An Overview of Cuckoo Sandbox

Jeff Malavasi
Dept. of Computing Security
Rochester Institute of Technology
Email: jm3378@rit.edu

I. Introduction

Cuckoo Sandbox is an open-source automated malware analysis tool. It works by creating a framework to spawn virtual machines, run a specific set of tasks, and output the data to a standardized report. It provides a high degree of separation by routing all traffic through the host machine. There are many different plugins that have been created which provide additional functionality such as advanced reporting and sending the file to VirusTotal for additional analysis. [1] The tool should be used when an unknown binary needs to be safely examined without risking the potential for spread. In this report, we analyze five malicious samples with the default configuration, before additionally exploring PDF reports, the Cuckoo web module, and memory analysis.

II. Methods

A. Improved Reporting

Cuckoo Sandbox creates an HTML and JSON report by default. However, these reports have a few limitations. For example, the HTML report does not include the VirusTotal analysis and is difficult to distribute. We can improve this by updating the reporting.conf file.

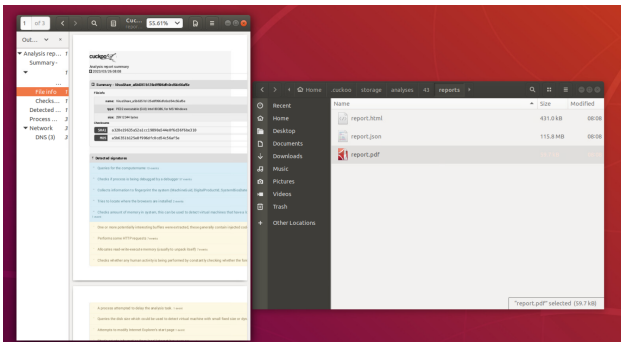
```
# What kind of data to show apart from default.
# Show virustotal hits.
show_virustotal = yes

# Show matched cuckoo signatures.
show_signatures = yes

# Show collected URL-s by signature "network_http".
show_urls = yes

# Hide filename and create hash of it
hash_filename = no
# Hide URL and create hash of it
hash_url = no
```

```
[singlefile]
# Enable creation of report.html and/or report.pdf?
enabled = yes
# Enable creation of report.html?
html = yes
# Enable creation of report.pdf?
pdf = yes
```

After changing pdf, show_virustotal, and show_urls to yes, we can reprocess the report for a specific task by running the below code and replacing x with the task id.

```
sudo cuckoo process -r x
```

After this completes, the analysis folder will now have a PDF in it, in addition to the existing report types.



B. Cuckoo Web

The Cuckoo web module utilizes MongoDB in order to generate a web server than can be used to submit samples as well as view reports. This makes it a bit easy to change configuration settings on the fly, without needing to update configuration files. In order to install MongoDB, run the following command:

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

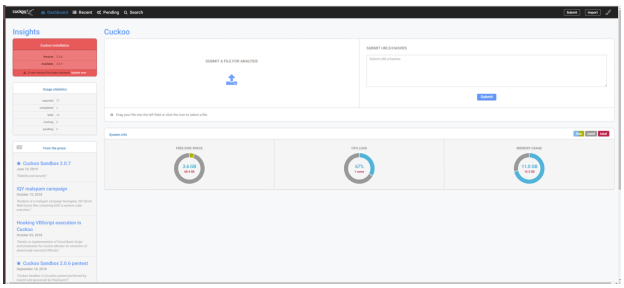
Next update the reporting.conf file and change enabled to yes in the MongoDB stanza.

```
[mongodb]
enabled = yes
host = 127.0.0.1
port = 27017
db = cuckoo
store_memdump = yes
paginate = 100
# MongoDB authentication (optional).
username =
password =
```

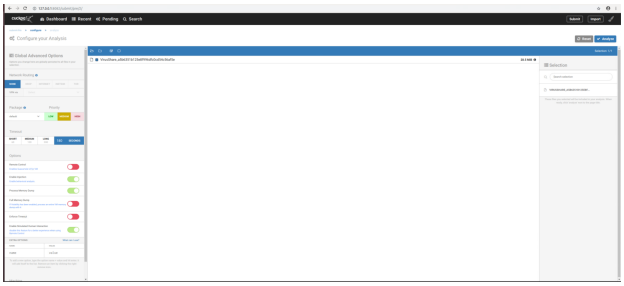
The web server can now be started by running the following command:

```
sudo cuckoo web --host 127.0.0.1 --port 8082
```

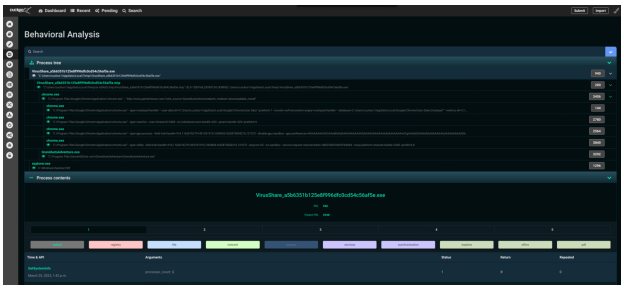
The web app can be accessed by navigating to http://localhost:8082. At the top right we can access the submission menu, which can be used to submit files, hashes or even URLs to be analyzed. Once all the files have been submitted, Cuckoo can be started normally. The results are then stored in the recent tab and provide more detail than the generated HTML report.



Additionally, during the submission phase, the web interface allows for customization of the specific task. This can include enabling memory dumps for example.

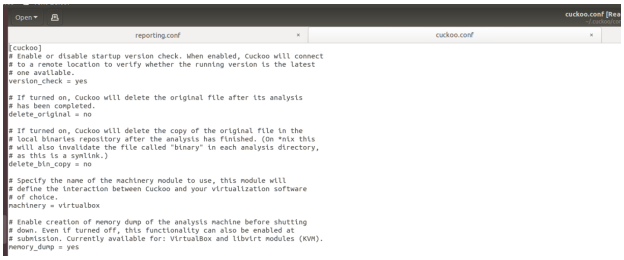


Below is an example analysis of one of the five malicious files:

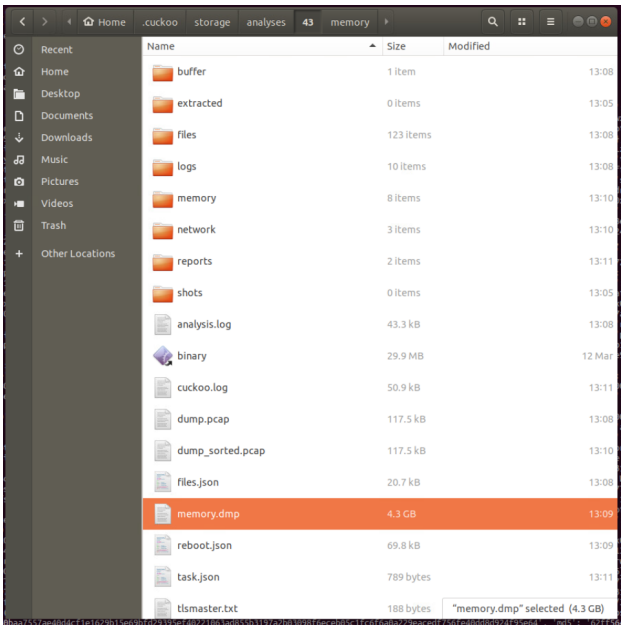


C. Memory Analysis

In addition to analyzing processes, network and file activity, Cuckoo can also save memory dumps which can later be used by applications such as Volatility. Additionally, Cuckoo can be configured to take a baseline sample of a VM without a sample in order to reduce the noise. Memory analysis can provide additional information about what the sample is doing, especially in the case of a buffer overflow exploit. In order to setup memory analysis, we must update cuckoo.conf and processing.conf. In the processing configuration, we have to enable baseline, and memory. For the main configuration, we have to enable memory_dump.



Next, we need to run a baseline task. This can be done by calling Cuckoo submit with the –baseline parameter. Once this is complete, we can rerun a malicious sample and will end up with a memory dump in the analysis folder.



III. Results

After analyzing all five malicious samples, Cuckoo generated reports for each sample. The HTML report provides basic information, such as behavioral signatures and the file hash. The following signatures were noted on the provided files:

VirusShare_9bd5b206ec96551f42279bc01b9061dd

- Create a windows hook that monitors keyboard input (keylogger).
- Appends a known multi-family ransomware file extension to files that have been encrypted
- Deletes a large number of files from the system indicative of ransomware, wiper malware or system destruction

VirusShare_3a4a5d40fd305a10063345ea96164ace

- Checks for the presence of known devices from debuggers and forensic tools
- Detects VMWARE through the in instruction feature
- Executed a process and injected code into it, probably while unpacking

VirusShare_2d839669a87a277d46f70c175ccbbd12

- Collects information about installed applications
- Installs itself for autorun at Windows startup
- Drops a binary and executes it

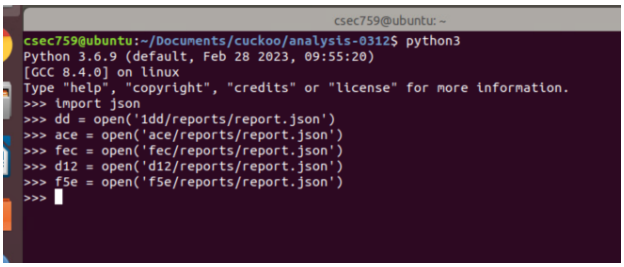
VirusShare_a5b6351b125e8f996dfc0cd54c56af5e

- Deletes executed files from disk
- One or more martian processes was created
- Allocates execute permission to another process indicative of possible code injection

VirusShare_e95ac5d39be9eca7af0ee72a62aeffec

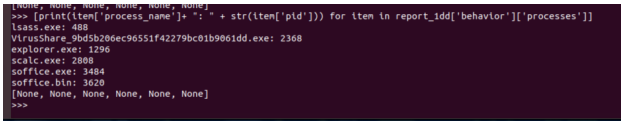
- Installs itself for autorun at Windows startup
- Attempts to detect Cuckoo Sandbox through the presence of a file
- Creates a suspicious process

In addition to the basic analysis provided by the HTML report, the JSON report was also investigated using the JSON Python library. For brevity, we will only report our findings on VirusShare_9bd5b206ec96551f42279bc01b9061dd.



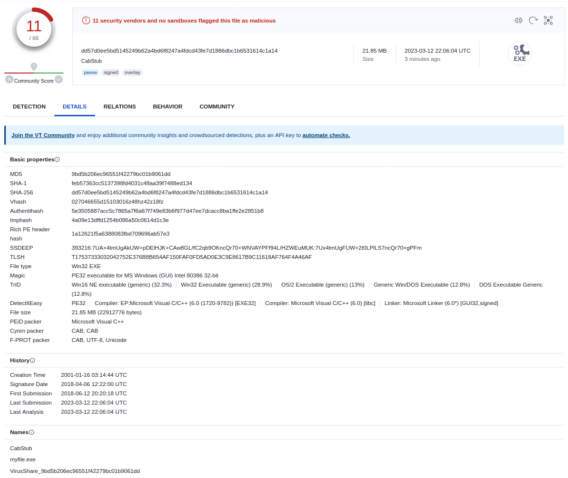
The process IDs spawned by the malware can be found with the following:

```
[ print(process['process_name'] + ": " +
str(process['pid'])) for process in
report['behavior']['processes'] ]
```



The virus total results can be found with the following:

```
report['behavior']['virustotal']
```



We can see that this file has been submitted with the following 3 names:

- VirusShare_9bd5b206ec96551f42279bc01b9061dd
- CabStub
- myfile.exe

The DLLS loaded made by each process be found with the following

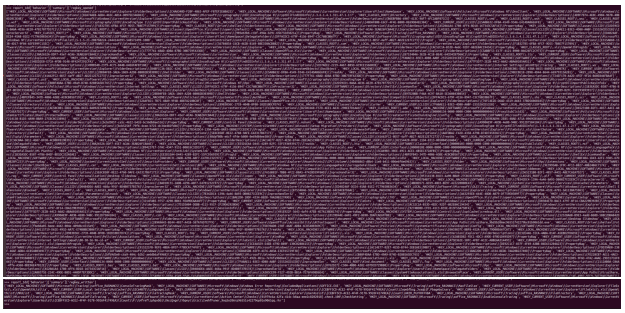
```
report['behavior']['summary']['dll_loaded']
```

The API calls made by each process be found with the following

```
report['behavior']['apistats']
```

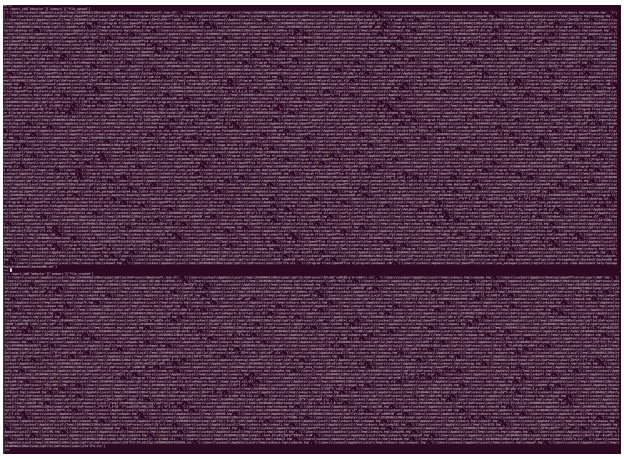
The registry operations performed by the malware be found with the following

```
report['behavior']['summary']['regkey_opened']
report['behavior']['summary']['regkey_written']
```



The file operations performed by the malware be found with the following

```
report['behavior']['summary']['file_opened']
report['behavior']['summary']['file_written']
```



IV. Conclusion

Cuckoo Sandbox provides an extremely flexible framework to automatically analyze unknown files in a safe and contained environment. From the samples we tested, we found indicators of spyware, ransomware and other types of malicious activity.

References

[1] "Cuckoo sandbox book - cuckoo sandbox v2.0.7 book," 2023. [Online]. Available: https://cuckoo.readthedocs.io/en/latest/