-4cd8-9a9e-8827af6ebf93" last-modified="2011-10-28T19:28:20"
xmlns="http://schemas.mandiant.com/2010/ioc">
  <short_description>Zeus</short_description>
  <description>Finds Zeus variants, twexts, sdra64, ntos</description>
  <keywords />
  <authored_by>Mandiant</authored_by>
  <authored_date>0001-01-01T00:00:00</authored_date>
  <links />
  <definition>
    <Indicator operator="OR" id="9c8df971-32a8-4ede-8a3a-c5cb2c1439c6">
      <Indicator operator="AND" id="0781258f-6960-4da5-97a0-ec35fb403cac">
        <IndicatorItem id="50455b63-35bf-4efa-9f06-aeba2980f80a" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/name" type="mir" />
          <Content type="string">winlogon.exe</Content>
        </IndicatorItem>
        <IndicatorItem id="b05d9b40-0528-461f-9721-e31d5651abdc" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Type" type="mir" />
          <Content type="string">File</Content>
        </IndicatorItem>
        <Indicator operator="OR" id="67505775-6577-43b2-bccd-74603223180a">
          <IndicatorItem id="c5ae706f-c032-4da7-8acd-4523f1dae9f6" condition="contains">
            <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Name" type="mir" />
            <Content type="string">system32¥sdra64.exe</Content>
          </IndicatorItem>
          <IndicatorItem id="25ff12a7-665b-4e45-8b0f-6e5ca7b95801" condition="contains">
            <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Name" type="mir" />
            <Content type="string">system32¥twain_32¥user.ds</Content>
          </IndicatorItem>
          <IndicatorItem id="fea11706-9ebe-469b-b30a-4047cfb7436b" condition="contains">
            <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Type" type="mir" />
            <Content type="string">¥WINDOWS¥system32¥twext.exe</Content>
          </IndicatorItem>
          <IndicatorItem id="94ac992c-8d6d-441f-bfc4-5235f9b09af8" condition="contains">
            <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Name" type="mir" />
            <Content type="string">system32¥twain32¥local.ds</Content>
          </IndicatorItem>
          <IndicatorItem id="bc12f44e-7d93-47ea-9cc9-86a2beeaa04c" condition="contains">
            <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Name" type="mir" />
            <Content type="string">system32¥twext.exe</Content>
          </IndicatorItem>
          <IndicatorItem id="1c3f8902-d4e2-443a-a407-15be3951bef9" condition="contains">
            <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Name" type="mir" />
            <Content type="string">system32¥lowsec¥user.ds</Content>
          </IndicatorItem>
          <IndicatorItem id="7fab12d1-67ed-4149-b46a-ec50fc622bee" condition="contains">
            <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Name" type="mir" />
            <Content type="string">system32¥lowsec¥local.ds</Content>
          </IndicatorItem>
        </Indicator>
      </Indicator>
      <Indicator operator="AND" id="9f7a5703-8a26-45cf-b801-1c13f0f15d40">
        <IndicatorItem id="cf77d82f-0ac9-4c81-af0b-d634f71525b5" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Type" type="mir" />
          <Content type="string">Mutant</Content>
        </IndicatorItem>
        <Indicator operator="OR" id="83f72cf7-6399-4620-b735-d08ce23ba517">
```

```
          <IndicatorItem id="a1250d55-cd63-46cd-9436-e1741f5f42c7" condition="contains">
            <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Name" type="mir" />
            <Content type="string">__SYSTEM__</Content>
          </IndicatorItem>
          <IndicatorItem id="e033b865-95ba-44ab-baa5-3b1e8e5f348c" condition="contains">
            <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Name" type="mir" />
            <Content type="string">_AVIRA_</Content>
          </IndicatorItem>
        </Indicator>
      </Indicator>
    </Indicator>
  </definition>
</ioc>
```

Appendix 4: a.bat

```
@echo off
Echo REGEDIT4>%temp%\1.reg
Echo.>>%temp%\1.reg
Echo  [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess]>>%temp%\1.reg
Echo "Start"=dword:00000002>>%temp%\1.reg
Echo.>>%temp%\1.reg
Echo  [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy
\StandardProfile]>>%temp%\1.reg
Echo "EnableFirewall"=dword:00000000>>%temp%\1.reg
Echo.>>%temp%\1.reg
Echo  [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\wuauserv]>>%temp%\1.reg
Echo "Start"=dword:00000004>>%temp%\1.reg
Echo.>>%temp%\1.reg
Echo  [HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\wscsvc]>>%temp%\1.reg
Echo "Start"=dword:00000004>>%temp%\1.reg
Echo.>>%temp%\1.reg
Echo  [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters]>>%temp%\1.reg
Echo "MaxFreeTcbs"=dword:000007d0>>%temp%\1.reg
Echo "MaxHashTableSize"=dword:00000800>>%temp%\1.reg
Echo "TcpTimedWaitDelay"=dword:0000001e>>%temp%\1.reg
Echo "MaxUserPort"=dword:0000f618>>%temp%\1.reg
Echo.>>%temp%\1.reg
START /WAIT REGEDIT /S %temp%\1.reg
DEL %temp%\1.reg
DEL %0
```

Diagram 40: A copy of a.bat

Appendix 5 : De-obfuscating PECompact

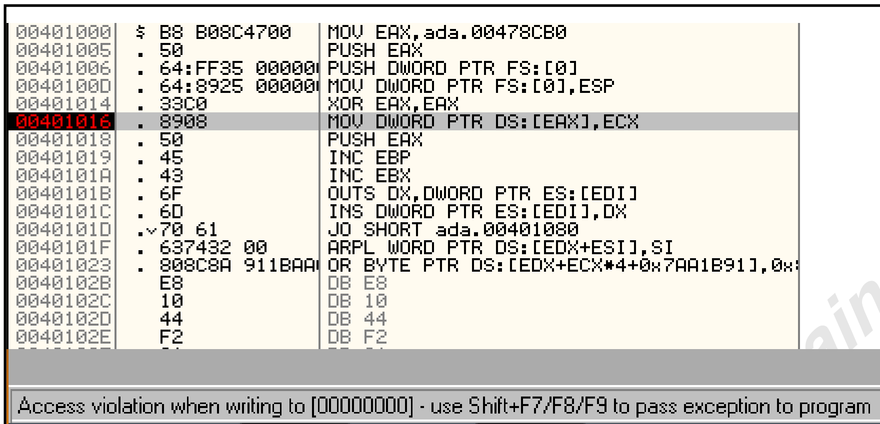

Diagram 41: OEP of obfuscated code

PECompact uses SEH (Structured Exception Handling) mechanism to hide the OEP of the malicious code. The OEP of the obfuscated code contains very few lines of code. In x86 machines, FS:[0][38] points to the head of the EXCEPTION_RECORD list. At 0x40100D, the address 0x478CB0 is move to FS:[0]. The "XOR EAX, EAX" command set the value of EAX register to 0. An exception is generated at 0x401016, when there is a move to address DS:[EAX]. The key press "Shift+F9" will return control to address 0x478CB0.

[38]http://msdn.microsoft.com/en-us/library/ms253960(v=vs.80).aspx

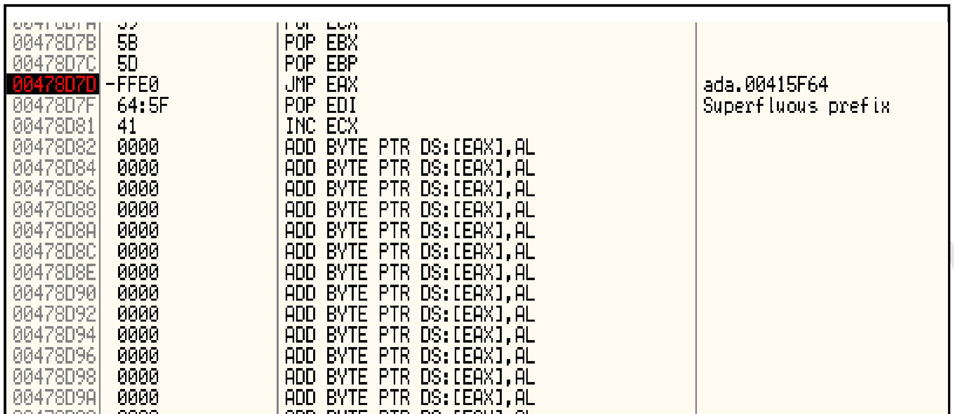

Diagram 42: JMP to OEP

0x478CB0 contains the de-obfuscation routines which ends at 0x0x478D7D with a "JMP EAX" which jumps to the OEP at 0x0415F64.

Appendix 6: IRC & Malicious Commands

IRC Term[39]	Remarks
ERROR	Use by servers to report serious errors to operators.
PRIVMSG	Use to send private messages between users.
KICK	Use to forcibly remove a user from a channel.
TOPIC	Use to change or view the topic of a channel.
001	Send to all clients when a connection is established.
332	Server reply to a TOPIC message. Indicates that a topic is set.
366	Server returned at the end of a NAMES list .
005	Server reply to a MAP command. The reply will contain a string showing the relative position of a server.
376	Server reply to a MOTD (Message Of The Day) request. This is sent after message of the day string is sent.
422	Server reply is a MOTD file is missing.
433	Server reply when the user is being invited into a channel that it is already on.

Table 6: IRC Command at 0x40A391

Strings	Remarks
l.in	Change to channel #2k38
log.in	Change to channel #2k38
l.out	-
lo	-
rmcc.die	Delete service “PlugPlayCM” and release

[39]https://tools.ietf.org/html/rfc1459#section-4.1.4

Strings	Remarks
	mutex
rmcc.now	Delete service "PlugPlayCM" and release mutex
advscan	-
asc	-
threads	-
t	-
ipcc.wget	-
ipcc.download	-
r0flzcc.updt	-
r4wrcc.nb	-
tcp	-
tfn2ksyn	-
akudp	-
aksyn	-
sky	-
ddosstop	-
bandwidthflood	-
udpx	-
udp	-
ping	-
trollflood	Launch thread 0x4153D3 which would continuously open a socket and connect to a random location.
ccflood	-
ccgetflood	-
tcpsyn	-
visit	-
akicmp	-
patcher	-

Strings	Remarks
opentem	-

Table 7: Malicious commands at 0x4014B0

Appendix 7: IOC Terms


```
<?xml version="1.0" encoding="us-ascii"?>
<ioc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" id="26184e25-a226-442a-9a0c-81f553afd7ea"
last-modified="2012-12-01T23:39:44" xmlns="http://schemas.mandiant.com/2010/ioc">
  <short_description>ada</short_description>
  <authored_by>lhy</authored_by>
  <authored_date>2012-10-25T08:40:38</authored_date>
  <links />
  <definition>
    <Indicator operator="OR" id="1eaa7fa8-ac8a-430b-96bc-a579064999cb">
      <Indicator operator="AND" id="2e271d9c-632c-4c27-9428-ae5a3377aa5f">
        <IndicatorItem id="b74ce978-280c-4d31-9b78-5442b826305d" condition="contains">
          <Context document="FileItem" search="FileItem/FullPath" type="mir" />
          <Content type="string">c:¥Windows¥System32</Content>
        </IndicatorItem>
        <IndicatorItem id="048e5e8b-a2c3-4fa6-b9f7-604302f3a85f" condition="contains">
          <Context document="FileItem" search="FileItem/FileName" type="mir" />
          <Content type="string">serives.exe</Content>
        </IndicatorItem>
        <IndicatorItem id="4edb0110-44be-4ce5-8b87-bf92e1e16ca3" condition="is">
          <Context document="FileItem" search="FileItem/Md5sum" type="mir" />
          <Content type="md5">aada169a1cbd822e1402991e6a9c9238</Content>
        </IndicatorItem>
      </Indicator>
      <Indicator operator="AND" id="f2d259ea-351c-4cb1-9b46-c879da03755a">
        <IndicatorItem id="b714f6f0-8e01-453a-8816-7b7a1d1a0a27" condition="contains">
          <Context document="RegistryItem" search="RegistryItem/KeyPath" type="mir" />
          <Content
type="string">HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥SharedAccess</Conte
nt>
        </IndicatorItem>
        <IndicatorItem id="082069e4-f589-48e9-989b-d1c1c39f0dbd" condition="contains">
          <Context document="RegistryItem" search="RegistryItem/ValueName" type="mir"
/>
          <Content type="string">Start</Content>
        </IndicatorItem>
        <IndicatorItem id="4ce571ae-f7ea-45c6-901c-396537eb4d45" condition="contains">
          <Context document="RegistryItem" search="RegistryItem/Value" type="mir" />
          <Content type="string">2</Content>
        </IndicatorItem>
      </Indicator>
      <Indicator operator="AND" id="f71f0662-bd9c-4f13-ac39-a0454655f565">
        <IndicatorItem id="e9773c12-d05e-4097-aa44-817e5a81a6f1" condition="contains">
          <Context document="RegistryItem" search="RegistryItem/KeyPath" type="mir" />
          <Content
type="string">HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥SharedAccess¥Parame
```

```
ters¥FirewallPolicy¥StandardProfile</Content>
        </IndicatorItem>
        <IndicatorItem id="3f269fac-ce74-4629-810e-4aa7f5ac8d4f" condition="contains">
          <Context document="RegistryItem" search="RegistryItem/ValueName" type="mir"
/>
          <Content type="string">EnableFirewall</Content>
        </IndicatorItem>
        <IndicatorItem id="39a1a564-0d94-40a4-a450-bc354d4a27ae" condition="contains">
          <Context document="RegistryItem" search="RegistryItem/Value" type="mir" />
          <Content type="string">0</Content>
        </IndicatorItem>
      </Indicator>
      <Indicator operator="AND" id="4c05075e-1345-4ba3-a349-ee78e599872b">
        <IndicatorItem id="1f6b857e-6f78-4843-ae58-3f2c511aea8c" condition="contains">
          <Context document="RegistryItem" search="RegistryItem/Value" type="mir" />
          <Content type="string">Start</Content>
        </IndicatorItem>
        <IndicatorItem id="f8da3d69-191e-4e15-9ed7-8f2aa9b13add" condition="contains">
          <Context document="RegistryItem" search="RegistryItem/Value" type="mir" />
          <Content type="string">4</Content>
        </IndicatorItem>
        <Indicator operator="AND" id="3464b433-cfb0-4c61-ae98-e29b5de2a37c">
          <IndicatorItem id="c2c58f87-7273-449e-97ae-54b1776c7a76"
condition="contains">
            <Context document="RegistryItem" search="RegistryItem/KeyPath" type="mir"
/>
            <Content
type="string">HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥wuauserv</Content>
          </IndicatorItem>
          <IndicatorItem id="5b872ff7-29b6-4e87-bc25-81912dc66ce0"
condition="contains">
            <Context document="RegistryItem" search="RegistryItem/KeyPath" type="mir"
/>
            <Content
type="string">HKEY_LOCAL_MACHINE¥SYSTEM¥ControlSet001¥Services¥wscsvc</Content>
          </IndicatorItem>
        </Indicator>
      </Indicator>
      <Indicator operator="AND" id="72e4fbfd-a3cc-4e29-86d5-3ebfcfe101f6">
        <IndicatorItem id="f7d42cbd-7b03-4734-bc97-e400b57d5fe5" condition="contains">
          <Context document="RegistryItem" search="RegistryItem/Path" type="mir" />
          <Content
type="string">HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥Tcpip¥Parameters</C
ontent>
        </IndicatorItem>
        <Indicator operator="AND" id="39ff1ac5-2bf5-4c7e-b502-43249890ad75">
          <IndicatorItem id="ebe9f693-1844-4dcc-9efe-f7319c934928"
condition="contains">
            <Context document="RegistryItem" search="RegistryItem/ValueName"
```

```
type="mir" />
            <Content type="string">MaxFreeTcbs</Content>
          </IndicatorItem>
          <IndicatorItem id="d3b872d5-a2dd-4eb4-a456-04a329d7e6e6"
condition="contains">
            <Context document="RegistryItem" search="RegistryItem/Value" type="mir" />
            <Content type="string">0x7d0</Content>
          </IndicatorItem>
        </Indicator>
        <Indicator operator="AND" id="92a30677-692d-4bd5-9040-40a4fca4d11f">
          <IndicatorItem id="61672e9b-960b-456e-a642-cc934c9678c8"
condition="contains">
            <Context document="RegistryItem" search="RegistryItem/ValueName"
type="mir" />
            <Content type="string">MaxHashTableSize</Content>
          </IndicatorItem>
          <IndicatorItem id="72a9e94c-af1a-4987-bced-97bced06986b"
condition="contains">
            <Context document="RegistryItem" search="RegistryItem/Value" type="mir" />
            <Content type="string">0x800</Content>
          </IndicatorItem>
        </Indicator>
        <Indicator operator="AND" id="95fa6cf2-edb6-457c-a5b8-81c44f4b4c04">
          <IndicatorItem id="5df0d18d-6c39-40fb-b634-b43a9e2f7113"
condition="contains">
            <Context document="RegistryItem" search="RegistryItem/ValueName"
type="mir" />
            <Content type="string">TcpTimedWaitDelay</Content>
          </IndicatorItem>
          <IndicatorItem id="4382586f-8991-49f6-bf06-652869c698f3"
condition="contains">
            <Context document="RegistryItem" search="RegistryItem/Value" type="mir" />
            <Content type="string">0x1e</Content>
          </IndicatorItem>
        </Indicator>
        <Indicator operator="AND" id="cfa20067-7e94-4367-8cf1-d2aa70b587d1">
          <IndicatorItem id="0caf1f49-2348-444e-9db0-82e139bfa73f"
condition="contains">
            <Context document="RegistryItem" search="RegistryItem/ValueName"
type="mir" />
            <Content type="string">MaxUserPort</Content>
          </IndicatorItem>
          <IndicatorItem id="b9372dcd-5ca3-4cbe-b259-8e652dd97b1e"
condition="contains">
            <Context document="RegistryItem" search="RegistryItem/Value" type="mir" />
            <Content type="string">0xf618</Content>
          </IndicatorItem>
        </Indicator>
      </Indicator>
```

```
      <Indicator operator="AND" id="7eb009b2-aa52-4553-8f96-d6ab93a504d3">
        <IndicatorItem id="35dc6746-f204-45ba-ae7e-71fd98b65f4e" condition="contains">
          <Context document="ServiceItem" search="ServiceItem/name" type="mir" />
          <Content type="string">Security Center</Content>
        </IndicatorItem>
        <IndicatorItem id="d0ca54fa-8120-4401-a438-d892ef62a465" condition="contains">
          <Context document="ServiceItem" search="ServiceItem/name" type="mir" />
          <Content type="string">PlugPlayCM</Content>
        </IndicatorItem>
      </Indicator>
      <Indicator operator="AND" id="28307beb-b70d-43fd-938d-40bff22979c9">
        <IndicatorItem id="691fc3fb-49b8-4778-9243-0b1695778498" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/name" type="mir" />
          <Content type="string">serivces.exe</Content>
        </IndicatorItem>
      </Indicator>
      <Indicator operator="OR" id="62c59ef1-6804-4494-a438-f5b77f69e11d">
        <IndicatorItem id="4c852419-e4df-44cd-b1cb-67e5feb7bb59" condition="contains">
          <Context document="Network" search="Network/String" type="network" />
          <Content type="string">blue3</Content>
        </IndicatorItem>
        <IndicatorItem id="60128b77-d1fc-4064-b078-8a6a16f9a5b2" condition="isnot">
          <Context document="Network" search="Network/URI" type="network" />
          <Content type="string">http://checkipdyndns.org</Content>
        </IndicatorItem>
        <IndicatorItem id="0b7289b2-2af3-4891-99c8-00a12b6632c7" condition="is">
          <Context document="Network" search="Network/URI" type="network" />
          <Content type="string">http://www.ip138.com</Content>
        </IndicatorItem>
        <Indicator operator="AND" id="3cbbad62-26aa-48c1-b83f-5f16095020b8">
          <IndicatorItem id="37f2aa9f-8d23-4ed4-b79a-e168ac3286ea"
condition="contains">
            <Context document="PortItem" search="PortItem/remoteIP" type="mir" />
            <Content type="IP">60.10.179.100</Content>
          </IndicatorItem>
          <IndicatorItem id="a13490ca-2e46-46f5-9410-c4f1256db815"
condition="contains">
            <Context document="PortItem" search="PortItem/remotePort" type="mir" />
            <Content type="string">8680 ? 8689</Content>
          </IndicatorItem>
        </Indicator>
      </Indicator>
      <Indicator operator="AND" id="0b5fd0cd-c37a-43f1-b8ba-07fb6795e839">
        <IndicatorItem id="37655418-7064-4c95-b18d-5137e70a5308" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/HandleList/Handle/Name"
type="mir" />
          <Content
type="string">gregHDGHRTEfghRTHNNBMJKR!!EADSVXDFSWEdhstoio4io34o432m19</Content>
        </IndicatorItem>
```

```
      </Indicator>
      <Indicator operator="OR" id="62cec0f3-e5f9-4bb1-b496-5cb63e136785">
        <Indicator operator="OR" id="dba4faa1-f463-4649-ae13-9264a567c773">
          <IndicatorItem id="08257809-514d-4eb3-b034-85f158102d07"
condition="contains">
            <Context document="ProcessItem" search="ProcessItem/StringList/string"
type="mir" />
            <Content
type="string">PRIVMSG,KICK,TOPIC,001,005,332,366,376,422,433</Content>
          </IndicatorItem>
          <IndicatorItem id="1653ed5a-7c3d-40e5-bf75-d2e078b03564"
condition="contains">
            <Context document="ProcessItem" search="ProcessItem/StringList/string"
type="mir" />
            <Content
type="string">l.in,log.in,l.out,lo,rmcc.die,rmcc.now,advscan,asc</Content>
          </IndicatorItem>
         <IndicatorItem id="1653ed5a-7c3d-40e5-bf75-d2e078b03564"
condition="contains">
            <Context document="ProcessItem" search="ProcessItem/StringList/string"
type="mir" />
            <Content
type="string">threads,t,ipcc.wget,ipcc.download,r0flzcc.updt,r4wrcc.nb</Content>
          </IndicatorItem>
           <IndicatorItem id="1653ed5a-7c3d-40e5-bf75-d2e078b03564"
condition="contains">
            <Context document="ProcessItem" search="ProcessItem/StringList/string"
type="mir" />
            <Content
type="string">trollflood,ccflood,ccgetflood,tcpsyn,visit,akicmp,patcher,opentem</Conte
nt>
          </IndicatorItem>
         <IndicatorItem id="1653ed5a-7c3d-40e5-bf75-d2e078b03564"
condition="contains">
            <Context document="ProcessItem" search="ProcessItem/StringList/string"
type="mir" />
            <Content
type="string">tcp,tfn2ksyn,akudp,aksyn,sky,ddosstop,bandwidthflood,udpx,udp,ping</Cont
ent>
          </IndicatorItem>
        </Indicator>
      </Indicator>
    </Indicator>
  </definition>
</ioc>
```

Last Updated: April 8th, 2014

Upcoming Training

SANS Security West 2014	San Diego, CA	May 08, 2014 - May 17, 2014	Live Event
Mentor Session - FOR 610	Columbia, MD	May 21, 2014 - Jul 23, 2014	Mentor
Digital Forensics & Incident Response Summit	Austin, TX	Jun 03, 2014 - Jun 10, 2014	Live Event
Community SANS Ottawa	Ottawa, ON	Jun 16, 2014 - Jun 21, 2014	Community SANS
SANSFIRE 2014	Baltimore, MD	Jun 21, 2014 - Jun 30, 2014	Live Event
SANS vLive - FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques	FOR610 - 201407,	Jul 14, 2014 - Aug 20, 2014	vLive
SANS Virginia Beach 2014	Virginia Beach, VA	Aug 18, 2014 - Aug 29, 2014	Live Event
SANS Baltimore 2014	Baltimore, MD	Sep 22, 2014 - Sep 27, 2014	Live Event
SANS DFIR Prague 2014	Prague, Czech Republic	Sep 29, 2014 - Oct 11, 2014	Live Event
SANS vLive - FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques	FOR610 - 201410,	Oct 13, 2014 - Nov 19, 2014	vLive
Community SANS Paris @ HSC - FOR610 (in French)	Paris, France	Nov 24, 2014 - Nov 28, 2014	Community SANS
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced

Last Updated: September 30th, 2017

Upcoming SANS Forensics Training

SANS DFIR Prague Summit & Training 2017	**Prague, Czech Republic**	**Oct 02, 2017 - Oct 08, 2017**	**Live Event**
SANS vLive - FOR500: Windows Forensic Analysis	**FOR500 - 201710,**	**Oct 09, 2017 - Nov 15, 2017**	**vLive**
SANS October Singapore 2017	**Singapore, Singapore**	**Oct 09, 2017 - Oct 28, 2017**	**Live Event**
Mentor Session - FOR500	**Austin, TX**	**Oct 12, 2017 - Nov 09, 2017**	**Mentor**
SANS Tysons Corner Fall 2017	**McLean, VA**	**Oct 14, 2017 - Oct 21, 2017**	**Live Event**
SANS Tokyo Autumn 2017	**Tokyo, Japan**	**Oct 16, 2017 - Oct 28, 2017**	**Live Event**
SANS vLive - FOR578: Cyber Threat Intelligence	**FOR578 - 201710,**	**Oct 16, 2017 - Nov 15, 2017**	**vLive**
SANS vLive - FOR508: Advanced Digital Forensics, Incident Response, and Threat Hunting	**FOR508 - 201710,**	**Oct 16, 2017 - Nov 22, 2017**	**vLive**
Mentor Session - FOR508	**Brasilia, Brazil**	**Oct 18, 2017 - Oct 21, 2017**	**Mentor**
SANS Berlin 2017	**Berlin, Germany**	**Oct 23, 2017 - Oct 28, 2017**	**Live Event**
SANS Seattle 2017	**Seattle, WA**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SANS Gulf Region 2017	**Dubai, United Arab Emirates**	**Nov 04, 2017 - Nov 16, 2017**	**Live Event**
SANS Milan November 2017	**Milan, Italy**	**Nov 06, 2017 - Nov 11, 2017**	**Live Event**
SANS Miami 2017	**Miami, FL**	**Nov 06, 2017 - Nov 11, 2017**	**Live Event**
SANS Sydney 2017	**Sydney, Australia**	**Nov 13, 2017 - Nov 25, 2017**	**Live Event**
Community SANS Madrid FOR585 (in Spanish)	**Madrid, Spain**	**Nov 13, 2017 - Nov 18, 2017**	**Community SANS**
SANS Paris November 2017	**Paris, France**	**Nov 13, 2017 - Nov 18, 2017**	**Live Event**
Community SANS Ottawa FOR500	**Ottawa, ON**	**Nov 20, 2017 - Nov 25, 2017**	**Community SANS**
SANS San Francisco Winter 2017	**San Francisco, CA**	**Nov 27, 2017 - Dec 02, 2017**	**Live Event**
San Francisco Winter 2017 - FOR500: Windows Forensic Analysis	**San Francisco, CA**	**Nov 27, 2017 - Dec 02, 2017**	**vLive**
SANS London November 2017	**London, United Kingdom**	**Nov 27, 2017 - Dec 02, 2017**	**Live Event**
SIEM & Tactical Analytics Summit & Training	**Scottsdale, AZ**	**Nov 28, 2017 - Dec 05, 2017**	**Live Event**
SANS Austin Winter 2017	**Austin, TX**	**Dec 04, 2017 - Dec 09, 2017**	**Live Event**
Mentor Session - AW FOR526	**Columbia, SC**	**Dec 04, 2017 - Dec 08, 2017**	**Mentor**
Community SANS Ottawa FOR610	**Ottawa, ON**	**Dec 04, 2017 - Dec 09, 2017**	**Community SANS**
SANS Munich December 2017	**Munich, Germany**	**Dec 04, 2017 - Dec 09, 2017**	**Live Event**
SANS Cyber Defense Initiative 2017	**Washington, DC**	**Dec 12, 2017 - Dec 19, 2017**	**Live Event**
SANS Cyber Defense Initiative 2017 - FOR572: Advanced Network Forensics and Analysis	**Washington, DC**	**Dec 14, 2017 - Dec 19, 2017**	**vLive**
SANS Security East 2018	**New Orleans, LA**	**Jan 08, 2018 - Jan 13, 2018**	**Live Event**
Northern VA Winter - Reston 2018	**Reston, VA**	**Jan 15, 2018 - Jan 20, 2018**	**Live Event**
SANS Amsterdam January 2018	**Amsterdam, Netherlands**	**Jan 15, 2018 - Jan 20, 2018**	**Live Event**