Cuckoo Malware Analysis

Digit Oktavianto
Iqbal Muhardianto

Chapter No. 1
"Getting Started with Automated Malware Analysis using Cuckoo Sandbox"
In this package, you will find:
A Biography of the authors of the book
A preview chapter from the book, Chapter NO.1 "Getting Started with Automated Malware Analysis using Cuckoo Sandbox"
A synopsis of the book’s content
Information on where to buy this book

About the Authors

Digit Oktavianto is an IT security professional and system administrator with experience in the Linux server, network security, Security Information and Event Management (SIEM), vulnerability assessment, penetration testing, intrusion analysis, incident response and incident handling, security hardening, PCI-DSS, and system administration.

He has good experience in Managed Security Services (MSS) projects, Security Operation Centre, operating and maintaining SIEM tools, configuring and setup of IDS/IPS, Firewall, Antivirus, Operating Systems, and Applications.

He works as an information security analyst in Noosc Global, a security consultant firm based in Indonesia. Currently, he holds CEH and GIAC Incident Handler certifications. He is very enthusiastic and has a good passion in malware analysis as his main interest for research. This book is the first book that he has written, and he plans to write more about malware analysis and incident response books.

For More Information:
Iqbal Muhardianto is a security enthusiast and he is working in the Ministry of Foreign Affairs of the Republic of Indonesia. He loves breaking things apart just to know how it works. In his computer learning career, he first started with learning MS-DOS and some C programming, after being a System admin, Network Admin, and now he is an IT Security Administrator with some skills in Linux, Windows, Network, SIEM, Malware Analysis, and Pentesting.

He currently lives Norway and works as an IT Staff in the Indonesia Embassy in Oslo.

I would like to thank Allah the God Almighty, my parents and family, my friend Digit Oktavianto for inviting me to write this book, and my colleagues for their support and inspiration.

For More Information:  
Cuckoo Malware Analysis

Welcome to Cuckoo Malware Analysis. This book has especially been created to provide you with all the information you need to get set up with Cuckoo Sandbox. In this book, you will learn the basics of malware analysis using Cuckoo Sandbox, get started with submitting your first malware sample, and create a report from it. You will also find out some tips and tricks for using Cuckoo Sandbox.

What This Book Covers

Chapter 1, Getting Started with Automated Malware Analysis using Cuckoo Sandbox, gets you started with the basic installation of Cuckoo Sandbox and teaches you the basic theory in Sandboxing, how to prepare a safe environment lab for malware analysis, and troubleshoot some problems after installing Cuckoo Sandbox.

Chapter 2, Using Cuckoo Sandbox to Analyze a Sample Malware, teaches you how to use Cuckoo Sandbox and its features, how to analyze sample malicious PDF files or malicious URLs, and also covers some basics of memory forensic analysis with Cuckoo Sandbox and Volatility.

Chapter 3, Analyzing Output of Cuckoo Sandbox, will help you analyze the results from Cuckoo sandbox, demonstrate the ability to analyze memory dump in a forensic process, and simulate an analysis of a sample APT attack in collaboration with other tools such as Volatility, Yara, Wireshark, Radare, and Bokken. This chapter will also help users analyze the output from Cuckoo Sandbox more easily and clearly.

Chapter 4, Reporting with Cuckoo Sandbox, will teach you how to create a malware analysis report using Cuckoo Sandbox reporting tools and export the output data report to another format for advanced report analysis. It will start with human-readable format (TXT and HTML), MAEC format (MITRE standard format), and the ability to export a data report to the most useful format in the world (PDF).

Chapter 5, Tips and Tricks for Cuckoo Sandbox, provides you with some tips and tricks for enhancing Cuckoo's analyzing abilities during the malware analysis process. Some people from the community created interesting plugins or modules that help users perform new experiments using Cuckoo Sandbox such as automating e-mail attachments scanning with CuckooMX, and integrating Cuckoo Sandbox with Maltego project using cuckooforcanari. You will also learn how to harden your VM environment for malware analysis.

For More Information:
Malware analysis is a process of identifying malware behavior, what they are doing, what they want, and what their main goals are. Malware analysis involves a complex process in its activity. Forensics, reverse engineering, disassembly, debugging, these activities take a lot of time in the progress. The goal of malware analysis is to gain an understanding of how a malware works, so that we can protect our organization by preventing malware attacks.

**Malware analysis methodologies**

There are two common methodologies of the malware analysis process commonly used by malware analysts: **static analysis** (or code analysis) and **dynamic analysis** (or behavior analysis). These two techniques allow analysts to understand quickly, and in detail, the risks and intentions of a given sample malware.

For performing static analysis, you need a strong understanding in programming and x86 assembly language concept. During the static analysis process, you don't have to execute the malware. Generally, the source code of malware samples is not readily available. You have to do disassembling and decompiling first, and after successfully performing reverse engineering you can analyze the low-level assembly code. Most malware analysts perform a static analysis at an earlier stage in the malware analysis process because it is safer than dynamic analysis. The challenge in static analysis is the complexity in modern malware, where some of the malware implement anti-debugging systems to prevent malware analysts from analyzing the pieces of code.

For More Information:

Dynamic analysis (behavior analysis) is a process in malware analysis that performs an execution of the malware itself and observes the malware activity. It also observes the changes that occur when the malware is being executed. Infecting a system with malware from the wild can be very dangerous. Malware infection on your system can cause damage to your system such as file deletion, change in registry, file modification, stealing confidential data/information, and so on. When performing malware analysis, you need a safe environment and the network should not connect to production networks. With dynamic analysis, you can monitor the changes made to the filesystem, registry, processes, and its network communication. The advantage of performing dynamic analysis is that you can fully understand how a malware works.

To handle the number of malware samples, some automated malware analysis techniques have been developed. Automating some aspects of malware analysis is critical for organizations processing large numbers of malicious programs. Automation will allow analysts to focus more on the tasks that need more attention in human analysis.

When using Cuckoo as an automated malware analysis tool, it is expected to reduce the amount of time analyzing a malware in a conventional way. There are some steps in dynamic malware analysis that require a lot of time; one of the instances are while we're setting up a virtualized environment for a malware to run. The process may seem easy, but if we have several malware to analyze, it will be pretty time-consuming.

**Basic theory in Sandboxing**

As malware became more sophisticated, we needed more technology that would allow us to analyze malware easily without compromising our system. One such technology that can be used is sandboxing. Sandboxing has a wide and various explanation among IT people. For a reference, you can see the explanation from Wikipedia at [http://en.wikipedia.org/wiki/Sandbox_(computer_security)](http://en.wikipedia.org/wiki/Sandbox_(computer_security)). In specific terminology (computer security), sandboxing is a technique for isolating a program (in this case, malware) by providing confined execution environments, which can be used for running unreliable programs from the main environment. To give a clear explanation about sandboxing technology, let's imagine a sandbox or sandpit playground for children. Sandpit is a container filled with sand for children to play. The "pit" or "box" itself is simply a container for storing the sand so that it does not spread outward across lawns or other surrounding surfaces. The children can do anything in the sandpits as long as they are still in the sandbox. By providing a sandbox, we can execute malicious applications and see the malware activities.
We can also analyze the malware safely and securely without worrying about the changes that will occur during the process. There are several malware sandboxes you can use for building your own automated malware analysis lab. For example, Buster Sandbox Analyzer, Zero Wine, Malheur, Cuckoo Sandbox, and so on. Cuckoo is the right tool to perform an analysis for a sandboxed malware because Cuckoo has a complete feature, it is fully open source, and has good support from its community.

**Malware analysis lab**

What is a malware analysis lab, and why should we build a malware lab? Malware lab is a safe environment to analyze malware. Basically, it is an isolated environment which contains a lot of useful tools for malware analysts that helps them in analyzing the malicious software. We should build a malware lab to be more proactive to new and modern threats that can suddenly attack our organization. It is also a form of advanced detection before antivirus vendors found a new malware specimen. The scope of the malware analysis lab can be determined by examining the processes that will occur in the malware analysis process.

Static analysis involves disassembling and reverse engineering the code of the malware. This can be done in a static state where the code is analyzed without being executed. No complex configuration is required for the lab, because actually you won't execute the malware itself. This lab is provided just to safeguard if you accidentally execute the binary malware when you are performing the code analysis. For dynamic analysis, you need to set up a more complex lab, as you need to execute the malware. Malware behaves differently depending on the operating system environment where they are being executed.

You should pay more attention regarding the location of malware analysis hosts on your network. Trojan, worms, and other types of malware can be self-replicating, so it's highly likely that simply running an executable code on a production network can lead to another machine on the same network being infected.

Setting up a malware analysis lab is actually quite simple and requires a minimum amount of hardware. Isolating your malware analysis lab from other computers in the network is not enough. In addition, you also need to isolate your lab from the Internet if you are not sure. You should consider this option, because sometimes a malware needs to communicate with the malware **author server**, for example, Botnet command and control servers.

For More Information:  
There are two options in building a malware analysis lab, that is, a physical environment and a virtualization environment. As mentioned earlier, both of them have advantages and disadvantages. Building your physical lab will require a lot of money and time in building the environment as well. In this situation, building a malware lab using the virtualization technique will save your money and time. Virtualization software allows you to save the state of a virtual machine as it runs so that you can revert back to it when necessary. This term is usually called snapshot. Using this snapshots feature, you can have a virtual machine environment that contains an operating system with a full set of weapons of dynamic and static analysis tools, and then perform a dynamic analysis with the malware, and finally you can save the session using the snapshot feature so that you can load the initial infected state at will. After finishing your malware analysis, you can choose to save or discard that snapshot and revert back to a clean image. Then, using the snapshot feature, you do not have to worry about malware that will infect your Guest OS, as you will be able to easily restore to the previous state.

From now on, you can be aware that the automated analyses of malware, which uses virtualization in operating systems, will help you to shorten the time in analyzing malware samples. Virtualization technologies have become a key component in automated malware analyses because of the cost effectiveness in hardware consumption and CPU resource utilization. By using a popular operating system and intentionally infecting it with a captured malware sample, it is generally useful to monitor the activities of the malware and determine the suspicious activities that occurs. The drawback of implementing automated malware analysis is that this method can be easily detected by malware writers as it frequently uses evasion techniques such as anti-debugging, packers, encryption, obfuscating code, and so on. But you can try to hide as many virtualization traces as possible. There is a lot of information on the Internet regarding virtualization detection techniques and countermeasures of malware analysis.

**Cuckoo Sandbox**

As described in its official website (http://www.cuckoosandbox.org/), Cuckoo is a malware sandboxing utility which has practical applications of the dynamical analysis approach. Instead of statically analyzing the binary file, it gets executed and monitored in real time. As a simple explanation, Cuckoo is an open source automated malware analysis system that allows you to perform analysis on sandboxed malware. Cuckoo Sandbox started as a Google Summer of Code project in 2010 within the Honeynet Project. After the initial work during the summer of 2010, the first beta release was published on February 5th, 2011, when Cuckoo was publicly announced and distributed for the first time.

For More Information:

Cuckoo was originally designed and developed by Claudio "nex" Guarnieri, who is still the main developer and coordinates all efforts from joined developers and contributors. In March 2012, Cuckoo Sandbox won the first round of the Magnificent7 program organized by Rapid7. Cuckoo was chosen by Rapid7 for the first round of Magnificent7 sponsorships due to the developers' innovative approach to traditional and mobile-based malware analysis. Cuckoo is used to automatically run and analyze files and collect comprehensive analysis results that outline what the malware does while running inside an isolated Windows operating system. Cuckoo is designed for use in analyzing the following kinds of files:

- Generic Windows executables
- DLL files
- PDF documents
- Microsoft Office documents
- URLs
- PHP scripts
- Almost everything else

Cuckoo can also produce the following types of results:

- Traces of win32 API calls performed by all processes spawned by the malware
- Files being created, deleted, and downloaded by the malware during its execution
- Memory dumps of the malware processes
- Network traffic trace in PCAP format
- Screenshots of the Windows desktop taken during the execution of the malware
- Full memory dumps of the machines

Cuckoo Sandbox consists of a central management software, which handles malware sample executions and analyses.

Each analysis is launched in a fresh and isolated virtual machine. Cuckoo's infrastructure is composed by a host machine (the management software) and a number of guest machines (virtual machines for analysis).
The host runs the core component of the sandbox that manages the whole analysis process, whereas the guests are the isolated environments where the malware actually get safely executed and analyzed. The following diagram shows Cuckoo’s architecture:

![Cuckoo's Architecture Diagram]

**Installing Cuckoo Sandbox**

Let us see what the important components are when installing Sandbox.

**Hardware requirements**

There are no specific requirements for hardware equipment. Requirements for minimum RAM is 2 GB (for virtualization) and free space in the hard disk drive of about 40 GB. In this book, I will use the following hardware specifications as the Host OS:

- Quad Core CPU
- 4 GB RAM
- 320 GB HDD

For More Information:

Preparing the host OS
Theoretically, Cuckoo Sandbox can run on every Linux operating system. In this book, all instructions in the Host OS will be conducted in Ubuntu 12.04.

Requirements
Before continuing to the installation and configuration process, you need to install some applications and libraries.

Install Python in Ubuntu
We need to type in the following command:

```
$ sudo apt-get install python
```

Cuckoo needs the **SqlAlchemy** application as the database toolkit for Python. So you need to install **SqlAlchemy** with the following command line:

```
$ sudo apt-get install python-sqlalchemy
```

You can also use **pip** command to install **SqlAlchemy**. **Pip** is a tool for installing and managing Python packages.

```
$ sudo pip install sqlalchemy
```

There are other optional dependencies that are mostly used by modules and utilities. The following libraries are not strictly required, but you should have the libraries to guarantee Cuckoo Sandbox runs smoothly in your environment:

- **dpkt**: This library is highly recommended and is used for extracting information from PCAP files
- **jinja2**: This library is highly recommended and is used for rendering the HTML reports and the web interface
- **magic**: This library is optional and is used for identifying files' formats (otherwise use the **file** command-line utility)
- **ssdeep**: This library is also optional and is used for calculating fuzzy hash or files
- **pydeep**: This library is optional and is used for calculating **ssdeep** fuzzy hash of files

For More Information:
Getting Started with Automated Malware Analysis using Cuckoo Sandbox

- **pymongo**: This library is optional and is used for storing the results in a MongoDB database
- **yara** and **yara python**: This library is optional and is used for matching Yara signatures (use the svn version)
- **libvirt**: This library is optional and it uses the KVM machine manager
- **bottlepy**: This library is optional and it uses the web.py and api.py utilities
- **pefile**: This library is optional and is used for static analysis of PE32 binaries

All the packages can be installed by using a one-line `apt-get` command:

```bash
$ sudo apt-get install python-dpkt python-jinja2 python-magic python-pymongo python-libvirt python-bottle python-pefile ssdeep
```

Or you can install all the packages using `pip` package management (except `python-magic` and `python-libvirt`):

```bash
$ sudo pip install dpkt jinja2 pymongo bottle pefile
```

You have to install `pydeep` for `ssdeep` fuzzy hashes of samples; but before installing `pydeep`, we need to install some dependencies with the following command line:

- Build-essential
- Git
- Libpcre3
- Libpcre3-dev
- Libpcre++-dev
  ```bash
  $ sudo apt-get install build-essential git libpcre3 libpcre3-dev libpcre++-dev
  ```

Next, you have to clone `pydeep` from the git source (put `pydeep` in the `/opt` folder):

```bash
$ cd /opt
$ git clone https://github.com/kbandla/pydeep.git pydeep
$ cd /opt/pydeep/
python setup.py build
sudo python setup.py install
```

You will also need to install `yara` to categorize malware samples (put `yara` in `/opt` folder):

```bash
$ sudo apt-get install automake
$ cd /opt
```

For More Information:

$ svn checkout http://yara-project.googlecode.com/svn/trunk/yara
$ cd /opt/yara
$ sudo ln -s /usr/bin/aclocal-1.11 /usr/bin/aclocal-1.12
$ ./configure
$ make
$ sudo make install
$ cd yara-python
$ python setup.py build
$ sudo python setup.py install

You need to install **tcpdump** in order to dump network traffic which occurs during analysis:

$ sudo apt-get install tcpdump

If you want to run the **tcpdump**, you need root privileges; but since you don't want Cuckoo to run as root, you'll have to set specific Linux capabilities to the binary, as shown in the following command line:

$ sudo setcap cap_net_admin,cap_net_raw=eip /usr/sbin/tcpdump

You can verify the results of the last command with:

$ getcap /usr/sbin/tcpdump
/usr/sbin/tcpdump = cap_net_admin,cap_net_raw+eip

If you don't have **setcap** installed, you should install this library:

$ sudo apt-get install libcap2-bin

Otherwise (not recommended) run the following command line:

$ sudo chmod +s /usr/sbin/tcpdump

The **chmod +s** command means SUID bit. You add both user ID and group ID permission to a file. In this case, it is tcpdump. If you set the SUID bit "s" on **tcpdump**, then other users can run it and they will become the root for as long as the **tcpdump** process is executing. That is why this step is not recommended.

After you finish setting up the Host OS, you need to install and configure Cuckoo Sandbox in your Host OS.

---

For More Information:
**Setting up Cuckoo Sandbox in the Host OS**

In this section, you will set up Cuckoo Sandbox and configure it:

1. First, download Cuckoo from its website at [http://www.cuckoosandbox.org/download.html](http://www.cuckoosandbox.org/download.html).

   There are two ways to set Cuckoo up in your Host OS. You can either download the tarball file or you can clone from source using git.
   - If you want to clone from git source, you can do this step:
     ```bash
     $ git clone git://github.com/cuckoobox/cuckoo.git
     ```
   - If you want to download the tarball file from the website, you can visit the website and then press the Download Cuckoo! button.

2. After you're finished downloading the file, you have to extract the files into a folder:

   ```bash
   $ tar -zxvf cuckoo-current.tar.gz
   ```
3. Before configuring Cuckoo in your Host OS, you need to set up the Guest OS, as the Guest OS will be mentioned in Cuckoo’s configuration files (you will write down the Guest OS name in the configuration file). In this book, we will use VirtualBox Version 4.2.12 for 64 bit. You can download VirtualBox from the website [https://www.virtualbox.org/wiki/Downloads](https://www.virtualbox.org/wiki/Downloads).

In this book, we will use VirtualBox 4.2.12 for the Linux Host (If you can't find Version 4.2.12, you can use newer versions. But if you want to download Version 4.2.12, please go to [https://www.virtualbox.org/wiki/Download_Old_Builds_4_2](https://www.virtualbox.org/wiki/Download_Old_Builds_4_2)). There are several versions of VirtualBox for your Linux OS. We will download **Ubuntu 12.04 LTS ("Precise Pangolin")** AMD64 version (this one is for the 64-bit version if you are using a 32-bit version, you can choose to download i386).

Before setting up your Guest OS in VirtualBox, you need to pay attention to Vbox driver. You need to set up `vboxdrv` first before creating your Guest OS. In order to set up the `vboxdrv`, you need to install kernel headers of your Linux. The kernel headers will be required in compiling `vboxdrv`. If you want to be sure about your kernel version, you can use this command:

```
$ uname -a
```

For More Information:
You will see an output like this:

```
Linux digit-labs 3.5.0.17-generic #28-ubuntu SMP Tue Oct 9 19:31:23 UTC 2012 x86_64 x86_64 x86_64 GNU/Linux
```

It means you are using kernel Version 3.5.0.17, and you need to install the kernel headers using this command:

```
$ apt-get install linux-headers-3.5.0.17-generic
```

After you're finished installing the Linux headers, you can set up `vboxdrv` with the following command lines:

```
$ sudo /etc/init.d/vboxdrv setup
```

```
* Stopping VirtualBox kernel modules         [OK]
* Recompiling VirtualBox kernel modules       [OK]
* Starting VirtualBox kernel modules          [OK]
```

If all the output is **OK**, it means you can now set up the Guest OS.

### Preparing the Guest OS

The required specifications to set up the Guest OS are listed as follows:

- 1GB RAM memory
- 10 GB of hard disk space
- VDI format for the virtual disk
- Dynamically allocated storage
- Windows XP SP3

When you are installing the Guest OS, you have to create the Guest OS name for the Cuckoo Sandbox VirtualBox configuration file.

In the first step, we will create the guest OS. You can write down your guest OS name, and operating system type. Since we are using Windows XP as guest OS, you can choose Windows XP in the OS type and version.
Before you start your Guest OS in VirtualBox, you need to configure the network, sharing folder, and the installing of VirtualBox Guest Addition to improve its capabilities in the malware analysis process.

**Configuring the network**

Basically, VirtualBox has several types of network configuration that can be used by the Guest OS. Each type has a different capability based on your need, we can learn more about it in the VirtualBox website:

http://www.virtualbox.org/manual/ch06.html

Cuckoo is written in Python language, so you will need to install Python and other libraries as dependencies. Here is a website for you to download malware samples from, which will be used in this book:

http://www.cuckoosandboxbook.com/

You can download malware samples from the website. They will also provide you with some useful tools that can be downloaded from the same website. If you want to get additional information about this book, you can visit the aforementioned website, and put your comments there.

For More Information:

Getting Started with Automated Malware Analysis using Cuckoo Sandbox

Based on the explanation in the website, we should use the **Host-only networking** type, because it will isolate our Guest OS from the outside network. With this networking type, Host OS and Guest OS can interact with each other, but the Guest OS can "see" the outside network or internet.

1. In the VirtualBox main window, click on the **File** button and select **Preferences...**:

2. Choose **Network** in the sidebar to configure your host-only networking, and then click on the green icon that says **Add host-only network (Ins)** if you hover over it:

For More Information:

3. Click on the last icon on the side pane that says Edit Host-only Network to view your network configuration. If the DHCP server is not enabled, you need to manually configure your Guest OS IP Address but I suggest you leave it as it is:

For More Information:  
4. Next, you need to set up your Guest OS. Choose your Guest OS first in the sidebar, then click on the Settings option in the VirtualBox main window, and choose Network:

5. Go to the Adapter 1 tab and tick the option Enable Network Adapter. In the Attached to drop-down menu, you have to choose Host-only Adapter and in the Name drop-down menu choose vboxnet0 (network adapter name is based on what you have created).

6. After finishing your configuration for the Guest OS, you can start your Guest OS into the beginning installation process.

I assume that you have already finished your Guest OS installation process and logged in to your Guest OS. You will need to manually configure your Guest OS, as the DHCP server is not enabled in the host-only network configuration. Give your OS IP address with the same network segment as the Host OS. In this case, if you leave the host-only configuration as it is, the Host OS and Guest OS IP addresses will be set as 192.168.56.1 and 192.168.56.101, respectively.

Try to ping each other to make sure that the Host OS and Guest OS is already connected.

For More Information:
Chapter 1

Setting up a shared folder between Host OS and Guest OS

1. In the Guest OS main window, click on the Devices option and select Shared Folders... as shown in the following screenshot:

![Shared Folders Screenshot]

2. Then click on the green icon at the top-right corner of your window that says Add Shared Folder (Ins):

![Add Shared Folder Screenshot]

3. Choose the folder (in your Host OS) that you want to be shared with your Guest OS in the **Folder Path** (for example `/home/username/Downloads` or we can make our own folder somewhere else).

4. Give the shared folder a name (by default your computer will give a shared folder name, you can change the folder name as you wish), and tick the sharing options according to your choice:

5. Now in your Windows Guest OS, click on the Start menu, right-click on **My Computer**, and choose **Map network drive**....

6. Select the drive you want from the drop-down menu.

7. In the **Folder** text field, fill it in with `\vboxsrv\shares` (*shares* is the shared folder name in the previous screenshot).

8. Go to **Computer** or Windows Explorer, and you will see the shared folder.

---

For More Information:
9. Now, to configure your Guest OS you have to:

   1. Install Python for Windows. You can download the software at http://python.org/download/.

   2. Install PIL (Python Imaging Library) Python module to created desktop screenshots. This software is available at http://www.pythonware.com/products/pil/.

   3. Turn off automatic Windows updates.

   4. Turn off Windows firewall.

   5. Install third-party applications (Microsoft Office 2003/2007, Acrobat Reader 9.5, Mozilla Firefox 3.6, and so on) at http://www.oldapps.com/. This step is optional.

10. Next, copy the Python agent to our Windows shared folder using this command line on the Host OS:

    `$ cp /home/digit/cuckoo/agent/agent.py /home/digit/cuckoo/shares/

11. From your Windows Guest OS, copy the agent.py file into C:/Python27 folder.

12. Rename the agent.py file to agent.pyw.

   PYW files run the script without invoking the console window, especially if your program is GUI based. If you double-click the agent.py file, a command prompt window will appear on your desktop. If you rename the file to a .pyw file, there will be no pop-up window appearing on your desktop. It is similar to a background process in Linux.

13. To always run the agent.pyw file in startup process, you need to put it in the Startup folder in the following paths:

    For Windows XP go to C:\Document and settings\username\Start Menu\Programs\Startup.

    For Windows 7 go to C:\Users\iKONspirasi\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup.

14. After executing agent.pyw, a new socket will be listening on the 0.0.0.0:8000 port. To check it, you should run this command in the command prompt:

    C:\>netstat -aon

For More Information:
As you can see in the screenshot below:

15. You also need to configure Host OS IP forwarding and filtering rules using `iptables`:

   ```bash
   $ 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
   ```

16. The next step is the configuration of Cuckoo Sandbox.
Creating a user

You can either run Cuckoo from your own user or create a new one dedicated just to your Sandbox setup. We recommend you to create a specific user for your Cuckoo Sandbox environment. Make sure that the user that runs Cuckoo is the same user that you will use to create and run the virtual machines, otherwise Cuckoo will not be able to identify and launch them. Just run the following command line in terminal:

$ sudo adduser cuckoo

If you're using VirtualBox, make sure the new user belongs to the `vboxusers` group (or the group you used to run VirtualBox):

$ sudo usermod -G vboxusers cuckoo

If you're using KVM or any other `libvirt`-based module, make sure the new user belongs to the `libvirtd` group (or the group your Linux distributor uses to run `libvirt`):

$ sudo usermod -G libvirtd cuckoo

Now it's time for the best part, let's install and configure Cuckoo Sandbox.

Installing Cuckoo Sandbox

Extract or checkout your copy of Cuckoo to a path of your choice and you're ready to go. For example, we can put it in the `~/home/username/cuckoo` path.

First things first, we need to configure Cuckoo's configuration files, which consist of the following main files:

- `cuckoo.conf`: This configuration file contains information about the general behavior and analysis options in Cuckoo Sandbox.
- `<machinemanager>.conf`: This file holds the information about your virtual machine configuration. (Depends on the name of virtualization that we used.)
- `processing.conf`: This file is used for enabling and configuring the processing of modules.
- `reporting.conf`: This file contains information about reporting methodologies.

The aforementioned `.conf` files are described in detail in the following sections.

For More Information:

cuckoo.conf

This file contains the basic and general configuration information of Cuckoo. For example, you can ask Cuckoo to check the newest version when it is being executed. If you use this feature, Cuckoo will download the newest version, and you can store the old version or delete it. It defines in the version_check on the cuckoo.conf file. You can describe your virtualization method in the cuckoo.conf file. For example, if you are using VirtualBox, you can write in machine_manager = virtualbox, or if you are using VMware, you can change this line to vmware.

You can also write down the Host OS IP address and port number that will be used by Cuckoo Sandbox. By default, the IP address is set as 192.168.56.1 (because we are using host-only networking method), and the default port is 2042. (Don't forget to define your networking interface.) We have defined the interface for Cuckoo, vboxnet0 (look at the discussion about VirtualBox configuration in the Configure the network section).

<machinemanager>.conf

Machine managers are the modules that define how Cuckoo will interact with your virtualization tools. In cuckoo.conf, you will write down your virtualization software. If you use VirtualBox, the <machinemanager>.conf will refer to the virtualbox.conf configuration. If you use VMware, <machinemanager>.conf will refer to the vmware.conf file.

In this book we use VirtualBox, so you just need to pay attention to the virtualbox.conf file. You can edit this file based on your need. For example, if you want to run VirtualBox in GUI, you should edit the mode and set it as gui. If you feel comfortable using VirtualBox with command lines, then you should write down mode = headless in virtualbox.conf.

Remember in the Guest OS installation, I mentioned that you need to pay attention while naming the Guest OS because you will edit the Guest OS name in this configuration. Therefore, in the [cuckoo1] section, you can specify the Guest OS name. If you give your Guest OS name cuckoo1, you can edit label as label = cuckoo1 (don't forget we created the Guest OS name Windows-cuckoo).

Since we are using Windows XP as the Guest OS, you have to define the platform section as windows:

platform = windows

Don't forget to write down the Guest OS IP address. We are using host-only networking, by default the first OS in guest system will be given the IP address 192.168.56.101.
processing.conf

This configuration file will allow you to enable, disable, and configure all the processing modules.

Basically, you do not need to make any changes to the default configuration in this file. But you can add your own VirusTotal API key in it. If you don't have a VirusTotal account yet and want to have one, just create an account in VirusTotal's website at https://www.virustotal.com/en/, and put the key in this line:

```plaintext
# Add your VirusTotal API key here. The default API key, kindly
# provided by the VirusTotal team, should enable you with a
# sufficient throughput and while being shared with all our users,
# it should not affect your use.
key = a0283a2c3d55728300d064874239b5346fb991317e8449fe43c902879d758088
```

reporting.conf

The conf/reporting.conf file contains information on automated reports generation. This file contains information about the methodologies or kinds of reporting that you want to use after the completion of the analysis process. You can either disable or enable the reporting method.

After you finish configuring your Cuckoo Sandbox environment, you can test your first malware analysis process.

The virtual machine is now ready to test malware, but for the first time you need to create a snapshot file using this command:

```bash
$ vboxmanage snapshot "Windows-cuckoo" take "Windows-cuckooSnap01" --pause
```

The following commands are used to restore the snapshot:

```bash
$ vboxmanage controlvm "Windows-cuckoo" poweroff
$ vboxmanage snapshot "Windows-cuckoo" restorecurrent
$ vboxheadless --startvm "Windows-cuckoo"
```

For More Information:
Getting Started with Automated Malware Analysis using Cuckoo Sandbox

The snapshot of the Guest OS is the most important part for the process of analyzing malware using Cuckoo Sandbox. Make sure everything is set and ready to analyze malware and carry out the following steps to perform the analysis:

1. To start your Cuckoo Sandbox, you need to run:

   $ ./cuckoo.py

   The output from your terminal will be something like the following screenshot:

   ![Cuckoo Sandbox Output](image)

For More Information:
2. Cuckoo is now running and waiting for analysis. You can submit sample malware or malicious URLs. You have to change the directory to `/cuckoo/utils/` and then use the `submit.py` file to perform a malware analysis:

```
root@digit-lab:~/cuckoo/utils

digit@digit-lab:~/cuckoo/utils

digit@digit-lab:~/cuckoo/utils$ ./submit.py /home/digit/malware_sample/windows\live\Messenger.exe
Success: File `/home/digit/malware_sample/Windows Live Messenger.exe` added as task with ID 3

digit@digit-lab:~/cuckoo/utils
```

For More Information:
Then, the output from Cuckoo's main window will be something like the following screenshot:
Summary

Now, you have successfully prepared the Host OS and Guest OS in the VirtualBox and then installed Cuckoo Sandbox. It is important to make sure that all the dependencies that are needed in the Host OS along with \texttt{pydeep} and \texttt{yara} are present. For the Guest OS, always turn off the defensive parameter and Windows firewall and use any software that the malware often use to interact with, for example, Adobe Reader 9.5, Internet Explorer 6, Microsoft Office 2003, and so on.

Always set your configuration in \texttt{<machinemanager>.conf} in exactly the same way as it is in the virtualization software you are using. For example, if you are using KVM, you have to set \texttt{kvm} in \texttt{machinemanager.conf}. Since we are using VirtualBox, you have to set \texttt{virtualbox} in the configuration. You have to be careful at the time of inserting the name of the Guest OS in VirtualBox to \texttt{cuckoo.conf} configuration file. For example, if you create a Guest OS named \texttt{cuckoo1}, you have to write down \texttt{cuckoo1} in the \texttt{cuckoo.conf} configuration file. The most important part of all is not to forget to make a backup of the whole system and configurations.

In the next chapter, we will continue learning about Cuckoo Sandbox's features, such as analyzing PDF files, URLs, and binary files, Memory Forensic using Cuckoo Sandbox (using the Memory dump feature), and additional Memory Forensic using Volatility.

For More Information:
Where to buy this book


Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.