Konfigürasyon

Cuckoo, birkaç ana yapılandırma dosyasına dayanır:

- cuckoo.conf: genel davranış ve analiz seçeneklerini yapılandırmak için.
- auxiliary.conf: yardımcı modülleri etkinleştirmek ve yapılandırmak için.
- <machinery>.conf: sanallaştırma yazılımınızın seçeneklerini tanımlamak için (cuckoo.conf'da seçtiğiniz makine modülünün aynı adını taşıyan dosya).
- memory.conf: Volatility yapılandırması.
- processing.conf: işleme modüllerini etkinleştirmek ve yapılandırmak için.
- reporting.conf: rapor formatlarını etkinleştirmek veya devre dışı bırakmak için.

Cuckoo'yu çalıştırmak için en azından cuckoo.conf ve <machinery>.conf dosyalarını düzenlemeniz gerekmektedir.

cuckoo.conf

Düzenlenmesi gereken ilk dosya `$CWD/conf/cuckoo.conf`'dur. `$CWD`'den bahsederken Cuckoo Working Directory'ye atıfta bulunacağız. cuckoo.conf dosyası, Cuckoo'yu başlatmadan önce kontrol etmek veya en azından kendinizi tanımak isteyeceğiniz genel yapılandırma seçeneklerini içerir.

Dosya büyük ölçüde açıklamalı ve kendini açıklayıcıdır, ancak bazı seçenekler özellikle dikkatinizi çekebilir:

- `[cuckoo]` içindeki `machinery`: Bu seçenek, Cuckoo'nun analiz makinelerinizle etkileşim kurmak için hangi Machinery modülünü kullanmasını istediğinizi tanımlar. Değer, modül adının uzantısı olmadan (örneğin, virtualbox veya vmware) olmalıdır.
- `[resultserver]` içindeki `ip` ve `port`: Bu, Cuckoo'nun sonuç sunucusunu bağlamaya çalışacağı yerel IP adresini ve portunu tanımlar. Analiz makinelerinizin ağ yapılandırmasıyla eşleştiğinden emin olun, aksi takdirde sonuçları iletemezler.
- `[database]` içindeki `connection`: Veritabanı bağlantı dizesi, Cuckoo'nun iç veritabanına nasıl bağlanacağını tanımlar. SQLAlchemy tarafından desteklenen herhangi bir DBMS'yi, geçerli bir Database Urls sözdizimini kullanarak kullanabilirsiniz."

> Resultserver IP'nizi kontrol edin! Bazı sanallaştırma yazılımları (örneğin, Virtualbox) sanal bir makine başlatılana kadar sanal ağ arayüzlerini devreye almaz. Cuckoo, resultserver'ı bağladığınız arayüzün başlamasından önce etkinleştirmelidir, bu nedenle lütfen ağ yapılandırmanızı kontrol edin. Arayüzü nasıl etkinleştireceğinizden emin değilseniz, iyi bir ipucu, bir analiz sanal makinesini manuel olarak başlatıp durdurmanızdır; bu, sanal ağları devreye alacaktır. Ağınızda NAT/PAT kullanıyorsanız, resultserver IP'sini tüm arayüzlerde dinlemek için 0.0.0.0 olarak ayarlayabilir, ardından <machinery>.conf'daki resultserver_ip

> ve resultserver_port seçeneklerini kullanarak her makinenin gördüğü adresi ve portu belirtmek için özel seçenekleri kullanabilirsiniz. Unutmayın ki eğer cuckoo.conf'de resultserver IP'sini 0.0.0.0 olarak ayarlarsanız, tüm sanal makineler için resultserver_ip'yi ayarlamanız gerekecektir.

auxiliary.conf

Auxiliary modüller, malware analizi ile eş zamanlı olarak çalışan betiklerdir; bu dosya, bu modüllerin seçeneklerini tanımlar.

Aşağıda varsayılan `$CWD/conf/auxiliary.conf` dosyası bulunmaktadır:

```
[sniffer]
# Enable or disable the use of an external sniffer (tcpdump) [yes/no].
enabled = yes

# Specify the path to your local installation of tcpdump. Make sure this
# path is correct.
tcpdump = /usr/sbin/tcpdump

# We used to define the network interface to capture on in auxiliary.conf, but
# this has been moved to the "interface" field of each Virtual Machinery
# configuration.

# Specify a Berkeley packet filter to pass to tcpdump.
# Note: packer filtering is not possible when using "nictrace" functionality
# from VirtualBox (for example dumping inter-VM traffic).
bpf =

[mitm]
# Enable man in the middle proxying (mitmdump) [yes/no].
enabled = no

# Specify the path to your local installation of mitmdump. Make sure this
# path is correct.
mitmdump = /usr/local/bin/mitmdump

# Listen port base. Each virtual machine will use its own port to be
# able to make a good distinction between the various running analyses.
# Generally port 50000 should be fine, in this case port 50001, 50002, etc
# will also be used - again, one port per analyses.
port_base = 50000
```

```
# Script file to interact with the network traffic. Please refer to the
# documentation of mitmproxy/mitmdump to get an understand of their internal
# workings. (https://mitmproxy.org/doc/scripting/inlinescripts.html)
script = stuff/mitm.py

# Path to the certificate to be used by mitmdump. This file will be
# automatically generated for you if you run mitmdump once. It's just that
# you have to copy it from ~/.mitmproxy/mitmproxy-ca-cert.p12 to somewhere
# in the analyzer/windows/ directory. Recommended is to write the certificate
# to analyzer/windows/bin/cert.p12, in that case the following option should
# be set to bin/cert.p12.
certificate = bin/cert.p12

[replay]
# Enable PCAP replay capabilities.
enabled = yes

# Specify the path to your local installation of mitmdump. Make sure this
# path is correct. Note that this should be mitmproxy 3.0.5 or higher,
# installed in a separate virtualenv (or similar).
mitmdump = /usr/local/bin/mitmdump

# Listen port base. Each virtual machine will use its own port to be
# able to make a good distinction between the various running analyses.
# Generally port 51000 should be fine, in this case port 51001, 51002, etc
# will also be used - again, one port per analyses.
port_base = 51000

# Path to the certificate to be used by mitmdump. This file will be
# automatically generated for you if you run mitmdump once. It's just that
# you have to copy it from ~/.mitmproxy/mitmproxy-ca-cert.p12 to somewhere
# in the analyzer/windows/ directory. Recommended is to write the certificate
# to analyzer/windows/bin/cert.p12, in that case the following option should
# be set to bin/cert.p12.
certificate = bin/cert.p12

[services]
# Provide extra services accessible through the network of the analysis VM
# provided in separate, standalone, Virtual Machines [yes/no].
enabled = no

# Comma-separated list with each Virtual Machine containing said service(s).
services = honeyd
```

```
# Time in seconds required to boot these virtual machines. E.g., some services
# will only get online after a minute because initialization takes a while.
timeout = 0

[reboot]
# This auxiliary module should be enabled for reboot analysis support.
enabled = yes
```

<machinery>.conf

Machinery modülleri, Cuckoo'nun tercih ettiğiniz sanallaştırma yazılımı ile nasıl etkileşimde bulunması gerektiğini tanımlayan betiklerdir.

Her modül, mevcut makineler hakkında ayrıntıları tanımlayan bir yapılandırma dosyasına sahiptir. Örneğin, Cuckoo, bir VMWare machinery modülü ile birlikte gelir. Onu kullanmak için `$CWD/conf/cuckoo.conf` dosyasında machinery seçeneğini vmware olarak belirtmek ve `$CWD/conf/vmware.conf` dosyasını kullanılabilir Sanal Makinelerle doldurmak gereklidir.

Cuckoo, varsayılan olarak bazı modüller sağlar ve bu kılavuzun kapsamında, VirtualBox'u kullanacağınızı varsayacağız.

Aşağıda varsayılan `$CWD/conf/virtualbox.conf` dosyası bulunmaktadır:

```
[virtualbox]
# Specify which VirtualBox mode you want to run your machines on.
# Can be "gui" or "headless". Please refer to VirtualBox's official
# documentation to understand the differences.
mode = headless

# Path to the local installation of the VBoxManage utility.
path = /usr/bin/VBoxManage
# If you are running Cuckoo on Mac OS X you have to change the path as
follows:
# path = /Applications/VirtualBox.app/Contents/MacOS/VBoxManage

# Default network interface.
interface = vboxnet0

# Specify a comma-separated list of available machines to be used. For each
# specified ID you have to define a dedicated section containing the details
# on the respective machine. (E.g. cuckoo1,cuckoo2,cuckoo3)
machines = cuckoo1
```

```
# If remote control is enabled in cuckoo.conf, specify a port range to use.
# Virtualbox will bind the VRDP interface to the first available port.
controlports = 5000-5050


[cuckoo1]
# Specify the label name of the current machine as specified in your
# VirtualBox configuration.
label = cuckoo1

# Specify the operating system platform used by current machine
# [windows/darwin/linux].
platform = windows

# Specify the IP address of the current virtual machine. Make sure that the
# IP address is valid and that the host machine is able to reach it. If not,
# the analysis will fail.
ip = 192.168.56.101

# (Optional) Specify the snapshot name to use. If you do not specify a snapshot
# name, the VirtualBox MachineManager will use the current snapshot.
# Example (Snapshot1 is the snapshot name):
snapshot =

# (Optional) Specify the name of the network interface that should be used
# when dumping network traffic from this machine with tcpdump. If specified,
# overrides the default interface specified in auxiliary.conf
# Example (vboxnet0 is the interface name):
interface =

# (Optional) Specify the IP of the Result Server, as your virtual machine sees it.
# The Result Server will always bind to the address and port specified in
cuckoo.conf,
# however you could set up your virtual network to use NAT/PAT, so you can
specify here
# the IP address for the Result Server as your machine sees it. If you don't
specify an
# address here, the machine will use the default value from cuckoo.conf.
# NOTE: if you set this option you have to set result server IP to 0.0.0.0 in
cuckoo.conf.
# Example:
resultserver_ip =

# (Optional) Specify the port for the Result Server, as your virtual machine sees
it.
```

```
# The Result Server will always bind to the address and port specified in
cuckoo.conf,
# however you could set up your virtual network to use NAT/PAT, so you can
specify here
# the port for the Result Server as your machine sees it. If you don't specify a
port
# here, the machine will use the default value from cuckoo.conf.
# Example:
resultserver_port =

# (Optional) Set your own tags. These are comma separated and help to identify
# specific VMs. You can run samples on VMs with tag you require.
tags =

# Mostly unused for now. Please don't fill it out.
options =

# (Optional) Specify the OS profile to be used by volatility for this
# virtual machine. This will override the guest_profile variable in
# memory.conf which solves the problem of having multiple types of VMs
# and properly determining which profile to use.
osprofile =


[honeyd]
# For more information on this VM please refer to the "services" section of
# the conf/auxiliary.conf configuration file. This machine is a bit special
# in the way that its used as an additional VM for an analysis.
# *NOTE* that if this functionality is used, the VM should be registered in
# the "machines" list in the beginning of this file.
label = honeyd
platform = linux
ip = 192.168.56.102
# The tags should at least contain "service" and the name of this service.
# This way the services auxiliary module knows how to find this particular VM.
tags = service, honeyd
# Not all services actually have a Cuckoo Agent running in the VM, for those
# services one can specify the "noagent" option so Cuckoo will just wait until
# the end of the analysis instead of trying to connect to the non-existing
# Cuckoo Agent. We can't really intercept any inter-VM communication from the
# host / gateway so in order to dump traffic between VMs we have to use a
# different network dumping approach. For this machine we use the "nictrace"
# functionality from VirtualBox (which is basically their internal tcpdump)
# and thus properly dumps inter-VM traffic.
options = nictrace noagent
```

Diğer machinery modülleri için yapılandırma genellikle aynı görünmektedir, gerektiğinde bazı değişikliklerle birlikte. Örneğin, XenServer bir API aracılığıyla çalıştığından, ona erişim sağlamak için bir URL ve kimlik bilgileri gereklidir.

Seçenekler için yapılan yorum satırları yeterince açıklayıcıdır.

Aşağıda varsayılan `$CWD/conf/kvm.conf` dosyası bulunmaktadır:

```
[kvm]
# Specify a libvirt URI connection string
dsn = qemu:///system

# Specify a comma-separated list of available machines to be used. For each
# specified ID you have to define a dedicated section containing the details
# on the respective machine. (E.g. cuckoo1,cuckoo2,cuckoo3)
machines = cuckoo1

# Specify the name of the default network interface that will be used
# when dumping network traffic with tcpdump.
# Example (virbr0 is the interface name):
interface = virbr0


[cuckoo1]
# Specify the label name of the current machine as specified in your
# libvirt configuration.
label = cuckoo1

# Specify the operating system platform used by current machine
# [windows/darwin/linux].
platform = windows

# Specify the IP address of the current virtual machine. Make sure that the
# IP address is valid and that the host machine is able to reach it. If not,
# the analysis will fail. You may want to configure your network settings in
# /etc/libvirt/<hypervisor>/networks/
ip = 192.168.122.101

# (Optional) Specify the snapshot name to use. If you do not specify a snapshot
# name, the KVM MachineManager will use the current snapshot.
# Example (Snapshot1 is the snapshot name):
snapshot =

# (Optional) Specify the name of the network interface that should be used
# when dumping network traffic from this machine with tcpdump.
```

```
# Example (virbr0 is the interface name):
interface =

# (Optional) Specify the IP of the Result Server, as your virtual machine sees it.
# The Result Server will always bind to the address and port specified in
cuckoo.conf,
# however you could set up your virtual network to use NAT/PAT, so you can
specify here
# the IP address for the Result Server as your machine sees it. If you don't
specify an
# address here, the machine will use the default value from cuckoo.conf.
# NOTE: if you set this option you have to set result server IP to 0.0.0.0 in
cuckoo.conf.
# Example:
resultserver_ip =

# (Optional) Specify the port for the Result Server, as your virtual machine sees
it.
# The Result Server will always bind to the address and port specified in
cuckoo.conf,
# however you could set up your virtual network to use NAT/PAT, so you can
specify here
# the port for the Result Server as your machine sees it. If you don't specify a
port
# here, the machine will use the default value from cuckoo.conf.
# Example:
resultserver_port =

# (Optional) Set your own tags. These are comma separated and help to identify
# specific VMs. You can run samples on VMs with tag you require.
tags =

# (Optional) Specify the OS profile to be used by volatility for this
# virtual machine. This will override the guest_profile variable in
# memory.conf which solves the problem of having multiple types of VMs
# and properly determining which profile to use.
osprofile =
```

memory.conf

Volatility aracı, bellek döküm analizi için büyük bir eklenti seti sunar. Bunlardan bazıları oldukça yavaştır. `$CWD/conf/volatility.conf` dosyası, isteğe bağlı olarak eklentileri etkinleştirmenize veya devre dışı bırakmanıza olanak tanır. Volatility'i kullanmak için iki adımı takip etmelisiniz:

- `$CWD/conf/processing.conf` dosyasında volatility'yi etkinleştirin.
- `$CWD/conf/cuckoo.conf` dosyasında memory_dump'ı etkinleştirin.

`$CWD/conf/memory.conf` dosyasının temel bölümünde, Volatility profili ve bellek dökümlerinin işlendikten sonra silinip silinmeyeceğini (bu, çok miktarda disk alanı tasarrufu sağlar) yapılandırabilirsiniz:

```
# Basic settings
[basic]
# Profile to avoid wasting time identifying it
guest_profile = WinXPSP2x86
# Delete memory dump after volatility processing.
delete_memdump = no
```

Sonrasında, her eklentinin kendi yapılandırma bölümü bulunmaktadır:

```
# Scans for hidden/injected code and dlls
# http://code.google.com/p/volatility/wiki/CommandReference#malfind
[malfind]
enabled = on
filter = on

# Lists hooked api in user mode and kernel space
# Expect it to be very slow when enabled
# http://code.google.com/p/volatility/wiki/CommandReference#apihooks
[apihooks]
enabled = off
filter = on
```

Filtre yapılandırması, sonuç raporundan bilinen temiz veriyi kaldırmanıza yardımcı olur. Her eklenti için ayrı ayrı yapılandırılabilir.

Filtre kendisi [mask] bölümünde yapılandırılır. pid_generic içinde süreçleri filtrelemek için pid'lerin bir listesini girebilirsiniz:

```
# Masks. Data that should not be logged
# Just get this information from your plain VM Snapshot (without running
malware)
# This will filter out unwanted information in the logs
[mask]
```

```
# pid_generic: a list of process ids that already existed on the machine before
the malware was started.
pid_generic = 4, 680, 752, 776, 828, 840, 1000, 1052, 1168, 1364, 1428, 1476,
1808, 452, 580, 652, 248, 1992, 1696, 1260, 1656, 1156
```

processing.conf

Bu dosya, tüm işleme modüllerini etkinleştirmenize, devre dışı bırakmanıza ve yapılandırmanıza olanak tanır. Bu modüller, `cuckoo.processing` modülü altında bulunur ve analiz sırasında toplanan ham verilerin nasıl işleneceğini tanımlar.

`$CWD/conf/processing.conf`dosyasında her işleme modülü için bir bölüm bulacaksınız.

```
# Enable or disable the available processing modules [yes/no].
# If you add a custom processing module to your Cuckoo setup, you have to add
# a dedicated entry in this file, or it won't be executed.
# You can also add additional options under the section of your module and
# they will be available in your Python class.

[analysisinfo]
enabled = yes

[apkinfo]
enabled = no
# Decompiling dex files with androguard in a heavy operation. For large dex
# files it can really take quite a while - it is recommended to limit to a
# certain filesize.
decompilation_threshold = 5000000

[baseline]
enabled = no

[behavior]
enabled = yes

[buffer]
enabled = yes

[debug]
enabled = yes

[droidmon]
enabled = no
```

```
[dropped]
enabled = yes

[dumptls]
enabled = yes

[extracted]
enabled = yes

[googleplay]
enabled = no
android_id =
google_login =
google_password =

[memory]
# Create a memory dump of the entire Virtual Machine. This memory dump will
# then be analyzed using Volatility to locate interesting events that can be
# extracted from memory.
enabled = no

[misp]
enabled = no
url =
apikey =

# Maximum amount of IOCs to look up (hard limit).
maxioc = 100

[network]
enabled = yes

# Allow domain whitelisting
whitelist_dns = no

# Allow DNS responses from your configured DNS server for whitelisting to
# deactivate when responses come from some other DNS
# Can be also multiple like : 8.8.8.8,8.8.4.4
allowed_dns =

[procmemory]
# Enables the creation of process memory dumps for each analyzed process
right
# before they terminate themselves or right before the analysis finishes.
enabled = yes
```

```
# It is possible to load these process memory dumps in IDA Pro through the
# generation of IDA Python-based script files. Although currently symbols and
# such are not properly recovered, it is still nice to get a quick look at
# specific memory addresses of a process.
idapro = no
# Extract executable images from this process memory dump. This allows us to
# relatively easily extract injected executables.
extract_img = yes
# Also extract DLL files from the process memory dump.
extract_dll = no
# Delete process memory dumps after analysis to save disk space.
dump_delete = no

[procmon]
# Enable procmon processing. This only takes place when the "procmon=1"
option
# is set for an analysis.
enabled = yes

[screenshots]
enabled = yes
# Set to the actual tesseract path (i.e., /usr/bin/tesseract or similar)
# rather than "no" to enable OCR analysis of screenshots.
# Note: doing OCR on the screenshots is a rather slow process.
tesseract = no

[snort]
enabled = no
# Following are various configurable settings. When in use of a recent 2.9.x.y
# version of Snort there is no need to change any of the following settings as
# they represent the defaults.
#
snort = /usr/local/bin/snort
conf = /etc/snort/snort.conf

[static]
enabled = yes
# On bigger PDF files PeePDF may take a substantial amount of time to perform
# static analysis of PDF files, with times of over an hour per file estimated
# in production. This option will by default limit the maximum processing time
# to one minute, but this may be adjusted accordingly. Note that if the timeout
# is hit, no static analysis results through PeePDF will be available.
pdf_timeout = 60
```

```
[strings]
enabled = yes

[suricata]
enabled = no

# Following are various configurable settings. When in use of a recent version
# of Suricata there is no need to change any of the following settings as they
# represent the defaults.
suricata = /usr/bin/suricata
conf = /etc/suricata/suricata.yaml
eve_log =  eve.json
files_log = files-json.log
files_dir = files

# By specifying the following line our processing module can use the socket
# mode in Suricata. This is quite the performance improvement as instead of
# having to load all the Suricata rules for each time the processing module is
# ran (i.e., for every task), the rules are only loaded once and then we talk
# to its API. This does require running Suricata as follows or similar;
# "suricata --unix-socket -D".
# (Please find more information in utils/suricata.sh for now).
# socket = /var/run/suricata/cuckoo.socket
socket =

[targetinfo]
enabled = yes

[virustotal]
enabled = no
# How much time we can wait to establish VirusTotal connection and get the
# report.
timeout = 60
# Enable this option if you want to submit files to VirusTotal not yet available
# in their database.
# NOTE: if you are dealing with sensitive stuff, enabling this option you could
# leak some files to VirusTotal.
scan = no
# 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 shouldn't affect your use.
key =
a0283a2c3d55728300d064874239b5346fb991317e8449fe43c902879d758088
```

```
[irma]
enabled = no
# IRMA @ github : https://github.com/quarkslab/irma
# How much time we can wait to establish IRMA connection and get the report.
timeout = 60
# Enable this option if you want to submit files to IRMA not yet available.
scan = no
# Force scan of submitted files
force = no
# URL to your IRMA installation
# For example : https://your.irma.host
url =
# Probes to use on your IRMA instance
# If not specified, will default to using all available probes
# Expects comma separated list
# For example : ClamAV,F-
Secure,Avast,ESET,eScan,Avira,Sophos,McAfee,Kaspersky,GData,Comodo,Bitdef
ender
probes =
```

reporting.conf

`$CWD/conf/reporting.conf` dosyası, rapor oluşturma ile ilgili bilgileri içerir.

Aşağıdaki gibidir:

```
# Enable or disable the available reporting modules [on/off].
# If you add a custom reporting module to your Cuckoo setup, you have to add
# a dedicated entry in this file, or it won't be executed.
# You can also add additional options under the section of your module and
# they will be available in your Python class.

[feedback]
# Automatically report errors that occurred during an analysis. Requires the
# Cuckoo Feedback settings in cuckoo.conf to have been filled out properly.
enabled = no

[jsondump]
enabled = yes
indent = 4
calls = yes

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

```
enabled = no
# Enable creation of report.html?
html = no
# Enable creation of report.pdf?
pdf = no

[misp]
enabled = no
url =
apikey =

# The various modes describe which information should be submitted to MISP,
# separated by whitespace. Available modes: maldoc ipaddr hashes url.
mode = maldoc ipaddr hashes url

distribution = 0
analysis = 0
threat_level = 4

# The minimum Cuckoo score for a MISP event to be created
min_malscore = 0

tag = Cuckoo
upload_sample = no

[mongodb]
enabled = no
host = 127.0.0.1
port = 27017
db = cuckoo
store_memdump = yes
paginate = 100
# MongoDB authentication (optional).
username =
password =

[elasticsearch]
enabled = no
# Comma-separated list of ElasticSearch hosts. Format is IP:PORT, if port is
# missing the default port is used.
# Example: hosts = 127.0.0.1:9200, 192.168.1.1:80
hosts = 127.0.0.1
# Increase default timeout from 10 seconds, required when indexing larger
# analysis documents.
timeout = 300
# Set to yes if we want to be able to search every API call instead of just
```

```
# through the behavioral summary.
calls = no
# Index of this Cuckoo instance. If multiple Cuckoo instances connect to the
# same ElasticSearch host then this index (in Moloch called "instance") should
# be unique for each Cuckoo instance.
index = cuckoo

# Logging time pattern.  This sets how elasticsearch creates indexes
# by default it is yearly in most instances this will be sufficient
# valid options: yearly, monthly, daily
index_time_pattern = yearly

# Cuckoo node name in Elasticsearch to identify reporting host. Can be useful
# for automation and while referring back to correct Cuckoo host.
cuckoo_node =

[moloch]
enabled = no
# If the Moloch web interface is hosted on a different IP address than the
# Cuckoo Web Interface then you'll want to override the IP address here.
host =
# If you wish to run Moloch in http (insecure) versus https (secure) mode,
# set insecure to yes.
insecure = no

# Following are various configurable settings. When in use of a recent version
# of Moloch there is no need to change any of the following settings as they
# represent the defaults.
moloch_capture = /data/moloch/bin/moloch-capture
conf = /data/moloch/etc/config.ini
instance = cuckoo

[notification]
# Notification module to inform external systems that analysis is finished.
# You should consider keeping this as very last reporting module.
enabled = no

# External service URL where info will be POSTed.
# example : https://my.example.host/some/destination/url
url =

# Cuckoo host identifier - can be hostname.
# for example : my.cuckoo.host
identifier =
```

```
[mattermost]
enabled = no

# Mattermost webhook URL.
# example : https://my.mattermost.host/hooks/yourveryrandomkey
url =

# Cuckoo host URL to make analysis ID clickable.
# example : https://my.cuckoo.host/
myurl =

# Username to show when posting message
username = cuckoo

# What kind of data to show apart from default.
# Show virustotal hits.
show_virustotal = no

# Show matched cuckoo signatures.
show_signatures = no

# Show collected URL-s by signature "network_http".
show_urls = no

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

Revision #1
Created 25 December 2023 09:04:27 by Ertan Sözer
Updated 25 December 2023 09:10:20 by Ertan Sözer