Cuckoo Kullanımı

Cuckoo’yu Çalıştırmak

Cuckoo'yu başlatmak için şu komutu kullanın:

```
$ cuckoo
```

Şuna benzer bir çıktı alacaksınız:

```
  eeee e   e eeee e   e  eeeee eeeee
  8  8 8   8 8  8 8   8  8  88 8  88
  8e   8e  8 8e   8eee8e 8   8 8   8
  88   88  8 88   88   8 8   8 8   8
  88e8 88ee8 88e8 88   8 8eee8 8eee8

 Cuckoo Sandbox 2.0.0
 www.cuckoosandbox.org
 Copyright (c) 2010-2017

 Checking for updates...
 Good! You have the latest version available.

2017-03-31 17:08:53,527 [cuckoo.core.scheduler] INFO: Using "virtualbox" as
machine manager
2017-03-31 17:08:53,935 [cuckoo.core.scheduler] INFO: Loaded 1 machine/s
2017-03-31 17:08:53,964 [cuckoo.core.scheduler] INFO: Waiting for analysis
tasks.
```

Cuckoo, güncellemeleri `api.cuckoosandbox.org` adresindeki uzaktan bir API üzerinden kontrol eder. Bu işlemi devre dışı bırakmak için yapılandırma dosyasındaki `version_check` seçeneğini devre dışı bırakabilirsiniz.

Şimdi Cuckoo çalışmaya hazır ve bekliyor.

Cuckoo, yardım komutu ile gösterildiği gibi bazı komut satırı seçeneklerini kabul eder:

```
$ cuckoo --help
Usage: cuckoo [OPTIONS] COMMAND [ARGS]...
```

```
Invokes the Cuckoo daemon or one of its subcommands.

To be able to use different Cuckoo configurations on the same
machine with the same Cuckoo installation, we use the so-called
Cuckoo Working Directory (aka "CWD"). A default CWD is
available, but may be overridden through the following options -
listed in order of precedence.

* Command-line option (--cwd)
* Environment option ("CUCKOO")
* Environment option ("CUCKOO_CWD")
* Current directory (if the ".cwd" file exists)
* Default value ("~/.cuckoo")

Options:
  -d, --debug             Enable verbose logging
  -q, --quiet             Only log warnings and critical messages
  -m, --maxcount INTEGER  Maximum number of analyses to process
  --user TEXT             Drop privileges to this user
  --cwd TEXT              Cuckoo Working Directory
  --help                  Show this message and exit.

Commands:
  api          Operate the Cuckoo REST API.
  clean        Clean the CWD and associated databases.
  community    Fetch supplies from the Cuckoo Community.
  distributed  Distributed Cuckoo helper utilities.
  dnsserve     Custom DNS server.
  import       Imports an older Cuckoo setup into a new CWD.
  init         Initializes Cuckoo and its configuration.
  machine      Dynamically add/remove machines.
  migrate      Perform database migrations.
  process      Process raw task data into reports.
  rooter       Instantiates the Cuckoo Rooter.
  submit       Submit one or more files or URLs to Cuckoo.
  web          Operate the Cuckoo Web Interface.
```

`--debug` ve `--quiet` flagleri, cuckoo komutu veya herhangi bir alt komut için günlüğün ayrıntısını artırır veya azaltır.

Arkaplanda Cuckoo

Cuckoo'yu manuel olarak çalıştırmak, başlangıçta kullanmaya başladığınızda faydalıdır, ancak Cuckoo'yu içeren birden çok makine çalıştırıyorsanız, Cuckoo'yu çalıştırma sürecini otomatikleştirmek isteyeceksiniz.

Neyse ki Cuckoo, Cuckoo Working Directory'de (bu konu bir sonraki sayfada açıklanacaktır) supervisord.conf dosyasını otomatik olarak sağlar; bu dosya ya CWD dizininden supervisord'yi çalıştırarak ya da yapılandırmayı doğrudan supervisord'a sağlayarak başlatılabilir:

```
$ supervisord -c $CWD/supervisord.conf
```

Varsayılan olarak, supervisord ayrıca dört Processing Utility örneğini de başlatacaktır, bu da demektir ki, belgelerine göre, `$CWD/conf/cuckoo.conf`'daki process_results yapılandırmasının devre dışı bırakılması gerekmektedir (yani, değeri on'dan off'a değiştirilmelidir).

Bu noktadan itibaren, çeşitli cuckoo işlemlerini (yani, ana cuckoo işlemi ve dört işleme örneği) başlatmak ve durdurmak için şu türden komutları çalıştırabilirsiniz (bunların CWD'den çalıştırıldığını varsayalım):

```
# Stop the Cuckoo daemon and the processing utilities.
$ supervisorctl stop cuckoo:

# Start the Cuckoo daemon and the processing utilities.
$ supervisorctl start cuckoo:
```

Unutmayın ki Cuckoo denetleyici grubunu belirtmek için (yani, cuckoo:), Cuckoo daemon işlemini ve çeşitli işleme yardımcı programlarını içerir.

Analiz

Submission Utility

Bir analiz göndermenin en kolay yolu, cuckoo submit yardımcı programını kullanmaktır. Şu anda şu seçeneklere sahiptir:

```
$ cuckoo submit --help
Usage: cuckoo submit [OPTIONS] [TARGET]...

  Submit one or more files or URLs to Cuckoo.

Options:
  -u, --url           Submitting URLs instead of samples
  -o, --options TEXT  Options for these tasks
  --package TEXT      Analysis package to use
  --custom TEXT       Custom information to pass along this task
  --owner TEXT        Owner of this task
  --timeout INTEGER   Analysis time in seconds
  --priority INTEGER  Priority of this task
  --machine TEXT      Machine to analyze these tasks on
  --platform TEXT     Analysis platform
  --memory            Enable memory dumping
  --enforce-timeout   Don't terminate the analysis early
  --clock TEXT        Set the system clock
  --tags TEXT         Analysis tags
  --baseline          Create baseline task
  --remote TEXT       Submit to a remote Cuckoo instance
  --shuffle           Shuffle the submitted tasks
  --pattern TEXT      Provide a glob-pattern when submitting a
                      directory
  --max INTEGER       Submit up to X tasks at once
  --unique            Only submit samples that have not been
                      analyzed before
  -d, --debug         Enable verbose logging
  -q, --quiet         Only log warnings and critical messages
  --help              Show this message and exit.
```

Bir seferde birden çok dosya veya dizini belirtebilirsiniz. cuckoo submit, dizinin tüm dosyalarını numaralandırır ve bunları birer birer gönderir.

Analiz paketleri kavramı, bu belgelerin ilerleyen kısımlarında ele alınacaktır (Analiz Paketleri bölümünde). İşte bazı kullanım örnekleri:

local binary gönderme:

```
$ cuckoo submit /path/to/binary
```

URL gönderme:

```
$ cuckoo submit --url http://www.example.com
```

Yerel bir binary dosya gönderme ve daha yüksek bir öncelik belirtme:

```
$ cuckoo submit --priority 5 /path/to/binary
```

Yerel bir binary dosya gönderme ve özel bir analiz süresi aşımını 60 saniye olarak belirtme:

```
$ cuckoo submit --timeout 60 /path/to/binary
```

Yerel bir binary dosya gönderme ve özel bir analiz paketi belirtme:

```
$ cuckoo submit --package <name of package> /path/to/binary
```

Yerel bir binary dosya gönderme ve özel bir yönlendirme belirtme:

```
$ cuckoo submit -o route=tor /path/to/binary
```

Yerel bir binary dosya gönderme ve özel bir analiz paketi ve bazı seçenekler belirtme (bu durumda kötü amaçlı yazılım için bir komut satırı argümanı):

```
$ cuckoo submit --package exe --options arguments=--dosomething
/path/to/binary.exe
```

Yerel bir binary dosya gönderme ve cuckoo1 adlı sanal makinede çalıştırılacak şekilde belirtme:

```
$ cuckoo submit --machine cuckoo1 /path/to/binary
```

Yerel bir binary dosya gönderme ve bir Windows makinesinde çalıştırılacak şekilde belirtme:

```
$ cuckoo submit --platform windows /path/to/binary
```

Yerel bir binary dosya gönderme ve analiz makinesinin tam bellek dökümünü almak için:

```
$ cuckoo submit --memory /path/to/binary
```

Yerel bir binary dosya gönderme ve analizin tam zaman aşımı süresince (Cuckoo'nun analizi sonlandırmaya ne zaman karar vereceği iç mekanizmasını göz ardı ederek) çalışmasını zorlama:

```
$ cuckoo submit --enforce-timeout /path/to/binary
```

Yerel bir binary dosya gönderme ve sanal makine saatinin ayarlanması. Biçimi %m-%d-%Y %H:%M:%S. Belirtilmezse, mevcut zaman kullanılır. Örneğin, örneği 24 Ocak 2001 tarihinde 14:41:20'de çalıştırmak istiyorsak:

```
$ cuckoo submit --clock "01-24-2001 14:41:20" /path/to/binary
```

Volatility analizi için bir örnek gönderme (cuckoo kanca kullanımının yan etkilerini azaltmak için seçenekleri free=True olarak kapatma):

```
$ cuckoo submit --memory --options free=yes /path/to/binary
```

Python Fonksiyonları

Gönderimleri, örnekleri ve genel yürütme durumunu takip etmek için, Cuckoo SQLite, MySQL veya MariaDB, PostgreSQL ve birçok diğer SQL veritabanı sistemini kullanmanıza izin veren popüler bir Python ORM olan SQLAlchemy kullanır.

Cuckoo, daha büyük çözümlere kolayca entegre edilebilen ve tamamen otomatik hale getirilebilen bir tasarıma sahiptir. Analiz gönderimini otomatikleştirmek için REST API arayüzünü kullanmanızı öneririz (bkz. REST API), ancak kendi Python gönderim betiğinizi yazmak istiyorsanız add_path() ve add_url() işlevlerini kullanabilirsiniz.

> **`add_path`** ***(file_path[, timeout=0[, package=None[, options=None[, priority=1[, custom=None[, owner=""[, machine=None[, platform=None[, tags=None[, memory=False[, enforce_timeout=False], clock=None[]]]]]]]]]]]])***

Bekleyen analiz görevleri listesine yerel bir dosya ekler. Yeni oluşturulan görevin ID'sini döndürür.

Parametreler:

- file_path (string): Gönderilecek dosyanın yolu
- timeout (integer): Analiz süresinin maksimum saniye cinsinden miktarı
- package (string veya None): Belirtilen dosya için kullanmak istediğiniz analiz paketi
- options (string veya None): Analiz paketine iletilmesi gereken seçeneklerin listesi (key=value, key=value formatında)
- priority (integer): Belirtilen dosyaya atanacak önceliğin sayısal temsili (1 düşük, 2 orta, 3 yüksek)
- custom (string veya None): İşleme veya raporlamada kullanılmak üzere iletilmesi gereken özel değer
- owner (string veya None): Görev sahibi
- machine (string veya None): Kullanmak istediğiniz sanal makinenin Cuckoo kimliği; belirtilmezse otomatik olarak bir tane seçilir
- platform (string veya None): Çalıştırmak istediğiniz analizin işletim sistemi platformu (şu anda yalnızca Windows)
- tags (string veya None): Makine seçimi için etiketler
- memory (True veya False): Analiz makinesinin tam bellek dökümünü oluşturmak için True olarak ayarlayın
- enforce_timeout (True veya False): Süreyi tam olarak uygulamak için True olarak ayarlayın
- clock (string veya None): Analiz makinesinde ayarlanacak özel bir saat zamanı

Return type: integer

Örnek kullanım:

```
>>> from cuckoo.core.database import Database
>>> db = Database()
>>> db.add_path("/tmp/malware.exe")
1
>>>
```

add_url(*url[, timeout=0[, package=None[, options=None[, priority=1[, custom=None[, owner=""[, machine=None[, platform=None[, tags=None[, memory=False[, enforce_timeout=False], clock=None[]]]]]]]]]]]])

Bekleyen analiz görevleri listesine yerel bir dosya ekler. Yeni oluşturulan görevin ID'sini döndürür.

Parametreler:

- url (string): Analiz edilecek URL
- timeout (integer): Analiz süresinin maksimum saniye cinsinden miktarı
- package (string veya None): Belirtilen URL için kullanmak istediğiniz analiz paketi
- options (string veya None): Analiz paketine iletilmesi gereken seçeneklerin listesi (key=value, key=value formatında)
- priority (integer): Belirtilen URL'ye atanacak önceliğin sayısal temsili (1 düşük, 2 orta, 3 yüksek)
- custom (string veya None): İşleme veya raporlamada kullanılmak üzere iletilmesi gereken özel değer
- owner (string veya None): Görev sahibi
- machine (string veya None): Kullanmak istediğiniz sanal makinenin Cuckoo kimliği; belirtilmezse otomatik olarak bir tane seçilir
- platform (string veya None): Çalıştırmak istediğiniz analizin işletim sistemi platformu (şu anda yalnızca Windows)
- tags (string veya None): Makine seçimi için etiketler
- memory (True veya False): Analiz makinesinin tam bellek dökümünü oluşturmak için True olarak ayarlayın
- enforce_timeout (True veya False): Süreyi tam olarak uygulamak için True olarak ayarlayın
- clock (string veya None): Analiz makinesinde ayarlanacak özel bir saat zamanı

Return type: integer

Örnek kullanım:

```
```

```
>>> from cuckoo.core.database import Database
>>> db = Database()
>>> db.connect()
>>> db.add_url("http://www.cuckoosandbox.org")
2
>>>
```

Web Arayüzü

Cuckoo, bir Django uygulaması olarak tam teşekküllü bir web arayüzü sağlar. Bu arayüz, dosyaları göndermenize, raporları göz atmanıza ve tüm analiz sonuçları arasında arama yapmanıza olanak tanır.

Konfigürasyon

Web arayüzü, verileri bir Mongo veritabanından çeker, bu nedenle Web Arayüzünün çalışması için `reporting.conf` dosyasında Mongo raporlama modülünün etkin olması gereklidir. Eğer öyle değilse, Web Arayüzü başlatılamaz ve bunun yerine bir istisna oluşturulur.

`$CWD/web/local_settings.py` konfigürasyon dosyasında bazı ek konfigürasyon seçenekleri bulunmaktadır.

```
# Copyright (C) 2013 Claudio Guarnieri.
# Copyright (C) 2014-2017 Cuckoo Foundation.
# This file is part of Cuckoo Sandbox - http://www.cuckoosandbox.org
# See the file 'docs/LICENSE' for copying permission.

import web.errors

# Maximum upload size (10GB, so there's basically no limit).
MAX_UPLOAD_SIZE = 10 * 1024 * 1024 * 1024

# Override default secret key stored in $CWD/web/.secret_key
# Make this unique, and don't share it with anybody.
# SECRET_KEY = "YOUR_RANDOM_KEY"

# Language code for this installation. All choices can be found here:
# http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = "en-us"

ADMINS = (
    # ("Your Name", "your_email@example.com"),
)

MANAGERS = ADMINS

# Allow verbose debug error message in case of application fault.
# It's strongly suggested to set it to False if you are serving the
# web application from a web server front-end (i.e. Apache).
DEBUG = False
DEBUG404 = False
```

```
# A list of strings representing the host/domain names that this Django site
# can serve.
# Values in this list can be fully qualified names (e.g. 'www.example.com').
# When DEBUG is True or when running tests, host validation is disabled; any
# host will be accepted. Thus it's usually only necessary to set it in production.
ALLOWED_HOSTS = ["*"]

handler404 = web.errors.handler404
handler500 = web.errors.handler500

#A list of strings representing the subnets or ipaddresses that can download
#samples and dropped files
#Values in this list can be ipv4 or ipv6 separated by ","
#(e.g. '127.0.0.0/8,10.0.0.0/8,fd00::/8').
ALLOWED_FILEDOWNLOAD_SUBNETS = '127.0.0.0/8,10.0.0.0/8,fd00::/8'
```

Üretim ortamlarında DEBUG değişkenini False olarak tutmanız ve en az bir ADMIN girişi yapılandırmanız, e-posta ile hata bildirimini etkinleştirmeniz önerilir.

2.0.0 sürümünde değişiklik: Varsayılan maksimum yükleme boyutu 25 MB'den 10 GB'a yükseltilmiştir, bu nedenle neredeyse herhangi bir dosyanın kabul edilmesi gerekmektedir.

Web Arayüzünün Başlatılması

Web arayüzünü başlatmak için web/ dizininden basitçe aşağıdaki komutu çalıştırabilirsiniz:

```
$ cuckoo web runserver
```

Web arayüzünü belirli bir portta herhangi bir IP'yi dinleyecek şekilde yapılandırmak istiyorsanız, aşağıdaki komutu kullanabilirsiniz (PORT'u istediğiniz port numarasıyla değiştirin):

```
$ cuckoo web runserver 0.0.0.0:PORT
```

Veya doğrudan aşağıdaki gibi runserver bölümü olmadan ve dinlenilecek host'u belirterek kullanabilirsiniz:

```
$ cuckoo web -H 0
```

Web Dağıtımı

Web Arayüzü sunucuyu başlatmanın varsayılan yöntemi birçok durum için uygundur, ancak bazı kullanıcılar sunucuyu daha güvenilir bir şekilde dağıtmak isteyebilir. Web Arayüzünü bir WSGI uygulaması olarak bir web sunucusuna açmak bunu mümkün kılar. Bu bölüm, Web Arayüzünün uWSGI ve nginx üzerinden nasıl dağıtılacağını basit bir örnek gösterir. Bu talimatlar Ubuntu GNU/Linux göz önüne alınarak yazılmıştır, ancak diğer platformlara uyarlanabilir.

Bu çözüm, uWSGI, uWSGI Python eklentisi ve nginx gerektirir. Tümü paketler olarak mevcuttur:

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

uWSGI Kurulumu

İlk olarak, uWSGI'yi Web Arayüzü sunucusunu bir uygulama olarak çalıştırmak için kullanın.

Başlamak için, `cuckoo web --uwsgi` komutu tarafından rapor edilen gerçek konfigürasyonu içeren `/etc/uwsgi/apps-available/cuckoo-web.ini` adında bir uWSGI yapılandırma dosyası oluşturun, örneğin:

```
$ cuckoo web --uwsgi
[uwsgi]
plugins = python
virtualenv = /home/cuckoo/cuckoo
module = cuckoo.web.web.wsgi
uid = cuckoo
gid = cuckoo
static-map = /static=/home/..somepath..
# If you're getting errors about the PYTHON_EGG_CACHE, then
# uncomment the following line and add some path that is
# writable from the defined user.
# env = PYTHON_EGG_CACHE=
env = CUCKOO_APP=web
env = CUCKOO_CWD=/home/..somepath..
```

Bu yapılandırma, dağıtımın varsayılan uWSGI yapılandırmasından bir dizi ayarı devralır ve gerçek işi yapmak için Cuckoo paketinden cuckoo.web.web.wsgi'yi içe aktarır. Bu örnekte, Cuckoo'yu /home/cuckoo/cuckoo konumunda bir sanal ortamda kurduk. Cuckoo global olarak yüklendiyse sanal ortam seçeneği gerekli değildir (ve cuckoo web --uwsgi bunu bildirmez).

Uygulama yapılandırmasını etkinleştirin ve sunucuyu başlatın.

```
$ sudo ln -s /etc/uwsgi/apps-available/cuckoo-web.ini /etc/uwsgi/apps-enabled/
$ sudo service uwsgi start cuckoo-web     # or reload, if already running
```

> Uygulama için günlükler, dağıtım uygulama örnekleri için standart dizinde bulunabilir, yani /var/log/uwsgi/app/cuckoo-web.log. UNIX soketi de geleneksel bir konumda oluşturulur, yani /run/uwsgi/app/cuckoo-web/socket.

nginx Kurulumu

Web Arayüzü sunucusu uWSGI'de çalışırken, nginx artık bir web sunucusu/ters proxy olarak ayarlanabilir ve HTTP isteklerini ona yönlendirebilir.

Başlamak için, cuckoo web --nginx komutu tarafından bildirilen gerçek konfigürasyonu içeren bir nginx konfigürasyon dosyası oluşturun:

```
$ cuckoo web --nginx
upstream _uwsgi_cuckoo_web {
    server unix:/run/uwsgi/app/cuckoo-web/socket;
}

server {
    listen localhost:8000;

    # Cuckoo Web Interface
    location / {
        client_max_body_size 1G;
        uwsgi_pass  _uwsgi_cuckoo_web;
        include     uwsgi_params;
    }
}
```

nginx'nin uWSGI soketine bağlanabilmesi için kullanıcısını cuckoo grubuna ekleyerek emin olun:

```
$ sudo adduser www-data cuckoo
```

Sunucu yapılandırmasını etkinleştirin ve sunucuyu başlatın:

```
$ sudo ln -s /etc/nginx/sites-available/cuckoo-web /etc/nginx/sites-enabled/
$ sudo service nginx start     # or reload, if already running
```

Bu noktada, Web Arayüzü sunucusunun sunucuda 8000 numaralı portta kullanılabilir olması gerekmelidir. Bu yapılandırmayı genişletmek için çeşitli konfigürasyonlar uygulanabilir, örneğin sunucu performansını ayarlamak, kimlik doğrulama eklemek veya HTTPS kullanarak iletişimi güvence altına almak gibi. Ancak, bunu kullanıcıya bir egzersiz bırakıyoruz.

REST API

"Analiz" bölümünde belirtildiği gibi, Cuckoo, Flask kullanılarak uygulanan basit ve hafif bir REST API sunucusu sağlar.

API Server’ı Başlatma

API sunucusunu başlatmak için şu komutu kullanabilirsiniz:

```
$ cuckoo api
```

Varsayılan olarak, servisi localhost:8090 adresinde bağlar. Bu değerleri değiştirmek istiyorsanız, aşağıdaki sözdizimini kullanabilirsiniz:

```
$ cuckoo api --host 0.0.0.0 --port 1337
$ cuckoo api -H 0.0.0.0 -p 1337
```

API'ya yalnızca kimlik doğrulamış erişime izin vermek için cuckoo.conf'daki api_token değeri gizli bir değere ayarlanmalıdır. Yeni Cuckoo yüklemelerinde, sizin için otomatik olarak rastgele bir belirteç oluşturulur. API'ye erişmek için tüm isteklerinizi yapılandırmadaki belirteç kullanılarak Authorization: Bearer <token> başlığını göndermeniz gerekmektedir. API'ye güvensiz bir ağ üzerinden, örneğin İnternet üzerinden erişmek istiyorsanız, API sunucusunu aşağıdaki bölümde açıklanan nginx'in arkasında çalıştırmalı ve HTTPS'yi etkinleştirmelisiniz.

Web Dağıtımı

API sunucusunu başlatmanın varsayılan yöntemi birçok durum için iyi çalışsa da, bazı kullanıcılar sunucuyu güvenilir bir şekilde dağıtmak isteyebilir. Bu, API'yi bir web sunucusu aracılığıyla WSGI uygulaması olarak açığa çıkarmak suretiyle yapılabilir. Bu bölüm, API'nin uWSGI ve nginx üzerinden nasıl dağıtılacağını basit bir örneği göstermektedir. Bu talimatlar Ubuntu GNU/Linux göz önüne alınarak yazılmış olup, diğer platformlara uyarlanabilir.

Bu çözüm, uWSGI, uWSGI Python eklentisi ve nginx gerektirir. Bunlar paket olarak mevcuttur:

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

uWSGI Kurulumu

İlk olarak, API sunucusunu bir uygulama olarak çalıştırmak için uWSGI'yi kullanın.

Başlamak için, `cuckoo api --uwsgi` komutu tarafından rapor edilen gerçek yapılandırmayı içeren `/etc/uwsgi/apps-available/cuckoo-api.ini` adlı bir uWSGI yapılandırma dosyası oluşturun:

```
$ cuckoo api --uwsgi
[uwsgi]
plugins = python
virtualenv = /home/cuckoo/cuckoo
module = cuckoo.apps.api
callable = app
uid = cuckoo
gid = cuckoo
env = CUCKOO_APP=api
env = CUCKOO_CWD=/home/..somepath..
```

Bu yapılandırma, dağıtımın varsayılan uWSGI yapılandırmasından bir dizi ayar devralır ve gerçek işi yapmak için Cuckoo paketinden `cuckoo.apps.api`'yi içeri alır. Bu örnekte Cuckoo'yu `/home/cuckoo/cuckoo` konumundaki bir sanal ortamda yükledik. Cuckoo global olarak yüklendiyse, sanal ortam seçeneği gerekli değildir.

Uygulama yapılandırmasını etkinleştirin ve sunucuyu başlatın.

```
$ sudo ln -s /etc/uwsgi/apps-available/cuckoo-api.ini /etc/uwsgi/apps-enabled/
$ sudo service uwsgi start cuckoo-api     # or reload, if already running
```

> Uygulama için günlük dosyaları, dağıtım uygulama örnekleri için standart dizinde bulunabilir, yani `/var/log/uwsgi/app/cuckoo-api.log`. UNIX soketi de geleneksel bir konumda oluşturulur, yani `/run/uwsgi/app/cuckoo-api/socket`.

nginx Kurulumu

API sunucusunun uWSGI ile çalıştığı bir ortamda, nginx şimdi ona HTTP isteklerini yönlendirmek üzere bir web sunucusu/ters proxy olarak kurulabilir.

Başlamak için, `cuckoo api --nginx` komutu tarafından raporlanan gerçek yapılandırmayı içeren `/etc/nginx/sites-available/cuckoo-api` adlı bir nginx yapılandırma dosyası oluşturun:

```
$ cuckoo api --nginx
upstream _uwsgi_cuckoo_api {
    server unix:/run/uwsgi/app/cuckoo-api/socket;
```

```
}

server {
    listen localhost:8090;

    # REST API app
    location / {
        client_max_body_size 1G;
        uwsgi_pass  _uwsgi_cuckoo_api;
        include     uwsgi_params;
    }
}
```

Nginx'in uWSGI soketiyle iletişim kurabilmesi için, kullanıcısını cuckoo grubuna ekleyerek emin olun:

```
$ sudo adduser www-data cuckoo
```

Sunucu konfigürasyonunu etkinleştirin ve sunucuyu başlatın.

```
$ sudo ln -s /etc/nginx/sites-available/cuckoo-api /etc/nginx/sites-enabled/
$ sudo service nginx start    # or reload, if already running
```

Bu aşamada API sunucusu, sunucuda 8090 numaralı bağlantı noktasında kullanılabilir olmalıdır. Sunucu performansını ayarlamak, kimlik doğrulama eklemek veya HTTPS kullanarak iletişimi güvenli hale getirmek gibi bu yapıyı genişletmek için çeşitli konfigürasyonlar uygulanabilir.

Kaynaklar

Bu bölümde şu anda kullanılabilir olan kaynakların bir listesi ve her birinin kısa bir açıklaması bulunmaktadır.

/tasks/create/file

POST /tasks/create/file

Bir dosyayı bekleyen görevler listesine ekler. Yeni oluşturulan görevin kimliğini döndürür.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" -F file=@/path/to/file
http://localhost:8090/tasks/create/file
```

Python kullanılan örnek request:

```
import requests

REST_URL = "<http://localhost:8090/tasks/create/file>"
SAMPLE_FILE = "/path/to/malwr.exe"
HEADERS = {"Authorization": "Bearer S4MPL3"}

with open(SAMPLE_FILE, "rb") as sample:
    files = {"file": ("temp_file_name", sample)}
    r = requests.post(REST_URL, headers=HEADERS, files=files)

# Add your code to error checking for r.status_code.

task_id = r.json()["task_id"]

# Add your code for error checking if task_id is None.
```

Örnek response:

```
{
   "task_id" : 1
}
```

Form parametreleri:

- file (zorunlu) - örnek dosya (multipart kodlu dosya içeriği)
- package (isteğe bağlı) - analiz için kullanılacak analiz paketi
- timeout (isteğe bağlı) (int) - analiz süresi aşımı (saniye cinsinden)
- priority (isteğe bağlı) (int) - göreve atanacak öncelik (1-3)
- options (isteğe bağlı) - analiz paketine iletmek için seçenekler
- machine (isteğe bağlı) - analiz için kullanılacak analiz makinesinin etiketi
- platform (isteğe bağlı) - analiz makinesini seçmek için platform adı (örneğin "windows")
- tags (isteğe bağlı) - makineyi başlatmak için etiketlere göre tanımlama. Bu kullanıldığında
- platformun ayarlanmış olması gerekir. Etiketler virgülle ayrılır
- custom (isteğe bağlı) - analiz ve işleme/bildirme modüllerine geçirilmek üzere özel bir dize
- owner (isteğe bağlı) - birden fazla kullanıcının aynı cuckoo örneğine dosya gönderebileceği durumlarda görev sahibi
- clock (isteğe bağlı) - sanal makine saatinin ayarlanması (biçim %m-%d-%Y %H:%M:%S)
- memory (isteğe bağlı) - analiz makinesinin tam bellek dökümü oluşturmayı etkinleştirme
- unique (isteğe bağlı) - daha önce analiz edilmemiş örnekleri yalnızca gönderme
- enforce_timeout (isteğe bağlı) - tam zaman aşımı değerini zorlamak için etkinleştirme

Durum kodları:

- 200 - hata yok
- 400 - yinelenen dosya tespit edildi (unique seçeneği kullanılıyorsa)

/tasks/create/url

POST /tasks/create/url

Bir dosyayı bekleyen görevler listesine ekler. Yeni oluşturulan görevin kimlik bilgilerini döndürür.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" -F url="http://www.malicious.site"
http://localhost:8090/tasks/create/url
```

Python kullanılan örnek request:

```
import requests

REST_URL = "http://localhost:8090/tasks/create/url"
SAMPLE_URL = "http://example.org/malwr.exe"
HEADERS = {"Authorization": "Bearer S4MPL3"}

data = {"url": SAMPLE_URL}
r = requests.post(REST_URL, headers=HEADERS, data=data)

# Add your code to error checking for r.status_code.

task_id = r.json()["task_id"]

# Add your code to error checking if task_id is None.
```

Örnek response:

```
{
   "task_id" : 1
}
```

Form parametreleri:

- url (zorunlu) - analiz edilecek URL (multipart kodlu içerik)
- package (isteğe bağlı) - analiz için kullanılacak paket
- timeout (isteğe bağlı) (int) - analiz süresi (saniye cinsinden)
- priority (isteğe bağlı) (int) - göreve atanacak öncelik (1-3)
- options (isteğe bağlı) - analiz paketine iletilmesi gereken seçenekler
- machine (isteğe bağlı) - analiz için kullanılacak makinenin etiketi
- platform (isteğe bağlı) - analiz makinesini seçmek için platform adı (örneğin "windows")
- tags (isteğe bağlı) - makineyi etiketlere göre başlat. Bu kullanmak için platformun ayarlanmış olması gerekir. Etiketler virgülle ayrılır
- custom (isteğe bağlı) - analiz ve işleme/bildirme modüllerine iletilmek üzere özel bir dize
- owner (isteğe bağlı) - aynı cuckoo örneğine birden fazla kullanıcının dosya gönderebileceği durumda görev sahibi
- memory (isteğe bağlı) - analiz makinesinin tam bellek dökümünün oluşturulmasını etkinleştir
- enforce_timeout (isteğe bağlı) - yürütmenin tam süre boyunca zorlanmasını etkinleştir
- clock (isteğe bağlı) - sanal makine saatinin ayarlanması (format %m-%d-%Y %H:%M:%S)

Durum kodları:

- 200 - hata yok

/tasks/create/submit

POST /tasks/create/submit

Bir veya daha fazla dosyayı ve/veya arşivlere gömülü dosyaları veya yeni oluşturulan görev(ler)in görev ID'lerini içeren bir satırda ayrılmış URL/hash'lerin listesini bekleyen görevlere ekler. Gönderi ID'sini ve yeni oluşturulan görev(ler)in görev ID'lerini döndürür.

Örnek request:

```
# Submit two executables.
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/tasks/create/submit
-F files=@1.exe -F files=@2.exe

# Submit http://google.com
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/tasks/create/submit
-F strings=google.com

# Submit http://google.com & http://facebook.com
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/tasks/create/submit
-F strings=$'google.com\nfacebook.com'
```

Python kullanılan örnek request:

```
import requests

HEADERS = {"Authorization": "Bearer S4MPL3"}

# Submit one or more files.
r = requests.post("http://localhost:8090/tasks/create/submit", files=[
    ("files", open("1.exe", "rb")),
    ("files", open("2.exe", "rb")),
], headers=HEADERS)

# Add your code to error checking for r.status_code.
```

```
submit_id = r.json()["submit_id"]
task_ids = r.json()["task_ids"]
errors = r.json()["errors"]

# Add your code to error checking on "errors".

# Submit one or more URLs or hashes.
urls = [
    "google.com", "facebook.com", "cuckoosandbox.org",
]
r = requests.post(
    "http://localhost:8090/tasks/create/submit",
    headers=HEADERS,
    data={"strings": "\n".join(urls)}
)
```

Örnek response:

```
{
    "submit_id": 1,
    "task_ids": [1, 2],
    "errors": []
}
```

Form parametreleri:

- file (isteğe bağlı) - /tasks/create/file için eski isimle uyumluluk
- files (isteğe bağlı) - kontrol edilecek örnek(ler) ve bekleme sıramıza eklenen örnek(ler)
- strings (isteğe bağlı) - URL'ler ve/veya hash'lerin (VirusTotal API anahtarınızı kullanarak elde edilecek) satırda ayrılmış listesi
- timeout (isteğe bağlı) (int) - analiz süresi sınırlaması (saniye cinsinden)
- priority (isteğe bağlı) (int) - göreve atılacak öncelik (1-3)
- options (isteğe bağlı) - analiz paketi için iletilmesi gereken seçenekler
- tags (isteğe bağlı) - etiketlere göre makine belirtme. Platformun kullanılması gerekiyor. Etiketler virgülle ayrılır
- custom (isteğe bağlı) - analiz ve işleme/bildirime geçirilmek üzere özel dize
- owner (isteğe bağlı) - aynı cuckoo örneğine birden fazla kullanıcının dosya göndermesine izin veriliyorsa görev sahibi
- memory (isteğe bağlı) - analiz makinesinin tam bellek dökümü oluşturmayı etkinleştirme
- enforce_timeout (isteğe bağlı) - yürütmenin tam zaman aşımı değerini zorlamak için etkinleştirme
- clock (isteğe bağlı) - sanal makine saatinin ayarlanması (format %m-%d-%Y %H:%M:%S)

Status kodları:

- 200 - hata yok

/tasks/list

GET /tasks/list/ *(int: limit)* / *(int: offset)*

Görevlerin listesini döndürür.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/tasks/list
```

Örnek response:


```
{
   "tasks": [
      {
         "category": "url",
         "machine": null,
         "errors": [],
         "target": "http://www.malicious.site",
         "package": null,
         "sample_id": null,
         "guest": {},
         "custom": null,
         "owner": "",
         "priority": 1,
         "platform": null,
         "options": null,
         "status": "pending",
         "enforce_timeout": false,
         "timeout": 0,
         "memory": false,
         "tags": []
         "id": 1,
         "added_on": "2012-12-19 14:18:25",
         "completed_on": null
      },
      {
```

```
            "category": "file",
            "machine": null,
            "errors": [],
            "target": "/tmp/malware.exe",
            "package": null,
            "sample_id": 1,
            "guest": {},
            "custom": null,
            "owner": "",
            "priority": 1,
            "platform": null,
            "options": null,
            "status": "pending",
            "enforce_timeout": false,
            "timeout": 0,
            "memory": false,
            "tags": [
                    "32bit",
                    "acrobat_6",
                ],
            "id": 2,
            "added_on": "2012-12-19 14:18:25",
            "completed_on": null
        }
    ]
}
```


Parametreler:

- limit (isteğe bağlı) (int) - döndürülen görevlerin maksimum sayısı
- offset (isteğe bağlı) (int) - veri ofseti

Durum kodları:

- 200 - hata yok

/tasks/sample

GET /tasks/sample/ *(int: sample_id)*

Belirtilen örnek için görev listesini döndürür.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/tasks/sample/1
```

Örnek response:


```
{
   "tasks": [
      {
         "category": "file",
         "machine": null,
         "errors": [],
         "target": "/tmp/malware.exe",
         "package": null,
         "sample_id": 1,
         "guest": {},
         "custom": null,
         "owner": "",
         "priority": 1,
         "platform": null,
         "options": null,
         "status": "pending",
         "enforce_timeout": false,
         "timeout": 0,
         "memory": false,
         "tags": [
                  "32bit",
                  "acrobat_6",
               ],
         "id": 2,
         "added_on": "2012-12-19 14:18:25",
```

```
            "completed_on": null
        }
    ]
}
```

Parametreler:

- sample_id (gereklidir) (int) - görevleri listelemek için örnek kimliği

Durum kodları:

- 200 - hata yok

/tasks/view

GET /tasks/view/ *(int: id)*

Belirtilen ID'ye sahip görevle ilgili ayrıntıları döndürür.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/tasks/view/1
```

Örnek response:


```
{
   "task": {
      "category": "url",
      "machine": null,
      "errors": [],
      "target": "http://www.malicious.site",
      "package": null,
      "sample_id": null,
      "guest": {},
      "custom": null,
      "owner": "",
      "priority": 1,
      "platform": null,
      "options": null,
      "status": "pending",
      "enforce_timeout": false,
      "timeout": 0,
      "memory": false,
      "tags": [
                "32bit",
                "acrobat_6",
            ],
      "id": 1,
      "added_on": "2012-12-19 14:18:25",
      "completed_on": null
```

```
    }
  }
```

Not: Key status için olası değerler:

- pending
- running
- completed
- reported

Parametreler:

- id (gereklidir) (int) - Bakılacak görevin ID'si

Durum kodları:

- 200 - hata yok
- 404 - görev bulunamadı

/tasks/reschedule

GET /tasks/reschedule/ *(int: id)* **/** *(int: priority)*

Belirtilen kimlik ve önceliğe sahip bir görevi yeniden planlayın (varsayılan öncelik 1'dir).

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/tasks/reschedule/1
```

Örnek response:

```
{
   "status": "OK"
}
```

Parametreler:

- Id (gerekli) (int) - Yeniden planlanması gereken görevin kimliği
- Öncelik (isteğe bağlı) (int) - Görev önceliği

Durum kodları:

- 200 - hata yok
- 404 - görev bulunamadı

/tasks/delete

GET /tasks/delete/ *(int: id)*

Verilen görevi veritabanından kaldırır ve sonuçları siler.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/tasks/delete/1
```

Parametreler:

- Id (gerekli) (int) - Silinecek görevin kimliği

Durum kodları:

- 200 - hata yok
- 404 - görev bulunamadı
- 500 - görevi silemiyor

/tasks/report

GET /tasks/report/ *(int: id)* **/** *(str: format)*

Belirtilen görev kimliğiyle ilişkili raporu döndürür.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/tasks/report/1
```

Parametreler:

- Id (gerekli) (int) - Raporu almak için görevin kimliği
- format (isteğe bağlı) - [json/html/all/dropped/package_files] dosyasını almak için raporun biçimi. Hiçbiri belirtilmezse JSON raporu döndürülecektir. hepsi tüm sonuç dosyalarını tar.bz2 olarak döndürür, bırakılan dosyaları tar.bz2 olarak düşürür, paket_files dosyaları analiz paketleri tarafından ana bilgisayara yüklenir.

Durum kodları:

- 200 - hata yok
- 400 - geçersiz rapor formatı
- 404 - rapor bulunamadı

/tasks/summary

GET /tasks/summary/ *(int: id)*

Belirtilen görev kimliğiyle ilişkili yoğunlaştırılmış bir raporu JSON biçiminde döndürür.

Örnek request:

```
curl http://localhost:8090/tasks/summary/1
```

Parametreler:

- Id (gerekli) (int) - Raporu almak için görevin kimliği

Durum kodları:

- 200 - hata yok
- 404 - rapor bulunamadı

/tasks/screenshots

GET /tasks/screenshots/ *(int: id)* **/** *(str: number)*

Belirtilen görev kimliğiyle ilişkili bir veya tüm ekran görüntülerini döndürür.

Örnek request:

```
wget http://localhost:8090/tasks/screenshots/1
```

Parametreler:

- Id (gerekli) (int) - Raporu almak için görevin kimliği
- screenshot (isteğe bağlı) - tek bir ekran görüntüsünün sayısal tanımlayıcısı (örn. 0001, 0002)

Durum kodları:

- 404 - dosya veya klasör bulunamadı

/tasks/rereport

GET /tasks/rereport/ *(int: id)*

Belirtilen görev kimliğiyle ilişkili görev için raporlamayı yeniden çalıştırın.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/tasks/rereport/1
```

Örnek response:

```
{
   "success": true
}
```

Parametreler:

- Id (zorunlu) (int) - Raporu yeniden çalıştıracak görevin kimliği

Durum kodları:

- 200 - hata yok
- 404 - görev bulunamadı

/tasks/reboot

GET /tasks/reboot/ *(int: id)* **

Mevcut bir analiz kimliğinden veritabanına bir yeniden başlatma görevi ekleyin.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/tasks/reboot/1
```

Örnek response:

```
{
   "task_id": 1,
   "reboot_id": 3
}
```

Parametreler:

- Id (gerekli) (int) - Görevin kimliği

Durum kodları:

- 200 - başarılı
- 404 - yeniden başlatma görevi oluşturma hatası

/memory/list

GET /memory/list/ *(int: id)*

Belirtilen görev kimliğiyle ilişkili bellek döküm dosyalarının veya bir bellek döküm dosyasının bir listesini döndürür.

Örnek request:

```
wget http://localhost:8090/memory/list/1
```

Parametreler:

- Id (gerekli) (int) - Raporu almak için görevin kimliği

Durum kodları:

- 404 - dosya veya klasör bulunamadı

/memory/get

GET /memory/get/ *(int: id)* **/** *(str: number)*

Belirtilen görev kimliğiyle ilişkili bir bellek döküm dosyası döndürür.

Örnek request:

```
wget http://localhost:8090/memory/get/1/1908
```

Parametreler:

- Id (gerekli) (int) - Raporu almak için görevin kimliği
- Pid (zorunlu) - tek bir bellek döküm dosyasının sayısal tanımlayıcısı (pid) (örn. 205, 1908)

Durum kodları:

- 404 - dosya veya klasör bulunamadı

/files/view

GET /files/view/md5/ *(str: md5)*

GET /files/view/sha256/ *(str: sha256)*

GET /files/view/id/ *(int: id)*

Belirtilen MD5 hash, SHA256 hash veya kimlikle eşleşen dosyadaki ayrıntıları döndürür.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/files/view/id/1
```

Örnek response:


```
{
   "sample": {
      "sha1": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
      "file_type": "empty",
      "file_size": 0,
      "crc32": "00000000",
      "ssdeep": "3::",
      "sha256":
"e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
      "sha512":
"cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce4
7d0d13c5d85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e",
      "id": 1,
      "md5": "d41d8cd98f00b204e9800998ecf8427e"
   }
}
```


Parametreler:

- md5 (isteğe bağlı) - Aranacak dosyanın MD5 hashi
- sha256 (isteğe bağlı) - Aranacak dosyanın SHA256 hashi

- Id (isteğe bağlı) (int) - Aranacak dosyanın kimliği

Durum kodları:

- 200 - hata yok
- 400 - geçersiz arama terimi
- 404 - dosya bulunamadı

/files/get

GET /files/get/ *(str: sha256)*

Belirtilen SHA256 hashi ile eşleşen dosyanın binary içeriğini döndürür.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3"
http://localhost:8090/files/get/e3b0c44298fc1c149afbf4c8996fb92427ae41e464
9b934ca495991b7852b855 > sample.exe
```

Durum kodları:

- 200 - hata yok
- 404 - dosya bulunamadı

/pcap/get

GET /pcap/get/ *(int: task)*

Verilen görevle ilişkili PCAP içeriğini döndürür.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/pcap/get/1 > dump.pcap
```

Durum kodları:

- 200 - hata yok
- 404 - dosya bulunamadı

/machines/list

GET /machines/list

Cuckoo için mevcut olan analiz makinelerinin ayrıntılarını içeren bir liste döndürür.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/machines/list
```

Örnek response:


```
{
   "machines": [
      {
         "status": null,
         "locked": false,
         "name": "cuckoo1",
         "resultserver_ip": "192.168.56.1",
         "ip": "192.168.56.101",
         "tags": [
                  "32bit",
                  "acrobat_6",
               ],
         "label": "cuckoo1",
         "locked_changed_on": null,
         "platform": "windows",
         "snapshot": null,
         "interface": null,
         "status_changed_on": null,
         "id": 1,
         "resultserver_port": "2042"
      }
   ]
}
```


Durum kodları:

- 200 - hata yok

/machines/view

GET /machines/view/ *(str: name)*

Verilen adla ilişkili analiz makinesindeki ayrıntıları döndürür.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3"
http://localhost:8090/machines/view/cuckoo1
```

Örnek response:


```
{
   "machine": {
      "status": null,
      "locked": false,
      "name": "cuckoo1",
      "resultserver_ip": "192.168.56.1",
      "ip": "192.168.56.101",
      "tags": [
                "32bit",
                "acrobat_6",
            ],
      "label": "cuckoo1",
      "locked_changed_on": null,
      "platform": "windows",
      "snapshot": null,
      "interface": null,
      "status_changed_on": null,
      "id": 1,
      "resultserver_port": "2042"
   }
}
```


Durum kodları:

- 200 - hata yok
- 404 - makine bulunamadı

/cuckoo/status

GET /cuckoo/status/

Cuckoo sunucunun durumunu döndürür. 1.3 sürümünde diskspace girişi eklenmiştir. Diskspace girişi, ilgili dizinlerin bulunduğu diskin kullanılan, boş ve toplam disk alanını gösterir. Diskspace girişi, bir Cuckoo düğümünü Cuckoo API aracılığıyla izleme olanağı sağlar. Unutulmamalıdır ki her dizin ayrı ayrı kontrol edilir, çünkü birisi $CUCKOO/storage/analyses için ayrı bir sabit diske bir sembolik bağ oluşturabilir, ancak $CUCKOO/storage/binaries'yi olduğu gibi bırakabilir. (Bu özellik yalnızca Unix altında kullanılabilir!)

1.3 sürümünde cpuload girişi de eklenmiştir - cpuload girişi, sırasıyla son bir dakika, son 5 dakika ve son 15 dakika için CPU yükünü gösterir. (Bu özellik yalnızca Unix altında kullanılabilir!)

Diskspace dizinleri:

- analyses - $CUCKOO/storage/analyses/
- binaries- $CUCKOO/storage/binaries/
- temporary - tmppath as specified in conf/cuckoo.conf

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/cuckoo/status
```

Örnek response:


```
{
   "tasks": {
      "reported": 165,
      "running": 2,
      "total": 167,
      "completed": 0,
      "pending": 0
   },
   "diskspace": {
      "analyses": {
         "total": 491271233536,
```

```
            "free": 71403470848,
            "used": 419867762688
        },
        "binaries": {
            "total": 491271233536,
            "free": 71403470848,
            "used": 419867762688
        },
        "temporary": {
            "total": 491271233536,
            "free": 71403470848,
            "used": 419867762688
        }
    },
    "version": "1.0",
    "protocol_version": 1,
    "hostname": "Patient0",
    "machines": {
        "available": 4,
        "total": 5
    }
}
```


Durum kodları:

- 200 - hata yok
- 404 - makine bulunamadı

/vpn/status

GET /vpn/status

VPN durumunu döndürür.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/vpn/status
```

Durum kodları:

- 200 - hata yok
- 500 - mevcut değil

/exit

GET /exit

Hata ayıklama modundaysa ve werkzeug sunucusunu kullanıyorsa sunucuyu kapatır.

Örnek request:

```
curl -H "Authorization: Bearer S4MPL3" http://localhost:8090/exit
```

Durum kodları:

- 200 - hata yok
- 403 - Bu çağrı sadece hata ayıklama modunda kullanılabilir
- 500 - error

Distributed Cuckoo

"Analiz" bölümünde belirtildiği gibi, Cuckoo, Distributed Cuckoo kullanımı için bir REST API sağlar. Distributed Cuckoo, örneklerin ve URL'lerin gönderilebileceği tek bir REST API noktası kurmanıza olanak tanır; bu gönderimler daha sonra yapılandırılmış Cuckoo düğümlerinden birine iletilir. Tipik bir kurulum, Distributed Cuckoo'nun çalıştırıldığı bir makine ve bir veya daha fazla makinede Cuckoo daemon ve Cuckoo REST API örneğinin çalıştığı bir veya daha fazla makine içerir. Birkaç not: - En az iki cuckoo node çalıştırıldığında, Distributed Cuckoo kullanmak anlam ifade eder. - Distributed Cuckoo, bir Cuckoo daemon ve REST API'nın da çalıştığı bir makinede çalıştırılabilir, ancak eğer amaç birçok örnek göndermekse yeterli disk alanına sahip olduğundan emin olun.

Distributed REST API Başlatmak

Distributed REST API'nin şu komut satırı seçenekleri bulunmaktadır:

```
$ cuckoo distributed server --help
Usage: cuckoo distributed server [OPTIONS]

Options:
  -H, --host TEXT     Host to bind the Distributed Cuckoo server on
  -p, --port INTEGER  Port to bind the Distributed Cuckoo server on
  --uwsgi             Dump uWSGI configuration
  --nginx             Dump nginx configuration
  --help              Show this message and exit.
```

Yardım çıktısından anlaşılacağı gibi, Distributed Cuckoo'yu başlatmak, basitçe "cuckoo distributed server" komutunu çalıştırmak kadar kolay olabilir.

Çeşitli yapılandırma seçenekleri yapılandırma dosyasında açıklanmış olsa da, daha ayrıntılı açıklamalara da sahibiz. Daha gelişmiş kullanım, doğal olarak uWSGI ve nginx kullanarak dağıtımı içerir.

Distributed Cuckoo Konfigürasyonu

Raporlama Formatları

Raporlama formatları, daha sonra almak istediğiniz raporları belirtir. Ancak, ilişkili raporlar alındıktan sonra Cuckoo nodelarından tüm görevle ilgili veriler kaldırılır; bu, makinelerin disk alanının tükenmemesi için yapılır. Bununla birlikte, bu sizi tüm ilgilendiğiniz rapor formatlarını belirtmeye zorlar, aksi takdirde bu bilgi kaybolacaktır.

Raporlama formatları arasında, ancak bunlarla sınırlı olmamak üzere kendi raporlama formatlarınız da bulunabilir: report.json, report.html, vb.

Samples Dizini

Samples dizini, gönderilen örneklerin ilgili görev silinene kadar geçici olarak depolanacağı dizini belirtir.

Reports Dizini

Samples Dizini gibi, Reports Dizini de raporların alınıp Distibuted REST API'den silinene kadar depolanacağı dizini tanımlar.

RESTful Kaynakları

GET /api/node

Etkin olan tüm nodeları döndürür. Her node için bilgiler ilişkili adı, API URL'sini ve makineleri içerir:

```
$ curl http://localhost:9003/api/node
{
   "success": true,
   "nodes": {
      "localhost": {
         "machines": [
            {
               "name": "cuckoo1",
               "platform": "windows",
               "tags": []
            }
         ],
         "name": "localhost",
         "url": "http://localhost:8090/"
      }
   }
}
```

POST /api/node

Adı ve URL'yi belirterek yeni bir cuckoo node kaydedin:

```
$ curl http://localhost:9003/api/node -F name=localhost \
    -F url=http://localhost:8090/
{
    "success": true
}
```

GET /api/node/<name>

Belirli bir cuckoo node hakkında temel bilgileri alın:

```
$ curl http://localhost:9003/api/node/localhost
{
   "success": true,
   "nodes": [
      {
         "name": "localhost",
         "url": "http://localhost:8090/"
         "machines": [
            {
               "name": "cuckoo1",
               "platform": "windows",
               "tags": []
            }
         ]
      }
   ]
}
```

PUT /api/node/<name>

Bir cuckoo node hakkındaki temel bilgilerini güncelleyin:

```
$ curl -XPUT http://localhost:9003/api/node/localhost -F name=newhost \
   -F url=http://1.2.3.4:8090/
{
   "success": true
}
```

POST /api/node/<name>/refresh

Cuckoo node tarafından ilişkilendirilen, özellikle de makineleri içeren metadata'yı yeniler:

```
$ curl -XPOST http://localhost:9003/api/node/localhost/refresh
{
   "success": true,
   "machines": [
      {
         "name": "cuckoo1",
         "platform": "windows",
         "tags": []
      },
      {
         "name": "cuckoo2",
         "platform": "windows",
         "tags": []
      }
   ]
}
```

DELETE /api/node/<name>

Bir cuckoo node devre dışı bırakın, bunun sonucunda yeni görevleri işleyemez, ancak geçmişini cuckoo distibuted veritabanında tutar:

```
$ curl -XDELETE http://localhost:9003/api/node/localhost
{
   "success": true
}
```

GET /api/task

Veritabanındaki tüm görevlerin bir listesini alın. Sonuçları sınırlamak için bir offset, limit, finished ve owner alanı bulunmaktadır:

```
$ curl http://localhost:9003/api/task?limit=1
{
   "success": true,
   "tasks": {
      "1": {
         "clock": null,
         "custom": null,
         "owner": "",
         "enforce_timeout": null,
         "machine": null,
         "memory": null,
         "options": null,
         "package": null,
         "path": "/tmp/dist-samples/tmphal8mS",
         "platform": "windows",
         "priority": 1,
         "tags": null,
         "task_id": 1,
         "timeout": null
      }
   }
}
```

POST /api/task

Analiz edilmek üzere yeni bir dosya veya URL gönderin:

```
$ curl http://localhost:9003/api/task -F file=@sample.exe
{
   "success": true,
   "task_id": 2
}
```

GET /api/task/<id>

Belirli bir görev hakkında temel bilgiler edinin:

```
$ curl http://localhost:9003/api/task/2
{
   "success": true,
   "tasks": {
      "2": {
         "id": 2,
         "clock": null,
         "custom": null,
         "owner": "",
         "enforce_timeout": null,
         "machine": null,
         "memory": null,
         "options": null,
         "package": null,
         "path": "/tmp/tmpPwUeXm",
         "platform": "windows",
         "priority": 1,
         "tags": null,
         "timeout": null,
         "task_id": 1,
         "node_id": 2,
         "finished": false
      }
   }
}
```

DELETE /api/task/<id>

Bir görevin tüm ilişkili verilerini, yani binary dosyayı, PCAP'i ve raporları silin:

```
$ curl -XDELETE http://localhost:9003/api/task/2
{
   "success": true
}
```

GET /api/report/<id>/<format>

Belirtilen formatta verilen görev için bir rapor alın:

```
# Defaults to the JSON report.
$ curl http://localhost:9003/api/report/2
...
```

GET /api/pcap/<id>

Verilen görev için PCAP'i alır:

```
$ curl http://localhost:9003/api/pcap/2
...
```

Önerilen Kurulum

Aşağıdaki açıklama, iki Cuckoo makinesi, cuckoo0 ve cuckoo1 ile bir Dağıtılmış Cuckoo kurulumunu tasvir eder. Bu kurulumda ilk makine, cuckoo0, aynı zamanda Distributed Cuckoo REST API'yi barındırır.

Konfigürasyon Ayarları

Kurulumumuz, yapılandırma dosyalarıyla ilgili birkaç güncelleme gerektirecektir.

conf/cuckoo.conf

`process_results`'ı kapatmak için güncelleme yapın, çünkü kendi sonuç işleme betiğimizi çalıştıracağız (performans nedenleriyle).

`tmppath`'i birkaç yüz ikili dosyayı depolamak için yeterli alan içeren bir şeye güncelleyin. Bazı sunucularda veya kurulumlarda /tmp kısıtlı bir alan içerebilir, bu yeterli olmayabilir.

Bağlantıyı sqlite3 kullanmak yerine başka bir şeyi kullanacak şekilde güncelleyin. Tercihen PostgreSQL veya MySQL. SQLite3 çoklu iş parçacıklı uygulamaları desteklemez ve bu nedenle Cuckoo gibi sistemler için iyi bir seçenek değildir (mevcut haliyle).

Dağıtılmış cuckoo kurulumu için özel bir veritabanı oluşturmalısınız. Veritabanı betikleriyle güncelleme sorunlarından kaçınmak için mevcut herhangi bir cuckoo veritabanını kullanmaktan kaçının. Yapılandırmada yeni veritabanı adını kullanın. Kullanıcı adları, sunucular vb. gibi kalan yapılandırmaları cuckoo kurulumunuz için aynı tutabilirsiniz. Her bir düğüm için bir DB ve Dağıtılmış Cuckoo'yu çalıştıran makine için bir tane kullanmayı unutmayın (bu "yönetim makinesi" veya "denetleyici" olarak adlandırılır).

conf/processing.conf

Virustotal gibi bazı process modüllerini devre dışı bırakmak isteyebilirsiniz.

conf/reporting.conf

Sisteminizle entegrasyon için hangi rapor(lar)ın gerekli olduğuna bağlı olarak, yalnızca kullanacağın rapor(lar)ı ayarlamak mantıklı olabilir. Böylece diğerlerini devre dışı bırakmak mantıklı olacaktır.

conf/virtualbox.conf

Varsayılan olarak Sanal Makine yöneticisi olarak VirtualBox'u seçmişseniz, modu başsız (headless) olarak değiştirmeniz gerekecek; aksi takdirde bazı sıkıntılarla karşılaşabilirsiniz.

Distributed Cuckoo Kurulumu

Distributed Cuckoo makinesinde Distributed Cuckoo REST API ve Distibuted Cuckoo Worker'ı kurmanız gerekecek.

Daha önce belirtildiği gibi, Distributed Cuckoo REST API, cuckoo distributed server komutunu çalıştırarak veya uWSGI ve nginx ile düzgün bir şekilde dağıtarak başlatılabilir.

Distributed Cuckoo Worker'ı, CWD'de (Cuckoo'yu arka planda çalıştırmak için Cuckoo'ya göre supervisord'ı önce başlatmayı unutmayın) supervisorctl start distributed komutuyla başlatılabilir. Bu, Worker'ı doğru yapılandırma ve argümanlarla otomatik olarak başlatacaktır.

Cuckoo Node Kaydetme

Hızlı kullanımda belirtildiği gibi, Cuckoo nodelarının Dağıtılmış Cuckoo REST API'sine kaydedilmesi gerekir:

```
$ curl http://localhost:9003/api/node -F name=cuckoo0 -F
url=http://localhost:8090/
$ curl http://localhost:9003/api/node -F name=cuckoo1 -F
url=http://1.2.3.4:8090/
```

Cuckoo nodelarını kaydettikten sonra yapmanız gereken tek şey, görevleri göndermek ve tamamlandığında raporları almak. Bu komutlar hakkındaki belgeleri Hızlı Kullanım bölümünde bulabilirsiniz. Cuckoo node `localhost`'ta değilse, `localhost`'u Cuckoo REST API'nin çalıştığı node'un IP adresiyle değiştirin.

Nodelar arasında yük dengelemesi yapmak istiyorsanız, `$CWD/distributed/settings.py` dosyasındaki `threshold` parametresi için daha düşük bir değeri denemek isteyebilirsiniz, çünkü varsayılan değer 500'dür (bu, görevlerin 500'lük gruplar halinde Cuckoo düğümlerine atanması anlamına gelir).

Hızlı Kullanım

Pratik kullanım için aşağıdaki birkaç komut işinizi görecektir.

Bir cuckoo node ve aynı makinede çalışan bir Cuckoo API'si kaydedin:

```
$ curl http://localhost:9003/api/node -F name=localhost -F ip=127.0.0.1
```

Cuckoo node devre dışı bırakmak için:

```
$ curl -XDELETE http://localhost:9003/api/node/localhost
```

Herhangi özel gereksinim olmadan yeni bir analiz görevi gönderin (örneğin, Cuckoo etiketleri, belirli bir makine kullanma, vb.).

```
$ curl http://localhost:9003/api/task -F file=@/path/to/sample.exe
```

Tamamlanmış bir görevin raporunu alın (eğer tamamlanmamışsa, 420 kodlu bir hata alırsınız). Aşağıdaki örnek, varsayılan olarak JSON raporunu kullanacaktır:

```
$ curl http://localhost:9003/api/report/1
```

Bir Cuckoo node takılırsa ve sıfırlanması gerekiyorsa, aşağıdaki adımlar temiz bir şekilde yeniden başlatmak için gerçekleştirilebilir. Bu, SaltStack yapılandırmasının kullanımını ve bazı manuel SQL komutlarını gerektirir (ve tercihen Distibuted Cuckoo Worker geçici olarak devre dışı bırakılmış olmalıdır, yani `supervisorctl stop distributed`):

```
$ psql -c "UPDATE task SET status = 'pending' WHERE status = 'processing' AND
node_id = 123"
$ salt cuckoo1 state.apply cuckoo.clean
$ salt cuckoo1 state.apply cuckoo.start
```

Eğer tüm Cuckoo kümesi bir şekilde kilitlendi, yani tüm görevler 'atanmış', 'işleniyor' veya 'tamamlandı' durumundayken hiçbir Cuckoo node söz konusu analizleri şu anda çalıştırmıyorsa (örneğin, birçok sıfırlama nedeniyle), o zaman tüm durumu sıfırlamak için aşağıdaki adımlar kullanılabilir:

```
$ supervisorctl -c ~/.cuckoo/supervisord.conf stop distributed
$ salt '*' state.apply cuckoo.stop
$ salt '*' state.apply cuckoo.clean
$ psql -c "UPDATE task SET status = 'pending', node_id = null WHERE status IN
('assigned', 'processing', 'finished')"
$ salt '*' state.apply cuckoo.start
$ supervisorctl -c ~/.cuckoo/supervisord.conf start distributed
```

Eğer bir Cuckoo node üzerinde işlenemeyen bir dizi görev varsa ve bu nedenle Cuckoo node tamamen kilitlemişse, o zaman Cuckoo örneklerini hata düzeltilmiş bir sürümle güncelleyip tüm analizleri yeniden işlemek işe yarayabilir:

```
$ salt cuckoo1 state.apply cuckoo.update  # Upgrade Cuckoo.
# To make sure there are failed analyses in the first place.
$ salt cuckoo1 cmd.run "sudo -u cuckoo psql -c \"SELECT * FROM tasks WHERE
status = 'failed_processing'\"
# Reset each analyses to be re-processed.
$ salt cuckoo1 cmd.run "sudo -u cuckoo psql -c \"UPDATE tasks SET status =
'completed', processing = null WHERE status = 'failed_processing'\""
```

Distributed Cuckoo ana bilgisayarını yükseltmek için aşağıdaki adımları gerçekleştirmek isteyebilirsiniz:

```
$ /etc/init.d/uwsgi stop
$ supervisorctl -c ~/.cuckoo/supervisord.conf stop distributed
$ pip uninstall -y cuckoo
$ pip install cuckoo==2.0.0         # Specify your version here.
$ pip install Cuckoo-2.0.0.tar.gz   # Or use a locally archived build.
$ cuckoo distributed migrate
$ supervisorctl -c ~/.cuckoo/supervisord.conf start distributed
$ /etc/init.d/uwsgi start
$ /etc/init.d/nginx restart
```

Tüm Cuckoo kümenizi, yani her Cuckoo nodedaki her makineyi test etmek için, örnek alınabilecek bir `stuff/distributed/cluster-test.py` betiği bulunmaktadır. Mevcut haliyle, kümedeki her yapılandırılmış

makinede aktif bir internet bağlantısını kontrol etmenize olanak tanır. Bu betik, hatalı veya bir şekilde bozulmuş makineleri belirlemek için kullanılabilir. Örnek kullanım aşağıdaki gibi olabilir:

```
# Assuming Distributed Cuckoo listens on localhost and that you want to
# run the 'internet' script (see also the source of cluster-test.py).
$ python stuff/distributed/cluster-test.py localhost -s internet
```

Yardımcı Programlar

Cuckoo, bir dizi yaygın görevi otomatikleştirmek için önceden oluşturulmuş yardımcı programlarla birlikte gelir. Bu yardımcı programlar başlangıçta utils/ dizininde bulunuyordu, ancak artık Cuckoo Apps'a taşındı.

Cuckoo Apps

Bir Cuckoo App aslında sadece bir Cuckoo alt komutudur. Çeşitli Cuckoo App'leri bulunmaktadır, her biri kendi işlevselliğine sahiptir. Her Cuckoo App'ini aynı şekilde çağırabilirsiniz. İşte bazı örnekler:

```
$ cuckoo submit --help
$ cuckoo api --help
$ cuckoo clean --help
```

Bu örneklerde, belirli bir Cuckoo App için işlevselliği ve tüm kullanılabilir parametreleri gösteren `--help` parametresini sağladık.

Submission Utility

Analiz için örnekleri gönderir. Bu araç, Analiz bölümünde açıklanmıştır.

Web Utility

Cuckoo’nun web arayüzü. Bu araç, Web Arayüzü bölümünde açıklanmıştır.

Processing Utility

2.0.0 sürümünde değişiklik: ./utils/process.py'nin rastgele donma sorunları vardı ve ./utils/process2.py'nin yalnızca PostgreSQL tabanlı veritabanlarıyla başa çıkabilme sorunları vardı. Bu iki komut şimdi tek bir Cuckoo App içinde birleştirildi ve artık söz konusu sorunları veya kısıtlamaları göstermiyor.

Daha büyük Cuckoo kurulumları için performans sorunları (çoklu iş parçacığı ve Python GIL ile) nedeniyle sonuç işleme işlemini Cuckoo analizlerinden ayırmak önerilir. cuckoo process kullanarak Cuckoo raporlarını yeniden oluşturmak da mümkündür, bu genellikle Cuckoo İşleme modülleri, Cuckoo İmzaları ve Cuckoo Raporlama modüllerini geliştirirken ve hata ayıklarken kullanılır.

Bir veya daha fazla ayrı işlemde sonuçları işlemek için, `$CWD/conf/cuckoo.conf` dosyasındaki `process_results` yapılandırma öğesini off olarak ayarlayarak devre dışı bırakmanız gerekir. Daha sonra bir Cuckoo İşleme örneği başlatılmalıdır, bu aşağıdaki gibi yapılabilir:

```
$ cuckoo process instance1
```

Eğer gelen tüm analizleri yönetmek için bir Cuckoo İşleme örneği yeterli değilse, sadece ikinci, üçüncü ve mümkünse daha fazla örnek oluşturun:

```
$ cuckoo process instance2
```

Bir analiz görevinin Cuckoo raporunu yeniden oluşturmak için -r anahtarını kullanın:

```
$ cuckoo process -r 1
```

Aynı anda birden çok veya bir aralıktaki Cuckoo raporlarını yeniden oluşturmak da mümkündür. Aşağıdaki örnek, görevleri 1, 2, 5, 6, 7, 8, 9, 10 yeniden işleyecektir:

```
$ cuckoo process -r 1,2,5-10
```

Daha fazla bilgi için bu Cuckoo App hakkındaki yardımı da inceleyin:

```
$ cuckoo process --help
Usage: cuckoo process [OPTIONS] [INSTANCE]

  Process raw task data into reports.

Options:
  -r, --report TEXT       Re-generate one or more reports
  -m, --maxcount INTEGER  Maximum number of analyses to process
  --help                  Show this message and exit.
```

Community Download Utility

Bu Cuckoo App, Cuckoo Topluluk Deposu'ndan Cuckoo İmzalarını, en son izleme ikililerini ve diğer öğeleri indirir ve bunları CWD'nize kurar.

Cuckoo Topluluğu'ndan en son ve en iyi öğeleri almak için sadece aşağıdaki gibi bir komutu yürütün ve bitene kadar bekleyin - şu anda herhangi bir ilerleme göstergesi yoktur:

```
$ cuckoo community
```

Daha fazla kullanım için aşağıdakine bakın:

```
$ cuckoo community --help
Usage: cuckoo community [OPTIONS]

  Utility to fetch supplies from the Cuckoo Community.

Options:
  -f, --force              Overwrite existing files
  -b, --branch TEXT        Specify a different community branch rather than
                           master
  --file, --filepath PATH  Specify a local copy of a community .tar.gz file
  --help                   Show this message and exit.
```

Stats Utility

2.0-rc2 sürümünden itibaren kullanım dışı: Bu yardımcı program, bu bilgiyi hem Cuckoo API hem de Cuckoo Web Arayüzü aracılığıyla almak mümkün olduğu için Cuckoo App'e taşınmayacak.

Machine Utility

2.0.0 sürümünde değişiklik: Bu eskiden bağımsız ve düzensiz bir betikti ve doğrudan Cuckoo konfigürasyonunu değiştiriyordu. Şimdi çok daha iyi entegre edilmiş ve Cuckoo ile oldukça uygun bir şekilde etkileşim kurabilecek.

Machine Cuckoo App, Cuckoo'daki sanal makinelerin yapılandırmasını otomatikleştirmenize yardımcı olmak için tasarlanmıştır. Argüman olarak bir makine ayrıntısı listesi alır ve bunları `cuckoo.conf`'de etkinleştirilmiş olan makina modülü için belirtilen yapılandırma dosyasına yazar. İşte kullanılabilir seçenekler:

```
$ cuckoo machine --help
Usage: cuckoo machine [OPTIONS] VMNAME [IP]

Options:
  --debug              Enable verbose logging
  --add                Add a Virtual Machine
  --delete             Delete a Virtual Machine
  --platform TEXT      Guest Operating System
  --options TEXT       Machine options
  --tags TEXT          Tags for this Virtual Machine
  --interface TEXT     Sniffer interface for this Virtual Machine
  --snapshot TEXT      Specific Virtual Machine Snapshot to use
  --resultserver TEXT  IP:Port of the Result Server
  --help               Show this message and exit.
```

Örnek olarak, Cuckoo'nun yapılandırmasına aşağıdaki gibi bir makine eklenebilir:

```
$ cuckoo machine --add cuckoo1 192.168.56.101 --platform windows --snapshot
vmcloak
```

Distributed Scriptleri

Bu araç Distributed Cuckoo'nda açıklanmıştır.

Mac OS X Bootstrap Scriptleri

2.0.0 sürümünden itibaren kullanım dışı.

Mac OS X analizi için kullanılan bazı başlangıç betikleri `utils/darwin` klasöründe bulunmaktadır; bunlar, Mac OS X kötü amaçlı yazılım analizi için konuk ve ana sistemleri başlatmak için kullanılır. Bazı ayarlar içeride sabit olarak tanımlanmıştır, bu nedenle bunlara göz atmanız ve ihtiyaçlarınıza göre yapılandırmanız önerilir.

Cuckoo Rooter

Cuckoo Rooter, Cuckoo'ya (genel olarak kendisi genellikle non-root olarak çalışır) çeşitli komutlar için `root` erişimi sağlayan yeni bir kavramdır. Bu komut şu anda yalnızca Ubuntu ve Debian benzeri sistemler için kullanılabilir.

Özellikle, `rooter`, Cuckoo'ya analiz başına yönlendirme seçenekleri sağlamak için ağ ile ilgili komutları çalıştırma konusunda yardımcı olur. Bu konuda daha fazla bilgi için lütfen Per-Analysis Network Routing belgesine başvurun. Cuckoo ve `rooter`, Cuckoo'nun ulaşabileceği bir UNIX soketi aracılığıyla iletişim kurar.

Kullanımı aşağıdaki gibidir:

```
$ cuckoo rooter --help
Usage: cuckoo rooter [OPTIONS] [SOCKET]

Options:
  -g, --group TEXT  Unix socket group
  --service PATH    Path to service(8) for invoking OpenVPN
  --iptables PATH   Path to iptables(8)
  --ip PATH         Path to ip(8)
  --sudo            Request superuser privileges
  --help            Show this message and exit.
```

Varsayılan olarak, rooter, Cuckoo'yu kurarken önerildiği gibi UNIX soketi için kullanıcı ve grup olarak cuckoo kullanıcısını kullanacaktır. Cuckoo'yu cuckoo dışında bir kullanıcı altında çalıştırıyorsanız, bunu rooter'a şu şekilde belirtmeniz gerekecektir:

```
$ sudo cuckoo rooter -g <user>
```

Diğer seçenekler oldukça açıktır - belirli Linux komutlarının yollarını belirtebilirsiniz. Ancak varsayılan olarak bunu yapmanıza gerek olmamalıdır, çünkü rooter, bir varsayılan kurulumda çeşitli yardımcı programlar için varsayılan yolları alır.

Virtualenv

Rooter'ın root kullanıcısı olarak çalıştırılması gerektiği gerçeği nedeniyle, bir `virtualenv` kullanırken bazı küçük karmaşıklıklar ortaya çıkar. Daha spesifik olarak, sudo cuckoo rooter komutunu

çalıştırırken `$VIRTUAL_ENV` çevresel değişkeni iletilmeyecektir, bu nedenle Python normalde olduğu gibi aynı virtualenv'den çalıştırılmayacaktır.

Bunu çözmek için, cuckoo ikilisini doğrudan `virtualenv` oturumundan çalıştırmak yeterlidir. Örneğin, virtualenv'iniz `~/venv` konumundaysa, rooter komutunu şu şekilde çalıştırabilirsiniz:

```
$ sudo ~/venv/bin/cuckoo rooter
```

Bunun yerine, `--sudo` parametresini kullanabilirsiniz, bu da doğru cuckoo ikilisine tüm sağlanan bayraklarla sudo çağrısı yapacaktır. Sırasıyla kullanıcı, şifresini girmesi gerekir ve tüm iyi giderse, Cuckoo Rooter düzgün bir şekilde başlatılır, örneğin:

```
(venv)$ cuckoo rooter --sudo
```

Cuckoo Rooter Kullanımı

Cuckoo Rooter'ı kullanmak aslında oldukça kolaydır. Nasıl başlatılacağını biliyorsanız, temelde işe koyulabilirsiniz. Cuckoo, None Routing dışındaki bir yönlendirme seçeneğiyle her analiz için Cuckoo Rooter ile iletişim kursa da, Cuckoo Rooter herhangi bir durumu saklamaz veya belirli bir Cuckoo örneğine bağlanmaz.

Bu nedenle Cuckoo Rooter başlatıldıktan sonra onu bırakabilirsiniz - Cuckoo Rooter, Cuckoo örneğinizi ne sıklıkta yeniden başlatırsanız başlatın, o andan itibaren kendine bakacaktır.

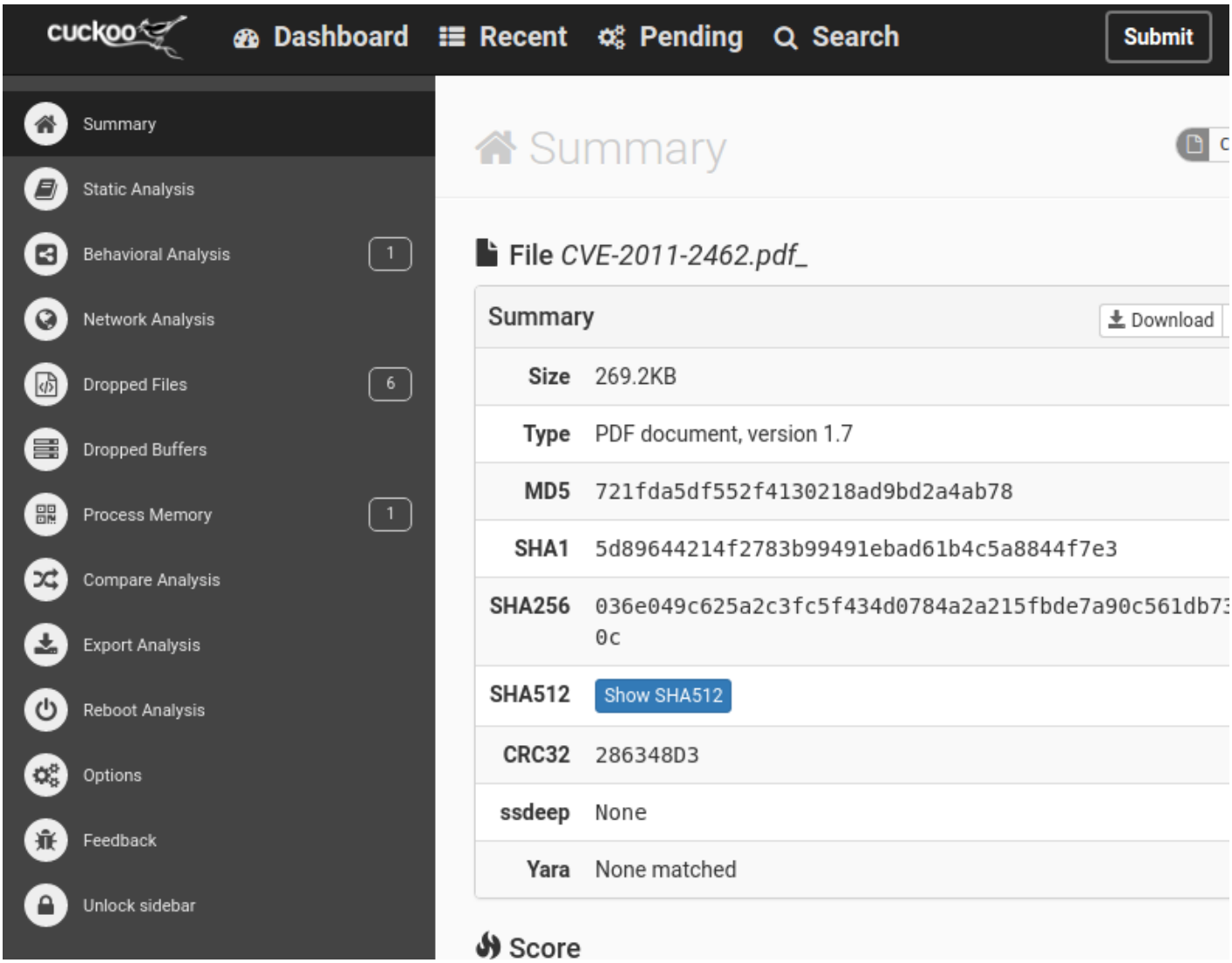
Cuckoo Feedback

2.0.0 sürümünde yeni.

Cuckoo Geri Bildirim formu, kullanıcıların Cuckoo Core Geliştirici ekibine anında geri bildirim sağlamalarına olanak tanır. Bunu yaparak, geliştirme ekibi hatalara, kısmen yanlış analiz sonuçlarına, bir analiz sırasında veya web arayüzünde meydana gelen hatalara ve kullanıcıların ekstra dikkat gerektiren herhangi bir şeye daha hızlı bir şekilde tepki verme olanağına sahip olacaktır. Sonuç olarak, bu isteğe bağlı özellik, ikinci bir görüşe ilgi duyan kullanıcılara, Cuckoo Sandbox'ın arkasındaki takım için kullanıcı ve takım için uygun bir şekilde bunu yapma yeteneği sağlar.

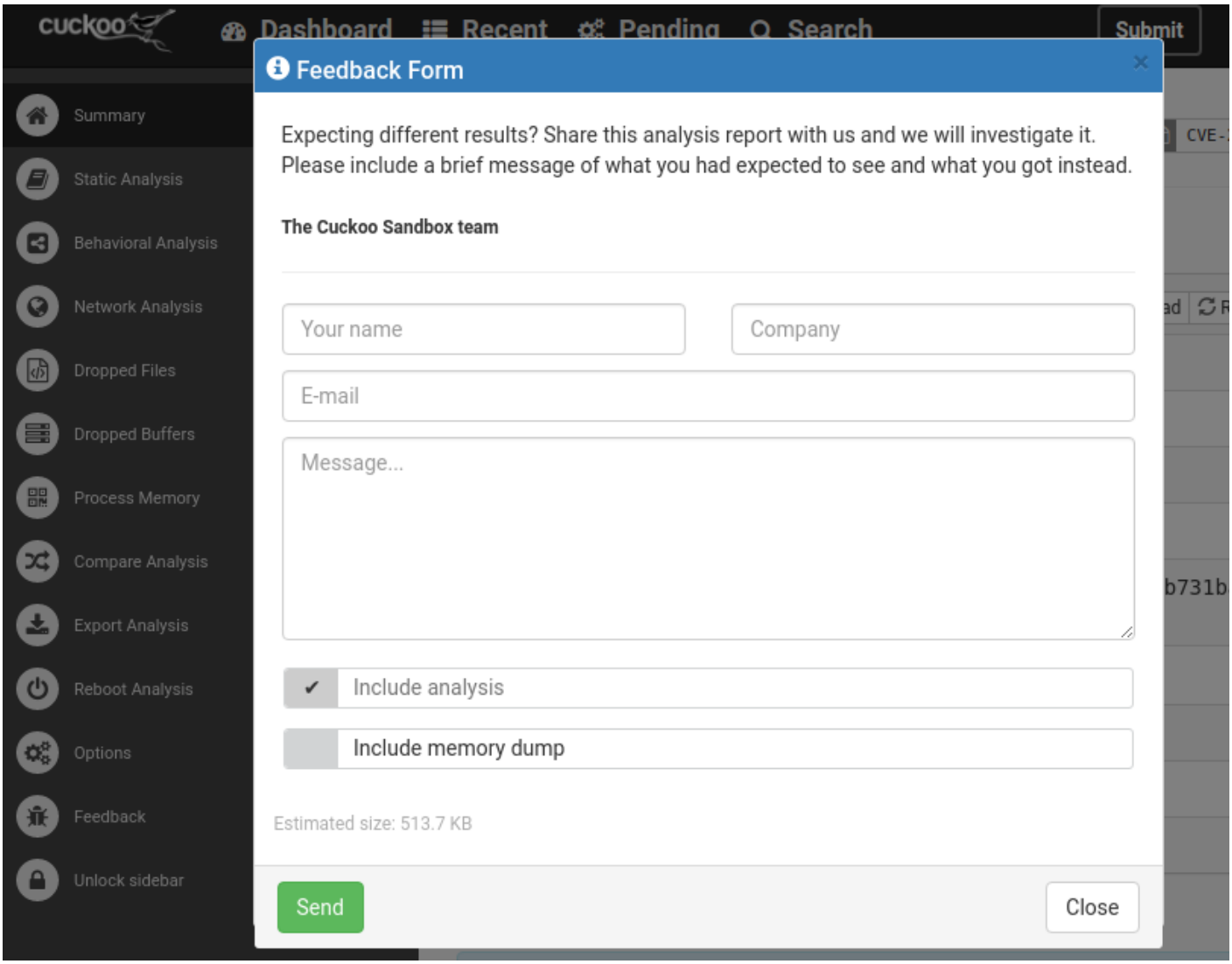
> Kullanıcı olarak, web arayüzünün çoğu sayfasına gömülü olan Cuckoo Geri Bildirim formu aracılığıyla geri bildirimde bulunabilirsiniz (örneğin, bir analiz sayfası veya bir 404 sayfa bulunamadı / 500 iç hata sayfası).

Cuckoo 2.0.0'den itibaren yeni bir analiz sonuçları sayfasının bir bölümünün ekran görüntüsü, kenar çubuğunun kilidinin açık olduğu durumuyla (yani, kalıcı olarak açık) aşağıda bulunmaktadır.



Kenar çubuğunun altında, size aşağıdaki geri bildirim formunu gösterecek olan Feedback düğmesini göreceksiniz. Bu formdaki tüm alanları doldurarak geri bildirimde bulunabilirsiniz.

Düzenli olarak geri bildirim sağlamaya karar verirseniz, adınızı, şirketinizi ve e-posta adresinizi `$CWD/conf/cuckoo.conf` yapılandırma dosyasında doldurabilirsiniz, böylece geri bildirim formunu açtığınızda bu alanlar otomatik olarak doldurulacaktır.


cuckoo
Dashboard
Recent
Pending
Search
Submit
Summary
Static Analysis
Behavioral Analysis
Network Analysis
Dropped Files
Dropped Buffers
Process Memory
Compare Analysis
Export Analysis
Reboot Analysis
Options
Feedback
Unlock sidebar
Feedback Form
Expecting different results? Share this analysis report with us and we will investigate it.
Please include a brief message of what you had expected to see and what you got instead.
The Cuckoo Sandbox team
Your name
Company
E-mail
Message...
Include analysis
Include memory dump
Estimated size: 513.7 KB
Send
Close

Analiz Paketleri

Analiz paketleri, Cuckoo Sandbox'ın temel bir bileşenidir. Bunlar, konuk makinelerde yürütüldüğünde Cuckoo'nun analizör bileşeninin analizi nasıl yürütmesi gerektiğini açıklayan yapılandırılmış Python sınıflarından oluşur.

Cuckoo, kullanabileceğiniz bazı varsayılan analiz paketleri sağlar, ancak kendi paketlerinizi oluşturabilir veya mevcut olanları değiştirebilirsiniz. Bunları `analyzer/windows/modules/packages/` dizininde bulabilirsiniz.

Analiz paketlerine Analiz bölümünde açıklandığı gibi `key1=value1,key2=value2` şeklinde bazı seçenekleri belirtebilirsiniz. Mevcut analiz paketleri zaten etkinleştirilebilecek bazı varsayılan seçenekleri içerir.

Aşağıda, açıkça belirtilmedikçe tüm analiz paketleri için çalışan seçeneklerin bir listesi bulunmaktadır:

- free [yes/no]: etkinleştirildiyse, davranışsal günlükler oluşturulmaz ve kötü amaçlı yazılım serbestçe çalıştırılır.
- procmemdump [yes/no]: etkinleştirildiyse, tüm aktif olarak izlenen süreçlerin bellek dökümlerini alır.
- human 0: devre dışı bırakıldığında, insan benzeri etkileşim (örneğin, fare hareketleri) etkinleştirilmez.

Aşağıda, alfabetik sırayla mevcut paketlerin bir listesi bulunmaktadır:

- applet: Java applet'leri analiz etmek için kullanılır. Seçenekler:
  - class: Yürütülecek sınıfın adını belirtin. Bu seçenek, doğru bir yürütme için zorunludur.
- bin: Genel binary veri, örneğin shellcode'ları analiz etmek için kullanılır.
- cpl: Control Panel Applet'lerini analiz etmek için kullanılır.
- dll: Dinamik Bağlantılı Kütüphaneleri çalıştırmak ve analiz etmek için kullanılır. Seçenekler:
  - function: Yürütülecek işlevi belirtin. Hiçbiri belirtilmezse, Cuckoo DllMain'ı çalıştırmaya çalışacaktır.
  - arguments: DLL'ye komut satırından iletilmesi için argümanları belirtin.
  - loader: Belirli kötü amaçlı yazılımların olası anti-sandbox hilelerini kandırmak için kullanılabilen rundll32.exe'nin yerine kullanılacak bir işlem adını belirtin.
- doc: Microsoft Word belgelerini çalıştırmak ve analiz etmek için kullanılır.
- exe: Genel Windows yürütülebilir dosyalarını analiz etmek için kullanılan varsayılan analiz paketi. Seçenekler:
  - arguments: Gönderilen kötü amaçlı yazılımın başlangıç sürecine iletilmek üzere herhangi bir komut satırı argümanını belirtin.

- generic: cmd.exe aracılığıyla genel örnekleri çalıştırmak ve analiz etmek için kullanılır.
- ie: Verilen URL veya HTML dosyasını açarken Internet Explorer'ın davranışını analiz etmek için kullanılır.
- jar: Java JAR konteynerlerini analiz etmek için kullanılır. Seçenekler:
  - class: Yürütülecek sınıfın yolunu belirtin. Hiçbiri belirtilmezse, Cuckoo Jar'ın MANIFEST dosyasında belirtilen ana işlemleri çalıştırmaya çalışacaktır.
- js: Javascript dosyalarını çalıştırmak ve analiz etmek için kullanılır (örneğin, e-posta eklerinde bulunanlar).
- hta: HTML Uygulama dosyalarını çalıştırmak ve analiz etmek için kullanılır.
- msi: MSI Windows yükleyiciyi çalıştırmak ve analiz etmek için kullanılır.
- pdf: PDF belgelerini çalıştırmak ve analiz etmek için kullanılır.
- ppt: Microsoft PowerPoint belgelerini çalıştırmak ve analiz etmek için kullanılır.
- ps1: PowerShell betiklerini çalıştırmak ve analiz etmek için kullanılır.
- python: Python betiklerini çalıştırmak ve analiz etmek için kullanılır.
- vbs: VBScript dosyalarını çalıştırmak ve analiz etmek için kullanılır.
- wsf: Windows Script Host dosyalarını çalıştırmak ve analiz etmek için kullanılır.
- xls: Microsoft Excel belgelerini çalıştırmak ve analiz etmek için kullanılır.
- zip: Zip arşivlerini çalıştırmak ve analiz etmek için kullanılır. Seçenekler:
  - file: Arşivde bulunan dosyanın adını belirtin. Hiçbiri belirtilmezse, Cuckoo örnek.exe'yi çalıştırmaya çalışacaktır.
  - arguments: Gönderilen kötü amaçlı yazılımın başlangıç sürecine iletilmek üzere herhangi bir komut satırı argümanını belirtin.
  - password: Arşivin şifresini belirtin. Hiçbiri belirtilmezse, Cuckoo arşivi şifresiz çıkarmaya veya "infected" şifresini kullanmaya çalışacaktır.

  Zaten bildiğiniz gibi, analiz paketini seçmek için gönderim sırasında (Analiz'e bakın) aşağıdaki gibi adını belirterek kullanabilirsiniz:

```
$ cuckoo submit --package <package name> /path/to/malware
```

Hiçbiri belirtilmezse, Cuckoo dosya türünü algılamaya çalışacak ve buna göre doğru analiz paketini seçecektir. Dosya türü varsayılan olarak desteklenmiyorsa, analiz durdurulacaktır, bu nedenle mümkünse paket adını belirtmenizi öneririz.

Örneğin, kötü amaçlı yazılımı başlatmak ve bazı seçenekleri belirtmek için şu adımları izleyebilirsiniz:

```
$ cuckoo submit --package dll --options
function=FunctionName,loader=explorer.exe /path/to/malware.dll
```

Analiz Sonuçları

Bir analiz tamamlandığında, çeşitli dosyalar özel bir dizine kaydedilir. Tüm analizler, analiz görevini veritabanında temsil eden artan sayısal ID'ye göre adlandırılmış bir alt dizin içinde `$CWD/storage/analyses/` altında saklanır.

Aşağıda bir analiz dizin yapısının örneği bulunmaktadır:

```
.
|-- analysis.log
|-- binary
|-- dump.pcap
|-- memory.dmp
|-- files
|   |-- 1234567890_dropped.exe
|-- logs
|   |-- 1232.bson
|   |-- 1540.bson
|   `-- 1118.bson
|-- reports
|   |-- report.html
|   |-- report.json
`-- shots
    |-- 0001.jpg
    |-- 0002.jpg
    |-- 0003.jpg
    `-- 0004.jpg
```

analysis.log

Bu, içerideki konuk ortamında gerçekleşen analiz yürütmenin izini içeren analizör tarafından oluşturulan bir günlük dosyasıdır. Bu, süreçlerin, dosyaların oluşturulmasını ve yürütme sırasında meydana gelen olası hataları raporlayacaktır.

dump.pcap

Bu, tcpdump veya diğer ilgili herhangi bir ağ dinleyicisi tarafından oluşturulan ağ dökümüdür.

dump_sorted.pcap

Bu, web arayüzü'nün TCP akışını hızlı bir şekilde aramasına izin veren dump.pcap'in sıralanmış bir versiyonudur.

memory.dmp

Etkinleştirilmiş olmanız durumunda, bu dosya analiz makinesinin tam bellek dökümünü içerir.

files/

Bu dizin, kötü amaçlı yazılımın üzerinde çalıştığı ve Cuckoo'nun dökebildiği tüm dosyaları içerir.

files.json

Bu dosya, mevcut tüm bırakılmış dosyalar için (yani, files/, shots/, vb. içindeki tüm dosyalar) her biri için bir JSON kodlu giriş içerir. Dosya hakkında mevcut olan tüm süreçlerle ilgili meta bilgileri içerir, konukta orijinal dosya yolu vb.

logs/

Bu dizin, Cuckoo'nun süreç izlemesi tarafından oluşturulan tüm raw günlükleri içerir.

reports/

Bu dizin, Konfigürasyon bölümünde açıklandığı gibi Cuckoo tarafından oluşturulan tüm raporları içerir.

shots/

Bu dizin, kötü amaçlı yazılım yürütme sırasında konuğun masaüstünün tüm ekran görüntülerini içerir.

tlsmaster.txt

Bu dosya, analiz sırasında yakalanan TLS Master Secrets'ları içerir. TLS Master Secrets, SSL/TLS trafiğini şifrelemek için kullanılabilir ve bu nedenle HTTPS akışlarını şifrelemek için kullanılır.

Tüm Görevleri ve Örnekleri Temizlemek

Sürüm 2.0.0'de Değişiklik: Bağımsız bir komut dosyası yerine düzgün bir Cuckoo Uygulamasına dönüştürüldü.

Cuckoo 1.2'den bu yana yerleşik bir temizleme özelliği bulunmaktadır. Bu özellik, veritabanındaki görevlerin ve örneklerin tüm ilişkili bilgilerini, sabit diskten, MongoDB'den ve ElasticSearch'ten bırakır. Clean'i çalıştırdıktan sonra bir görev gönderirseniz, `Task #1` ile tekrar başlarsınız.

Temizlemek için aşağıdaki komutu çalıştırın:

```
$ cuckoo clean
```

Özetle, bu komut şunları yapar:

- Analiz sonuçlarını siler.
- Gönderilen binary dosyaları siler.
- Yapılandırılmış veritabanındaki görevlerin ve örneklerin tüm ilişkili bilgilerini siler.
- Yapılandırılmış MongoDB veritabanındaki tüm verileri siler (eğer `$CWD/conf/reporting.conf`'da yapılandırılmış ve etkinse).
- Yapılandırılmış ElasticSearch veritabanındaki tüm verileri siler (eğer `$CWD/conf/reporting.conf`'da yapılandırılmış ve etkinse).

> Bu komutu kullanırsanız, Cuckoo tarafından tüm mevcut depolama alanlarında (dosya sistemi, SQL veritabanı, MongoDB veritabanı ve ElasticSearch veritabanı) depolanan tüm verileri kalıcı olarak sileceksiniz. Bu komutu yalnızca tüm verileri temizleyeceğinizden emin olduğunuzda kullanın.