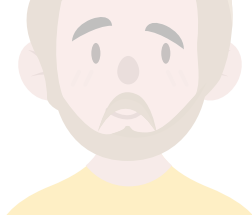
Cuckoo Sandbox Setup for People in a Hurry

Written by

2019-07-03



Alwin Peppels and Ricardo van Zutphen

Introduction

This blog post aims to provide a straightforward install for Cuckoo that is easy to follow, using as much of the built-in automation as possible.

The first part of the guide will be setting up a basic install that should be sufficient for a single-user setup with a low number of machines.

For people who want to run Cuckoo on more resources, or run a production web interface for multiple people, some different dependencies have to be used. When using parallel results processing and multiple VMs, SQLite should be replaced with a DBMS such as Postgres, which handles multiple processes more reliably. The development web server can be replaced with uWSGI + nginx. This is discussed in the *Cuckoo Web Interface* section of this blog post.

This setup guide is tested to work on a clean install of Ubuntu 18.04 LTS, on other distros your mileage may vary.

We will install VirtualBox, set up a virtualenv for Python and create a low-privilege user for cuckoo. Then create the analysis VMs using VMCloak to automatically install Windows 7, software, and to create snapshots, after which we will use the built-in 'cuckoo machine' command to add them to the configuration.

First of all, let's get our package manager up-to-date, install virtualenv, and make sure we have the right core tools installed:

```
sudo apt-get update
sudo apt-get -y install python virtualenv python-pip python-dev build-esse
```

Secondly, we will create a new user to run Cuckoo under. Running Cuckoo under a separate user is important for a safe setup, we're dealing with malware; if a vulnerability is found and exploited it will be harder to compromise the whole machine from a low-privileged user:

```
sudo adduser --disabled-password --gecos "" cuckoo
```

The user should be able to create network dumps during Cuckoo analyses, so we give it permission to do so:

```
sudo groupadd pcap
sudo usermod -a -G pcap cuckoo
sudo chgrp pcap /usr/sbin/tcpdump
sudo setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump
```

One more step before we start, we need a Windows 7 ISO. Let's download one that we found and know that it works. After downloading, we have to mount the ISO to be used at a later step:

```
wget https://cuckoo.sh/win7ultimate.iso
mkdir /mnt/win7
sudo mount -o ro,loop win7ultimate.iso /mnt/win7
```

Installing VirtualBox

We will install VirtualBox from the VirtualBox repository, as this allows for easier upgrading to newer releases. It is important to install updates for the virtualization layer, as they might include security updates. Especially for VirtualBox, which has seen numerous 0day vulnerabilities in the last years, it's important to run the latest version of VirtualBox 5 or VirtualBox 6 (at the time of writing, this would be either 5.2.30 or 6.0.8).

First, we add the repository keys:

```
wget -q https://www.virtualbox.org/download/oracle_vbox_2016.asc -O- | suc
wget -q https://www.virtualbox.org/download/oracle_vbox.asc -O- | sudo apt
```

Adding the VirtualBox repository:

```
sudo add-apt-repository "deb [arch=amd64] http://download.virtualbox.org/v
```

Let's update the package list and download VirtualBox. Doing so will allow us to install the latest version of VirtualBox 5.2 or 6.0, whichever is preferable. After the installation, we add the cuckoo user to the vboxusers group.

```
sudo apt-get update
sudo apt-get install virtualbox-5.2
sudo usermod -a -G vboxusers cuckoo
```

Or for VirtualBox 6.0:

```
sudo apt-get update
sudo apt-get install virtualbox-6.0
sudo usermod -a -G vboxusers cuckoo
```

Cuckoo and VMCloak installation

Before we install Cuckoo and VMCloak, the installation of multiple packages is required. These are dependencies VMCloak or Cuckoo require to function.

VMCloak and Cuckoo required packages:

```
sudo apt-get -y install build-essential libssl-dev libffi-dev python-dev g
sudo apt-get -y install zlib1g-dev libjpeg-dev
sudo apt-get -y install python-pip python-virtualenv python-setuptools sw
```

Now that the dependencies have been installed, we can install Cuckoo and VMCloak. Start by switching to the cuckoo user and creating a new virtualenv:

```
sudo su cuckoo
virtualenv ~/cuckoo
. ~/cuckoo/bin/activate
```

The virtualenv will allow us to install dependencies within our home directory and to prevent interference with other, globally installed, Python packages.

Install both VMCloak and Cuckoo Sandbox within the same virtualenv:

```
pip install -U cuckoo vmcloak
```

Automatic VM creation

Manually installing Windows, required software, editing registry keys, etc is a lot of work. Luckily, we don't have to do that because we will use VMCloak!

First, start by defining and instantiating a VirtualBox Host-Only network adapter for the VMs to use:

```
vmcloak-vboxnet0
```

To set up a Windows VM, we will use the ISO that we mounted earlier. VMCloak will try /mnt/win7 (and /mnt/win7x64) by default. The following step will create the VM and automatically install Windows. This step will take approximately 15 to 20 minutes. A Cuckoo analysis VM should have at least 2GB of memory and preferably two or more CPU cores.

The syntax of the command we will use is: `vmcloak init <os flag> <vmname> <options>`:

```
vmcloak init --verbose --win7x64 win7x64base --cpus 2 --ramsize 2048
```

After VMCloak is finished we can start installing software that should be present in VM snapshots. When we have created snapshots of an image, it can no longer be changed, therefore we clone the cleanly installed base image so we can install software on the clone and snapshot that:

```
vmcloak clone win7x64base win7x64cuckoo
```

VMCloak supports the installation of multiple software packages. A full list of supported packages and versions can be listed:

```
vmcloak list deps
```

A software package can be installed with the following syntax: `vmcloak install <image name> <package>`. A specific version or a serialkey can be provided by adding: `package.version=X` or

`package.serialkey=X`. If no version is selected, the default version will be picked. We will be installing some basic software packages:

```
vmcloak install win7x64cuckoo adobepdf pillow dotnet java flash vcredist v
```

Optional step: installing Internet Explorer 11:

```
vmcloak install win7x64cuckoo ie11
```

Optional step: Installing a Microsoft Office version so that Office document can be analyzed. Office 2007 is most likely to work, some builds of higher versions of Office sometimes cause issues with the Cuckoo Monitor.

```
vmcloak install win7x64cuckoo office office.version=2007 office.isopath=/p
```

When finished with installing software packages, we can create the VM snapshots. VMCloak will register a VirtualBox VM for each snapshot created. After snapshotting, it is no longer possible to change the image. The syntax of the snapshot command is:

```
vmcloak snapshot <options> <image name> <vmname> <ip to use>
```

Using the `--count` parameter we can create multiple snapshots at once. Let's create four:

```
vmcloak snapshot --count 4 win7x64cuckoo 192.168.56.101
```

This command will create VMs `win7x64cuckoo1-4` with IPs `192.168.56.101-104`.

After VMCloak is finished, the VMs can be listed using:

```
vmcloak list vms
```

Configuring Cuckoo

Cuckoo loads its configuration files, signatures, and other user-changeable files from its *Cuckoo Working Directory* (CWD). By default this will be `$USERHOME/.cuckoo`. Before we can use Cuckoo, we first have to create this directory:

```
cuckoo init
```

If a custom CWD is desired, the `--cwd <path>` option must be used upon creation and must be supplied when using Cuckoo commands, e.g.:

```
cuckoo --cwd /tmp/cuckoo init
```

Postgres as DBMS

You can skip this section and jump to [Adding VMs][Adding VMs] if you are not using more than a couple of analysis VMs and will not be using Cuckoo processing instances.

First, start by installing Postgres if it is not already installed:

```
sudo apt-get postgresql postgresql-contrib
```

Then we install the Postgres database driver for Cuckoo:

```
pip install psycopg2
```

Now let's create a user and database for Cuckoo to use:

```
sudo -u postgres psql
CREATE DATABASE cuckoo;
CREATE USER cuckoo WITH ENCRYPTED PASSWORD 'password';
GRANT ALL PRIVILEGES ON DATABASE cuckoo TO cuckoo;
\q
```

After that, we have to tell Cuckoo to use Postgres instead of SQLite. Open the `$CWD/conf/cuckoo.conf` file and find the `[database]` section. Change the `connection =` line to:

```
connection = postgresql://cuckoo:password@localhost/cuckoo
```

Adding VMs

We are using VirtualBox in our setup, this is the default Machinery module that Cuckoo uses. We have to remove some default settings from its configuration file `virtualbox.conf`.

All Cuckoo configuration files can be found at `$CWD/conf/`. Open `$CWD/conf/virtualbox.conf` and remove the entries in the `machines = cuckoo1` line.

Time to add the created VMs to Cuckoo. We will use the `cuckoo machine --add <vm name> <ip>` to tell Cuckoo to add the

machine to its configuration. This has to be done for each machine, so let's make life easier and use `vmcloak list vms`:

```
while read -r vm ip; do cuckoo machine --add $vm $ip; done < <(vmcloak lis
```

To install the Cuckoo signatures and latest monitor, we run the following command:

```
cuckoo community --force
```

Network configuration

Next, providing analysis VMs with an internet connection.

Internet on the VMs is not required, however, not having an internet connection restricts malware from retrieving data such payloads and instructions. This can affect the accuracy of the analysis results.

Configuring traffic forwarding can be done globally or on a per-analysis basis. The latter requires more steps, so we will start with the global forwarding rules.

First, switch to an account that has root privileges and enable forwarding. Do this for the `vboxnet0` interface and the outgoing interface. We will be using `eth0` as a dummy value here. Replace it with your outgoing interface (which can be identified through, e.g., `ifconfig`):

```
sudo sysctl -w net.ipv4.conf.vboxnet0.forwarding=1
sudo sysctl -w net.ipv4.conf.eth0.forwarding=1
```

Global forwarding rules

To enable global routing for all VMs connected to the vboxnet0 interface, use the following rules:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -s 192.168.56.0/24 -j MASQUERA
sudo iptables -P FORWARD DROP
sudo iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -s 192.168.56.0/24 -j ACCEPT
```

Per-analysis routing - Advanced

Cuckoo allows for multiple types of per-analysis routes, among which are ‘internet’ and ‘none’. The route can be specified whens

submitting an analysis.

To accomplish per-analysis routing, Cuckoo uses the Cuckoo Rooter. This is a separate process that runs with root privileges and is able to execute pre-defined commands.

We start by creating a Cuckoo Rooter process. This creates a UNIX socket owned by root and giving the 'cuckoo' group permission to use it. **Do not run these commands with sudo**. The `--sudo` flag will take care of this:

```
cuckoo rooter --sudo --group cuckoo
```

Since Cuckoo is installed in a virtualenv, and the Cuckoo user should not have root privileges, we can't do the following from a root privileged user:

```
/home/cuckoo/cuckoo/bin/cuckoo rooter --sudo --group cuckoo
```

Next, we have to edit `$CWD/conf/routing.conf` to tell Cuckoo what our outgoing interface is. Open routing.conf and change `internet = none` to `internet = eth0`.

This config file also contains a `route = none` line. This is the default routing it will use. This can be changed to `internet` to give each analysis internet access, unless a different routing option is provided upon sample submission. A command line submission example: `cuckoo submit <file path> --options "route=internet"`.

Note: if the routing configuration is changed to support per-analysis routing, Cuckoo requires the Cuckoo Rooter to be running and will not start otherwise.

The Cuckoo Web Interface

The Cuckoo web interface can be used to submit new tasks and view analysis results. It requires MongoDB to be installed and enabled in the `reporting.conf`.

If MongoDB is not installed, start by installing it:

```
sudo apt-get install mongodb
```

The web interface can be used by starting the built-in development server or by configuring a uWSGI + NGINX setup for a production environment.

Results will only show if they have been processed after MongoDB has been enabled. As Cuckoo will not store them in there by default.

We start by opening `$CWD/conf/reporting.conf` and find the `[MongoDB]` section. Change `enabled = no` to `enabled = yes`. No further configuration changes are required, unless your MongoDB setup requires a user, runs on a non-standard port, or runs remotely.

The built-in web server

This server should not be used for production environments. It is a development server. It can be used for small setups, but should not be exposed to the internet.

The server can be started by running:

```
cuckoo web --host 127.0.0.1 --port 8080
```

We can now submit tasks and view results in the web interface. Cuckoo must be running for analyses to start, otherwise tasks will remain on the *pending* status.

Using uWSGI and nginx - Advanced

Cuckoo web can be set up to be served by nginx. This greatly increases the maximum amount of users a Cuckoo web interface can have at one time, and allows for the usage of TLS.

To run Cuckoo web with uWSGI, we must first install the `uwsgi` Python dependency in the virtualenv where we installed Cuckoo:

```
pip install uwsgi
```

After this, ensure the following packages are installed:

```
sudo apt-get install uwsgi uwsgi-plugin-python nginx
```

Cuckoo can generate the configurating files for uWSGI and nginx. First, let's set up uWSGI:

```
cuckoo web --uwsgi > cuckoo-web.ini
sudo cp cuckoo-web.ini /etc/uwsgi/apps-available/cuckoo-web.ini
sudo ln -s /etc/uwsgi/apps-available/cuckoo-web.ini /etc/uwsgi/apps-enable
```

Ensure that the `www-data` user can read the Cuckoo web files by adding it to the cuckoo group:

```
sudo adduser www-data cuckoo
sudo systemctl restart uwsgi
```

Second, let's set up nginx. The generated nginx configuration contains values that you might want to change, such as the listening IP and port:

```
cuckoo web --nginx > cuckoo-web.conf
sudo cp cuckoo-web.conf /etc/nginx/sites-available/cuckoo-web.conf
sudo ln -s /etc/nginx/sites-available/cuckoo-web.conf /etc/nginx/sites-ena
sudo systemctl restart nginx
```

The web interface should be served by nginx now.

Starting Cuckoo

When Cuckoo starts for the first time, it creates the required database tables in the configured database type. By default this is SQLite.

Let's start Cuckoo now. If we enable debug mode, we can see all the steps it takes along the way of starting:

```
cuckoo --debug
```

After starting, it will either start analyzing submitted files or display *waiting for analyses...*

Cuckoo processing instances - Advanced

When using 4+ VMs, it is recommended to use Cuckoo processing instances. These are separate processes that perform all results processing and reporting.

Normally, the processing is part of the analysis flow and performed by the main Cuckoo process. Having a lot VMs can cause a large backlog of analysis results to be processed. By moving this process to multiple dedicated processes, this problem is prevented. For more information about this, see the processing instance documentation.

Before starting processing instances, we must first configure Postgres (see [Postgres as DBMS][Postgres as DBMS].) and disable processing of results in the main Cuckoo process. Open the

`cuckoo.conf` and change `process_results = yes` to `process_results = no`.

We can now start one or more processing instances. The syntax of the command is `cuckoo process <instance name>`:

```
cuckoo process p1
```

The instance(s) will process tasks that reach the *completed* status.

Starting Cuckoo in the background - Advanced

It might be desirable to run Cuckoo and supporting processes in the background. This is possible with `supervisord`. Cuckoo generates a `supervisord.conf` file for this in the CWD. This configuration starts Cuckoo and 4 processing instances, no other Cuckoo components. This configuration assumes the required changes in `cuckoo.conf` has already been made. For the full documentation on this, see the cuckoo supervisord docs.

To start setting up Cuckoo with supervisord, let's start by installing supervisord:

```
sudo apt-get install supervisord
```

We start by telling supervisord what configuration to use:

```
supervisord -c /home/cuckoo/.cuckoo/supervisord.conf
```

Now we can now start Cuckoo as a background process and stop it when we want:

```
supervisorctl start cuckoo:
supervisorctl stop cuckoo:
```

Conclusion

After following the above steps, one may now enjoy a fully functional Cuckoo Sandbox setup with multiple VMs, network routing capabilities, the Cuckoo Web Interface, and potentially more goodies. Don't forget to check out the extensive Cuckoo Sandbox documentation and let us know if there are questions and/or feedback.

Looking for Cuckoo Sandbox support? Check our solutions.

You may also like:

Release of Cuckoo-compatible

Read

Jobs

Read

General Family Updates

Read

New Integrations, Updated DLL

Read

Home

About

Triage

Cuckoo

Blog

Jobs

Contact us

Contact

Want to know more?

Get in touch!

Hatching International B.V.
Korte Hogendijk 4
1506 MA Zaandam
The Netherlands

Call us: +31 75 207 30 90
Email us: info@hatching.io

IBAN: NL52 INGB 0006 9672 73
BIC: INGBNL2A
ING Groep N.V.
Amsterdam-Zuidoost

REG: 64146707
VAT: NL855541891B01