

# Kurulum

Bu bölüm, Cuckoo'yu nasıl kuracağınızı açıklar. Önerilen kurulum GNU/Linux (tercihen Debian veya Ubuntu) olsa da, Cuckoo'nun Mac OS X ve Microsoft Windows 7 işletim sistemlerinde de sorunsuz çalıştığı kanıtlanmıştır. Windows analizi için önerilen ve test edilen konuk sistemler Windows XP ve 64 bit Windows 7 iken, Mac OS X analizi için Mac OS X Yosemite ve Linux analizi için Debian olarak önerilir.

- **Host'u Hazırlamak**
  - Python Kütüphanelerinin Kurulumu (Ubuntu/Debian tabanlı Linux dağıtımları)
  - Python Kütüphanelerinin Kurulumu (Mac OS X)
  - Sanallaştırma Uygulaması
  - tcpdump Kurulumu
  - Volatility Kurulumu
  - M2Crypto Kurulumu
  - guacd Kurulumu
  - Kullanıcı Oluşturma
  - Dosya Sınırını Arttırma
  - Cuckoo'yu Kurma
  - Cuckoo'yu Dosyadan Kurma
  - Cuckoo Working Directory ve Konfigürasyonu
  - CWD Yolu
  - Konfigürasyon
  - Per-Analysis Network Routing ve Basit Global Routing
  - Per-Analysis Network Routing Ayarları
  - Per-Analysis Network Routing Kullanımı
  - iproute2 Konfigürasyonu
- **Guest'i Hazırlamak**
  - Sanal Makineyi Oluşturmak
  - Python Kurulumu
  - Ek Yazılımlar

- Network Konfigürasyonu
- Agent Kurulumu
- Sanal Makineyi Kaydetmek
- Sanal Makinenin Klonlanması
- Linux Host Kurulumu

# Host'u Hazırlamak

# Python Kütüphanelerinin Kurulumu (Ubuntu/Debian tabanlı Linux dağıtımları)

Cuckoo ana bilgisayar bileşenleri tamamen Python'da yazılmıştır, bu nedenle uygun bir Python sürümünün yüklü olması gereklidir. Şu an sadece Python 2.7 tamamen destekleniyor. Python'ın eski sürümleri ve Python 3 sürümleri desteklenmemektedir.

Cuckoo'nun düzgün bir şekilde yüklenmesi ve çalıştırılması için apt depolarındaki aşağıdaki yazılım paketlerine ihtiyaç vardır:

```
$ sudo apt-get install python python-pip python-dev libffi-dev libssl-dev  
$ sudo apt-get install python-virtualenv python-setuptools  
$ sudo apt-get install libjpeg-dev zlib1g-dev swig
```

Django tabanlı web arayüzünü kullanabilmek için MongoDB gereklidir:

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

Veritabanı olarak PostgreSQL'i kullanmak için, PostgreSQL'in de yüklenmesi gerekecektir:

```
$ sudo apt-get install postgresql libpq-dev
```

KVM'i makine modülü olarak kullanmak isterseniz KVM'i yüklemeniz gerekecektir:

```
$ sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils  
python-libvirt
```

XenServer'ı kullanmak isterseniz XenAPI Python paketini yüklemeniz gerekecektir:

```
$ sudo pip install XenAPI
```

SSL/TLS oluşturulan trafiği engellemek için mitm yardımcı modülünü kullanmak istiyorsanız, mitmproxy'i yüklemeniz gerekecektir. Yükleme talimatları için lütfen mitmproxy web sitesine başvurun. Lütfen unutmayın ki mitmproxy'in en son sürümü Python 3.6 veya daha yüksekini gerektirdiğinden, Cuckoo'nun Python 2.7 ortamından izole etmek için ayrı bir sanal ortamda yüklemek gereklidir. Mitmproxy'i ayrı bir sanal ortamda yükledikten sonra, Cuckoo yapılandırmasına binary yolunu dahil edin, eğer sanal ortam /tmp/mitmproxy3 ise /tmp/mitmproxy3/bin/mitmdump şeklinde ayarlayın.

# Python Kütüphanelerinin Kurulumu (Mac OS X)

Mac OS için kurulum, Ubuntu/Debian'da yapılan kurulumun çoğuyla aynıdır, tek fark brew paket yöneticisini kullanıyor olmamız. Gerekli bağımlılıkları aşağıdaki gibi yükleyin:

```
brew install libmagic cairo pango openssl
```

Buna ek olarak, yara-python'ın başarılı bir şekilde derlenebilmesi için openssl başlık dosyalarını standart GCC/Clang include dizininde bulundurmanız da gerekecektir. Bu aşağıdaki gibi yapılabilir:

```
$ cd /usr/local/include  
$ ln -s ../opt/openssl/include/openssl .
```

# Sanallaştırma Uygulaması

Cuckoo Sandbox, çoğu sanallaştırma yazılımı çözümünü destekler.

Bu rehber için varsayalım ki VirtualBox kurulu (bu varsayılan olarak gelir), ancak bu sandbox'ın yürütülmesi ve genel yapılandırmasını etkilemez.

Sanallaştırma yazılımınızın seçimi, yapılandırılması ve yürütülmesi konusundaki özgürlük tamamen size aittir. VirtualBox'ı tercih etmeye karar vererseniz, dağıtımınıza uygun paketi resmi indirme sayfasından alabilirsiniz. Lütfen Ubuntu LTS makinenize en son VirtualBox sürümünü nasıl kuracağınızı bulmak için aşağıdaki komutları kullanın.

Cuckoo, VirtualBox 4.3, 5.0, 5.1 ve 5.2'yi destekler:

```
$ echo deb http://download.virtualbox.org/virtualbox/debian xenial contrib | sudo  
tee -a /etc/apt/sources.list.d/virtualbox.list  
$ wget -q https://www.virtualbox.org/download/oracle_vbox_2016.asc -O- | sudo  
apt-key add -  
$ sudo apt-get update  
$ sudo apt-get install virtualbox-5.2
```

# tcpdump Kurulumu

Malware'nin yürütülme sırasında gerçekleştirdiği ağ etkinliğini yakalamak ve bu trafiği bir dosyaya dökmek için uygun şekilde yapılandırılmış bir ağ dinleyiciye ihtiyacınız olacaktır.

Varsayılan olarak, Cuckoo, önde gelen açık kaynak çözümü olan tcpdump'ı kullanır.

Ubuntu'da kurulumu yapmak için:

```
$ sudo apt-get install tcpdump apparmor-utils  
$ sudo aa-disable /usr/sbin/tcpdump
```

AppArmor profil devre dışı bırakma (aa-disable komutu) yalnızca varsayılan CWD dizinini kullandığınızda gereklidir, aksi takdirde AppArmor gerçek PCAP dosyalarının oluşturulmasını engeller (bkz. [tcpdump Permission Denied hatası](#)).

AppArmor devre dışı bırakılmış (örneğin Debian gibi) Linux platformları için tcpdump'u kurmak için aşağıdaki komut yeterlidir:

```
$ sudo apt-get install tcpdump
```

Tcpdump, root ayrıcalıklarını gerektirir, ancak Cuckoo'nun root yetkileriyle çalışmasını istemediğiniz için binary dosyasına belirli Linux yetkileri ayarlamalısınız:

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

Son komutun sonuçlarını şu şekilde doğrulayabilirsiniz:

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



Eğer setcap yüklü değilse, aşağıdaki komutla yükleyebilirsiniz:

```
$ sudo apt-get install libcap2-bin
```

Aksi takdirde (tavsiye edilmez), aşağıdaki komutu kullanabilirsiniz:

```
$ sudo chmod +s /usr/sbin/tcpdump
```

Lütfen unutmayın ki setcap yöntemi dahi, potansiyel güvenlik açıkları nedeniyle, sistemin diğer güvenilir olmayan potansiyel kullanıcılarına sahip olduğu durumlarda tamamen güvenli değildir. Cuckoo'yu ayrılmış bir sistemde veya ayrıcalıklı tcpdump yürütmesinin başka şekilde sınırlı olduğu güvenilir bir ortamda çalıştırmanızı tavsiye ederiz.

# Volatility Kurulumu

Volatility, bellek döküntülerinde adli analiz yapmak için isteğe bağlı bir araçtır. Cuckoo ile birleştirildiğinde, işletim sisteminde derin değişikliklere ilişkin ek görünürlük sağlayabilir ve Cuckoo'nun analizörünün izleme alanından kaçan kökkit teknolojisinin varlığını tespit edebilir.

Doğru çalışabilmesi için Cuckoo'nun en azından Volatility 2.3 sürümünü gerektirdiğini, ancak en son sürümü olan Volatility 2.5'i önerdiğini unutmayın. Resmi depolarından indirebilirsiniz.

Kurulum talimatları için Volatility belgelerine başvurun.

Host'u Hazırlamak

# M2Crypto Kurulumu

M2Crypto kütüphanesi şu anda yalnızca SWIG yüklü olduğunda desteklenmektedir. Ubuntu/Debian benzeri sistemlerde bu işlem aşağıdaki gibi gerçekleştirilebilir:

```
$ sudo apt-get install swig
```

Sistemde SWIG bulunuyorsa, M2Crypto aşağıdaki gibi yüklenir:

```
$ sudo pip install m2crypto==0.24.0
```

# guacd Kurulumu

guacd, Cuckoo web arayüzündeki uzaktan kontrol işlevselliği için RDP, VNC ve SSH çeviri katmanını sağlayan isteğe bağlı bir hizmettir.

Bu olmadan, uzaktan kontrol çalışmaz. 0.9.9 sürümü ve üzeri sürümler çalışacaktır, ancak en son sürümü yüklemenizi öneririz. Ubuntu 17.04 makinesinde aşağıdaki komut, 0.9.9-2 sürümünü yükleyecektir:

```
$ sudo apt install libguac-client-rdp0 libguac-client-vnc0 libguac-client-ssh0  
guacd
```

Sadece RDP desteği istiyorsanız, libguac-client-vnc0 ve libguac-client-ssh0 paketlerinin kurulumunu atlayabilirsiniz.

Eğer eski bir dağıtım kullanıyorsanız veya sadece en son sürümü kullanmak istiyorsanız (önerimiz), aşağıdaki komut en son sürümü (0.9.14) kaynaktan derleyecektir:

```
$ sudo apt -y install libcairo2-dev libjpeg-turbo8-dev libpng-dev liboss-p-uid-dev  
libfreerdp-dev  
$ mkdir /tmp/guac-build && cd /tmp/guac-build  
$ wget https://www.apache.org/dist/guacamole/0.9.14/source/guacamole-server-  
0.9.14.tar.gz  
$ tar xvf guacamole-server-0.9.14.tar.gz && cd guacamole-server-0.9.14  
$ ./configure --with-init-dir=/etc/init.d  
$ make && sudo make install && cd ..  
$ sudo ldconfig  
$ sudo /etc/init.d/guacd start
```

Kaynaktan kurulum yaparken, paket yöneticinizden herhangi bir libguac kütüphanesinin başka bir sürümünün yüklü olmadığından emin olun; aksi takdirde uyumsuzluklardan kaynaklanan sorunlarla karşılaşabilir ve guacd'nin çökmesine neden olabilir.

Ayrıca, Guacamole tarafından açığa çıkarılan Cuckoo Control işlevselliğinden yararlanmak için VirtualBox Extension Pack'in de kurulu olması gerektiğini unutmayın.

# Kullanıcı Oluřturma

Cuckoo'yu kendi kullanıcınızdan alıřtırabilir veya yalnızca sandbox kurulumunuz iin ayrılan yeni bir kullanıcı oluřturabilirsiniz. Cuckoo'yu alıřtıran kullanıcının, sanal makineleri oluřturup alıřtırmak iin kullanacaėınız kullanıcı ile aynı olmasını saėlayın (en azından VirtualBox durumunda), aksi takdirde Cuckoo, bu sanal makineleri tanımlayıp bařlatamaz.

Yeni kullanıcı oluřturmak iin:

```
$ sudo adduser cuckoo
```

Eėer VirtualBox kullanıyorsanız, yeni kullanıcının "vboxusers" grubuna (veya VirtualBox'u alıřtırmak iin kullandıėınız grubuna) ait olduėundan emin olun:

```
$ sudo usermod -a -G vboxusers cuckoo
```

Eėer KVM veya bařka bir libvirt tabanlı modl kullanıyorsanız, yeni kullanıcının "libvirtd" grubuna (veya Linux daėıtımınızın libvirt'i alıřtırmak iin kullandıėı gruba) ait olduėundan emin olun:

```
$ sudo usermod -a -G libvirtd cuckoo
```

Host'u Hazırlamak

# Dosya Sınırını Arttırma

SSS girdisinde belirtildiđi gibi [IOError: \[Errno 24\] Too many open files hatası](#) ile karşılaşılmaması durumunda, Cuckoo'yu başlatmadan önce dosya sayısı sınırlarını arttırmak istenebilir; aksi takdirde, işletim sistemi tarafından izin verilenden daha fazla dosya açılması nedeniyle bazı örneklerin raporu düzgün işlenemeyebilir.

# Cuckoo'yu Kurma

Cuckoo'nun en son sürümünü kurmak oldukça basittir. Ancak, Cuckoo'yu kurmaya çalışırken genellikle eski sürümlerle karşılaşılan sorunları önlemek için ilk olarak `pip` ve `setuptools` kütüphanelerini güncellemek önerilir (ayrıca [DistributionNotFound / No distribution matching the version.. hatası](#) için bakınız).

Çeşitli Python bağımlılıklarını oluşturmak için gerekli olan bir veya daha fazla sistem paketinin eksik olma olasılığı oldukça yüksektir. Bu tür sorunları çözmek için [gereksinimleri](#) tekrar okuyun.

```
$ sudo pip install -U pip setuptools
$ sudo pip install -U cuckoo
```

Yukarıdaki yöntemle, işletim sisteminizde Cuckoo'nun genel bir kurulumu genellikle sorunsuz çalışır, ancak Cuckoo'yu bir `virtualenv`'e kurmanızı şiddetle öneririz. `virtualenv`'e kurmak için:

```
$ virtualenv venv
$ . venv/bin/activate
(venv)$ pip install -U pip setuptools
(venv)$ pip install -U cuckoo
```

Environment'a (virtualenv'ler vb.) bağlı olarak, pip'in hangi sürümünü kullanmanız gerektiğini belirtmeniz gerekebilir. Yukarıdaki komutlardaki pip'i sadece pip2 ile değiştirin.

`virtualenv` kullanmanın bazı nedenleri:

- Cuckoo'nun bağımlılıkları tamamen güncel olmayabilir, ancak bilinen bir şekilde düzgün çalışan bir sürüme sabitlenebilir.
- Sistem üzerinde yüklenmiş diğer yazılımların bağımlılıkları, uyumsuz sürüm gereksinimleri nedeniyle Cuckoo'nun gereksinim duyduklarıyla çakışabilir (ve evet, Cuckoo en son sürümü destekliyor olsa bile, diğer yazılım basitçe daha eski bir sürüme sabitlenmiş olabilir).

- `Virtualenv`, kök kullanıcı olmayan kullanıcılara ek paketler kurma veya Cuckoo'yu daha sonradan güncelleme olanağı sağlar.

Basitçe söylemek gerekirse, `virtualenv` genel olarak en iyi uygulama olarak kabul edilir.



# Cuckoo'yu Dosyadan Kurma

Cuckoo Paketi'nin bir kopyasını indirip çevrimdışı olarak kurarak, Cuckoo'yu önbellekteki bir kopyayı kullanarak kurabilir ve/veya gelecekteki mevcut Cuckoo sürümlerinin bir yedek kopyasına sahip olabilirsiniz. Ayrıca, böyle bir tarball'ı web sitemizden indirme seçeneği de bulunmaktadır.

Cuckoo ve tüm bağımlılıklarının tarball'ını manuel olarak edinmek şu şekilde yapılabilir:

```
$ pip download cuckoo
```

`Cuckoo-2.0.0.tar.gz` (veya en son yayınlanan kararlı sürüme bağlı olarak daha yüksek bir numara) adında bir dosya elde edeceksiniz, ayrıca tüm bağımlılıkları da içerir (örneğin, `alembic-0.8.8.tar.gz`).

Tam olarak bu sürümü kurmak, pip'i doğrudan kullanarak kurulum yaparken aşına olduğunuz şekilde yapılabilir, ancak bu sefer tarball'ın dosya adını kullanarak:

```
$ pip install Cuckoo-2.0.0.tar.gz
```

İnternet bağlantısının olmadığı sistemlerde, \$ pip download cuckoo komutu, gerekli tüm bağımlılıkları getirmek için kullanılabilir ve bu şekilde teorik olarak Cuckoo'yu tamamen çevrimdışı olarak bu dosyaları kullanarak kurmak mümkün olmalıdır, yani şöyle bir şeyi yürütme yeteneğine sahip olmalısınız:

```
$ pip install *.tar.gz
```

- Yapılandırma
- Cuckoo İmzaları
- Cuckoo Analizörü
- Cuckoo Ajanı
- Yara kuralları
- Cuckoo Depolama (analiz sonuçlarının gittiği yer)
- Ve daha fazlası...

Eğer Cuckoo kurulumunuzu daha yeni bir sürüme güncellediyseniz, yapılandırmanızın yedeğini almanız, Cuckoo örneğinizi güncellemeniz ve yapılandırmanızı ya geri yüklemeniz ya da tamamen yeniden uygulamanız gereken bir sorunla karşılaşmış olabilirsiniz.

Cuckoo'yu ilk kez çalıştırdığınızda, **CWD** kontrolü sizin için otomatik olarak oluşturulur, bu aşağı yukarı şu şekilde gerçekleşir:

$\bar{\Lambda} \backslash$	$\bar{\Lambda} \_ \backslash$	$\bar{\Lambda} \backslash$	$\bar{\Lambda} \_ \backslash$	$\bar{\Lambda} \backslash$	$\bar{\Lambda} \backslash$
$/ \backslash \backslash$	$///$	$\_ / \backslash \backslash$	$/// \_$	$/ \backslash \backslash$	$/ \backslash \backslash$
$/ \Lambda \backslash \backslash \backslash \_$	$\Lambda \_ / \Lambda \backslash \backslash$	$/// \Lambda \_$	$/ \Lambda \backslash \backslash$	$/ \Lambda \backslash \backslash$	
$/// \Lambda \backslash \backslash \_ \backslash$	$//// / \Lambda \backslash \backslash$	$/// \_ ///$	$/// \Lambda \backslash \backslash$	$/// \Lambda \backslash \backslash$	
$/// \backslash \_ \backslash \_ /$	$//// / \backslash \_ \backslash$	$/ \Lambda \_ \_ /$	$/// \backslash \_ \backslash / / \backslash \_ \backslash$		
$/// \vee \_ / ///$	$//// / \vee \_ / \Lambda \_ \_ /$	$/// / / / / / /$	$///$		
$///$	$/// / / / / /$	$/// \Lambda \backslash \backslash$	$/// / / / / / /$	$///$	

```
///_____ ///_// ///_____ /// \\ \\ ///_// ///_//  
///_____V///_V///_V///_V/// \\ \\ ///_V///_V/  
V_____N_____/V_____N_/ \\_\\V_____/V_____/
```

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

=====  
=====

Welcome to Cuckoo Sandbox, this appears to be your first run!  
We will now set you up with our default configuration.  
You will be able to modify the configuration to your likings  
by exploring the /home/cuckoo/.cuckoo directory.

Among other configurable things of most interest is the  
new location for your Cuckoo configuration:

/home/cuckoo/.cuckoo/conf

=====  
=====

Cuckoo has finished setting up the default configuration.  
Please modify the default settings where required and  
start Cuckoo again (by running `cuckoo` or `cuckoo -d`).

Bilgi mesajları tarafından belirtildiği gibi, `CWD`'nizi artık varsayılan olarak `~/cuckoo` şeklinde bulabileceksiniz, yani `/home/cuckoo/.cuckoo`. Bildiğiniz gibi tüm yapılandırma dosyalarını `$CWD/conf` dizininde bulabilirsiniz. Yani, `$CWD/conf/cuckoo.conf`, `$CWD/conf/virtualbox.conf` vb.

Şimdi, `CWD` dizini Cuckoo'nun kendisinin bir parçası olmadığından, yani Git deposunun veya en son sürümlerden birinin bir parçası olmadığından, `CWD`'yi ellemeye gerek kalmadan Cuckoo'yu yükseltebileceksiniz. (Tabii ki, güncellenmiş bir Yapılandırma gerektiren bir güncelleme yüklenirse, yapılandırma dosyalarını kendisi üzerine yazmak yerine Cuckoo kullanıcıyı bununla ilgili olarak yönlendirecektir).

# CWD Yolu

Her ne kadar `CWD` varsayılan olarak `~/cuckoo` olarak belirlenmiş olsa da, bu yol tamamen yapılandırılabilir. Aşağıda, Cuckoo'nun `CWD`'yi belirlemek için öncelik sırası listelenmiştir.

- "`--cwd` komut satırı seçeneği aracılığıyla (örneğin, `--cwd ~/cuckoo`)."
- "`CUCKOO` ortam değişkeni aracılığıyla (örneğin, `export CUCKOO=~/cuckoo`)."
- "`CUCKOO_CWD` ortam değişkeni aracılığıyla."
- "Eğer mevcut dizin bir `CWD` ise (örneğin, `cd ~/cuckoo`, bu dizinde bir `CWD` oluşturulduğunu varsayarsak)."
- "Varsayılan olarak, `~/cuckoo`."

Farklı `CWD` yollarını kullanarak, aynı Cuckoo kurulumunu kullanarak farklı yapılandırmalara sahip birden fazla Cuckoo örneğini çalıştırmak mümkündür. Örneğin, Windows analizi ve Android analizi gibi iki veya üç ayrı Cuckoo kurulumuna ihtiyaç duyulursa, her güncelleme olduğunda her örneği tek tek yükseltme zorunluluğu olmamak kesinlikle büyük bir adımdır.

CWD'nin nasıl yapılandırılacağını göstermek için bazı örnekler aşağıda verilmiştir:

```
# Places the CWD in /opt/cuckoo. Note that Cuckoo will normally create the
# CWD itself, but in order to create a directory in /opt root capabilities
# are usually required.
$ sudo mkdir /opt/cuckoo
$ sudo chown cuckoo:cuckoo /opt/cuckoo
$ cuckoo --cwd /opt/cuckoo

# You could place this line in your .bashrc, for example.
$ export CUCKOO=/opt/cuckoo
$ cuckoo
```

# 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 eşleştikten 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

[dumpts]  
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
```

# Per-Analysis Network Routing ve Basit Global Routing

Cuckoo `2.0-rc1`'den itibaren per-analysis network routing özelliği bulunmaktadır. Başka bir deyişle, bir VM'niz varsa ve analiz edilecek üç örnek varsa, ilk analiz için internet erişimini engellemek, ikinci analizi bir VPN üzerinden yönlendirmek ve üçüncü analizi Tor ağı üzerinden çekmek mümkündür.

Ancak, daha gelişmiş analiz başına yönlendirme dışında, daha önce lüks yönlendirme henüz mevcut olmadığına popüler olan bir varsayılan rota da mümkündür.

Örneklerimizde, varsayılan makina seçeneğimiz olduğu için `VirtualBox`'a odaklanacağız.

Daha karmaşık ve özellik açısından zengin per-analysis network routing'e girmeden önce, bir kez ayarlandığında değiştirilmeyen global `iptables` kurallarına dayanan daha eski bir yaklaşımı ilk ele alacağız.

Aşağıdaki kurulumda, VirtualBox VM'imize atanan arayüzün `vboxnet0` olduğunu, VM'nin IP adresinin `192.168.56.101` olduğunu (bir `/24` alt ağda), internete bağlı çıkış arayüzünün `eth0` olduğunu varsayıyoruz. Bu kurulumla, aşağıdaki `iptables` kuralları, VM'lerin Cuckoo ana makinesine (bu kurulumda `192.168.56.1`) ve internete tam erişim sağlamasına izin verecektir, bu da internete bağlanan herhangi bir uygulamadan beklediğiniz gibi.

```
$ sudo iptables -t nat -A POSTROUTING -o eth0 -s 192.168.56.0/24 -j MASQUERADE

# Default drop.
$ sudo iptables -P FORWARD DROP

# Existing connections.
$ sudo iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

```
# Accept connections from vboxnet to the whole internet.
$ sudo iptables -A FORWARD -s 192.168.56.0/24 -j ACCEPT

# Internal traffic.
$ sudo iptables -A FORWARD -s 192.168.56.0/24 -d 192.168.56.0/24 -j ACCEPT

# Log stuff that reaches this point (could be noisy).
$ sudo iptables -A FORWARD -j LOG
```

Bu kurallar ayarlandığında neredeyse başlamaya hazırız. Ancak, bu kurallar, çekirdekte IP yönlendirmesi açıkça etkinleştirilmediği sürece herhangi bir paket yönlendirmesi yapmaz. Bunu yapmak için, geçici bir yöntem ve kapatma veya yeniden başlatma kadar süren geçici bir yöntem bulunmaktadır. Basitçe söylemek gerekirse, genellikle bu iki komutu çalıştırmanız gerekecektir:

```
$ echo 1 | sudo tee -a /proc/sys/net/ipv4/ip_forward
$ sudo sysctl -w net.ipv4.ip_forward=1
```

Iptables kuralları, yeniden başlatmalar arasında kalıcı değildir, bu nedenle onları korumak istiyorsanız bir komut dosyası kullanmalısınız veya sadece `iptables-persistent` kurmalısınız.

Daha yeni Linux dağıtımları, udev'in arayüz adlandırma düzenini benimsemiştir. Bu, `eth0`'ın artık ana arayüz olmayabileceği anlamına gelir. Olası arayüz adları, `ensXX`, `enp0sXX` ve `emXX`'i içerir, burada `XX` kısmı bir numarayı tanımlar. Bu, yukarıdaki NAT ifadesi için özellikle önemli bir nottur.

# Per-Analysis Network Routing Ayarları

Bir ağ arayüzü üzerinden analizleri yönlendirmenin old school yöntemini tartıştıktan sonra, şimdi çok daha ayrıntılı ağ yönlendirmesine izin veren dinamik ağ yönlendirme bileşenlerini inceleyeceğiz.

Dokümantasyonun bu bölümünün girişinde belirtildiği gibi, Cuckoo 2.0-rc1'den itibaren, Cuckoo Rooter tanıtıldığında, per-analysis network routing yapmak mümkün olmuştur. O zamandan beri çeşitli hatalar çözülmüş ve daha fazla ağ yönlendirme seçeneği eklenmiştir.

Aşağıda mevcut yönlendirme seçeneklerinin bir listesi bulunmaktadır.

Routing Seçeneği	Açıklama
None Routing	Hiçbir yönlendirme yapılmaz, Cuckoo Rooter'ın çalıştırılmasını gerektirmeyen (ve bu nedenle aynı zamanda varsayılan yönlendirme seçeneği olan) tek seçenek.
Drop Routing	Tüm Cuckoo trafiğini, VM'lerin alt ağındaki trafiği de içeren tüm trafiği tamamen engeller.
Internet Routing	Verilen ağ arayüzü tarafından sağlanan tam internet erişimi (Basit Global Routing kurulumuna benzer).
InetSim Routing	Tüm trafiği, ana makinede çalışan sahte hizmetler sağlayan bir InetSim örneğine yönlendirir.
Tor Routing	Tüm trafiği Tor üzerinden yönlendirir.
VPN Routing	Tüm trafiği belki de çoklu önceden tanımlanmış VPN uç noktalarından biri üzerinden yönlendirir.

## None Routing

Cuckoo'nun üçüncü taraf tarafından tanımlanan şekilde analizi yönlendirmesine izin veren anlamda varsayılan yönlendirme mekanizması. Yani, gerçekten hiçbir şey yapmaz. Bir kişi, None Routing'i Basit Global Routing ile birlikte kullanabilir.

## Drop Routing



Drop routing seçeneği, temelde hiçbir global `iptables` kuralı oluşturulmamış bir makinede (VM'lere tam internet erişimi sağlayan veya benzeri) varsayılan None Routing kurulumuna biraz benzemektedir, ancak VM'lere sağlanan internet erişimini etkin bir şekilde kapatmak konusunda çok daha agresiftir.

Drop routing ile mümkün olan tek trafik içsel Cuckoo trafiğidir ve bu nedenle DNS istekleri veya çıkış `TCP/IP` bağlantıları engellenir.

## Internet Routing

İnternet yönlendirmeyi kullanarak, VM'lere bağlı ağ arayüzlerinden biri aracılığıyla tam internet erişimi sağlanabilir. Bu seçeneği, tüm potansiyel olarak kötü amaçlı örneklerin aynı yükseltici üzerinden internete bağlanmasına izin verdiği doğası nedeniyle kirli hat olarak da adlandırıyoruz.

Kirli hat ağ arayüzünü, `iproute2` ile açıklandığı gibi kaydetmek gerekmektedir.

## InetSim Routing

InetSim hakkında bilgi sahibi olmayanlar için, InetSim, kötü amaçlı yazılımların iletişim kurması için sahte hizmetler sağlayan bir projedir. InetSim yönlendirmesini kullanmak için, InetSim'i ana makinede (veya ayrı bir VM'de) kurmanız ve Cuckoo'yu, InetSim sunucusunu nerede bulacağını bilmesi için yapılandırmanız gerekecektir.

InetSim için yapılandırma kendinden açıklamalıdır ve `$CWD/conf/routing.conf` yapılandırma dosyasının bir parçası olarak bulunabilir:

```
[inetsim]
enabled = yes
server = 192.168.56.1
```

InetSim ile hızlı bir başlangıç yapmak için REMnux dağıtımının en son sürümünü indirmek mümkündür; bu dağıtım, birçok diğer aracın yanı sıra InetSim'in en son sürümünü içermektedir. Bu VM'nin doğal olarak kendi sabit IP adresine ihtiyacı olacak ve ardından bu adres, `routing.conf` yapılandırma dosyasında yapılandırılmalıdır.

## Tor Routing

Tor'un kötü amaçlı yazılım analizi için kullanılması kesinlikle tavsiye edilmiyor.

Öncelikle Tor'un kurulması gerekmektedir. Tor'un en son kararlı sürümünü yüklemek için [buradaki](#) talimatları bulabilirsiniz.

Ardından Tor yapılandırma dosyasını değiştirmemiz gerekecek (Henüz Cuckoo'nun Tor için yapılandırması hakkında konuşmuyoruz!). Bunun için, Tor'a TCP/IP bağlantıları ve UDP istekleri için dinleme adresi ve bağlantı noktasını sağlamamız gerekecek. Varsayılan bir VirtualBox kurulumunda, ana makinenin IP adresi 192.168.56.1 ise, aşağıdaki satırlar /etc/tor/torrc dosyasında yapılandırılmalıdır:

```
TransPort 192.168.56.1:9040
DNSPort 192.168.56.1:5353
```

Tor'u yeniden başlatmayı unutmayın ( `/etc/init.d/tor restart` ). Bu bizi Cuckoo için Tor yapılandırmasıyla bırakır, bu dosya \$CWD/conf/routing.conf dosyasında bulunabilir:

```
[tor]
enabled = yes
dnsport = 5353
proxyport = 9040
```

Unutulmaması gereken bir nokta, `/etc/tor/torrc` ve `$CWD/conf/routing.conf` dosyalarındaki bağlantı noktası numaralarının, ikisinin de doğru şekilde etkileşimde bulunabilmesi için eşleşmesi gerektiğidir.

## VPN Routing

Son olarak, analizleri bir dizi VPN üzerinden yönlendirmek mümkündür. Birkaç farklı ülkede sona eren birkaç VPN tanımlayarak, potansiyel olarak kötü amaçlı örneklerin IP adresinin ülkesine bağlı olarak farklı davranıp davranmadığını görmek mümkün olabilir.

VPN için yapılandırma, bir VM'nin yapılandırması gibi. Her VPN için `$CWD/conf/routing.conf` yapılandırma dosyasında ilgili bilgileri detaylandıran bir bölüme ihtiyacınız olacak. Yapılandırmada VPN, mevcut VPN'lerin listesine kaydedilmelidir (tam olarak daha fazla VM kaydetmek için yapacağınız gibi).

Tek bir VPN için yapılandırma yaklaşık olarak aşağıdaki gibi görünür:

```
[vpn]
# Are VPNs enabled?
enabled = yes

# Comma-separated list of the available VPNs.
vpns = vpn0
```

[vpn0]

# Name of this VPN. The name is represented by the filepath to the  
# configuration file, e.g., cuckoo would represent /etc/openvpn/cuckoo.conf  
# Note that you can't assign the names "none" and "internet" as those would  
# conflict with the routing section in cuckoo.conf.

name = vpn0

# The description of this VPN which will be displayed in the web interface.  
# Can be used to for example describe the country where this VPN ends up.  
description = Spain, Europe

# The tun device hardcoded for this VPN. Each VPN *\*must\** be configured to use  
# a hardcoded/persistent tun device by explicitly adding the line "dev tunX"  
# to its configuration (e.g., /etc/openvpn/vpn1.conf) where X in tunX is a  
# unique number between 0 and your lucky number of choice.

interface = tun0

# Routing table name/id for this VPN. If table name is used it *\*must\** be  
# added to /etc/iproute2/route\_tables as "<id> <name>" line (e.g., "201 tun0").  
# ID and name must be unique across the system (refer /etc/iproute2/route\_tables  
# for existing names and IDs).

rt\_table = tun0

Her VPN ağ arayüzünü, iproute2 ile açıklandığı gibi kaydetmek gerekmektedir.

# Per-Analysis Network Routing Kullanımı

Kullanılabilir ağ yönlendirme seçenekleri hakkında bilgi sahibi olduktan sonra, bunu pratikte kullanma zamanı gelmiştir. Cuckoo'nun doğru bir şekilde yapılandırıldığını varsayarsak, Cuckoo Rooter'ı başlatmak ve analiziniz için bir ağ yönlendirme seçeneği seçmek gerçekten de çok basittir.

Cuckoo Rooter'ı başlatma hakkında belgeler, Cuckoo Rooter Kullanım belgesinde bulunabilir.

Hem genel yönlendirme hem de analiz başına yönlendirme için ip yönlendirmenin etkinleştirilmesi gerekmektedir:

```
$ echo 1 | sudo tee -a /proc/sys/net/ipv4/ip_forward $ sudo sysctl -w net.ipv4.ip_forward=1
```

# iproute2 Konfigürasyonu

Linux kernel TCP/IP kaynak yönlendirme nedeniyle kullandığımız her ağ arayüzünü `iproute2` ile kaydetmek gerekmektedir. Bu basit ancak gerekli bir adımdır.

Örneğin, Internet Routing'i yapılandıracağız ve bunun için `eth0` ağ arayüzünü kullanacağız - burada bir saniyelik Ubuntu 14.04 ve daha eski terimlerine geri dönüyoruz (Ubuntu 16.04, muhtemelen BSD tabanlı sistemlerde daima gördüğünüz donanım üreticisine dayalı ağ arayüzü adları kullanır).

`eth0` ile `iproute2`'yi yapılandırmak için `/etc/iproute2/rt_tables` dosyasını açacağız ve bu dosya yaklaşık olarak şu şekilde görünecek:

```
#
# reserved values
#
255    local
254    main
253    default
0      unspec
#
# local
#
```

Şimdi, henüz bu dosyada bulunmayan bir rasgele sayı oluşturun ve bu sayıyı dosyanın sonuna yeni bir satır oluşturmak için kullanın. Örneğin, `eth0`'ı `iproute2` ile kaydetmek aşağıdaki gibi görünebilir:

```
#
# reserved values
#
255    local
254    main
253    default
0      unspec
#
# local
#
```

400 eth0

İşlem bu kadar. Ağ yönlendirmesi için kullanmayı düşündüğünüz her ağ arayüzü için bunu yapmanız gerekecek.

# Guest'i Hazırlamak

Bu noktada Cuckoo ana bilgisayar bileşenini yapılandırmış olmalısınız ve kötü amaçlı yazılım yürütme için kullanacağınız sanal makinelerin sayısını ve adlarını tasarlamış ve tanımlamış olmalısınız. Şimdi bu makineleri oluşturmanın ve uygun şekilde yapılandırmanın zamanı geldi.

# Sanal Makineyi Oluşturmak

Sanallaştırma yazılımınızı doğru bir şekilde kurduktan sonra, ihtiyacınız olan tüm sanal makineleri oluşturabilirsiniz.

Sanallaştırma yazılımınızı kullanmak ve yapılandırmak, bu kılavuzun kapsamı dışındadır.

Sanallaştırılmış ortamınızı nasıl tasarlayıp oluşturacağınıza dair bazı ipuçları ve düşünceleri [Sandboxing](#) bölümünde bulabilirsiniz.

64-bit Windows 7 veya Windows XP sanal makineleri önerilir. Windows 7 için Kullanıcı Erişim Kontrolü'nü devre dışı bırakmanız gerekecektir. 2.0-rc2 sürümünde değiştirildi: Eskiden Windows XP bir konuk VM olarak önerilirdi, ancak günümüzde 64-bit Windows 7 makinesi çok daha iyi sonuçlar vermektedir.

KVM Kullanıcıları - Kesinlikle snapshot'ları destekleyen bir hard disk görüntü formatı seçtiğinizden emin olun. Daha fazla bilgi için Sanal Makineyi Kaydetme bölümüne bakın.

Sanal makine oluşturulurken, Cuckoo'nun belirli bir yapılandırmaya ihtiyacı yoktur. İhtiyaçlarınıza en uygun seçenekleri seçebilirsiniz.

Cuckoo'yu sanallaştırılmış Windows sisteminizde düzgün çalıştırmak için bazı gerekli yazılımları ve kütüphaneleri kurmanız gerekecektir.



# Python Kurulumu

Cuckoo konuk bileşeninin düzgün çalışabilmesi için Python kesin bir gerekliliktir.

Resmi web sitesinden uygun Windows yükleyicisini indirebilirsiniz. Bu durumda da Python 2.7 tercih edilir.

Bazı Python kütüphaneleri ise Cuckoo konuk bileşenine bazı ek özellikler sağlamak için isteğe bağlıdır. Bunlar şunları içerir:

- Python Pillow: Analiz sırasında Windows masaüstünün ekran görüntülerini almak için kullanılır.

Bunlar, Cuckoo'nun düzgün çalışması için kesin olarak gerekliliği olmayan ancak tüm mevcut özelliklere erişmek istiyorsanız bunları kurmanızı önerdiğimiz kütüphanelerdir. Python sürümünüze göre doğru paketleri indirip kurduğunuzdan emin olun.

# Ek Yazılımlar

Bu noktada, Cuckoo'nun düzgün çalışması için gerekli olan her şeyi kurmuş olmalısınız.

Hangi tür dosyaları analiz etmek istediğinize ve hangi türde bir kumlu Windows ortamında kötü amaçlı yazılım örneklerini çalıştırmak istediğinize bağlı olarak tarayıcılar, PDF okuyucular, ofis paketleri vb. gibi ek yazılımları kurabilirsiniz. Ek yazılımların 'otomatik güncelleme' veya 'güncellemeleri kontrol et' özelliğini devre dışı bırakmayı unutmayın.

Bu tamamen size ve ihtiyaçlarınıza bağlıdır. [Sandboxing](#) bölümünü okuyarak bazı ipuçları alabilirsiniz.

Guest'i Hazırlamak

# Network Konfigürasyonu

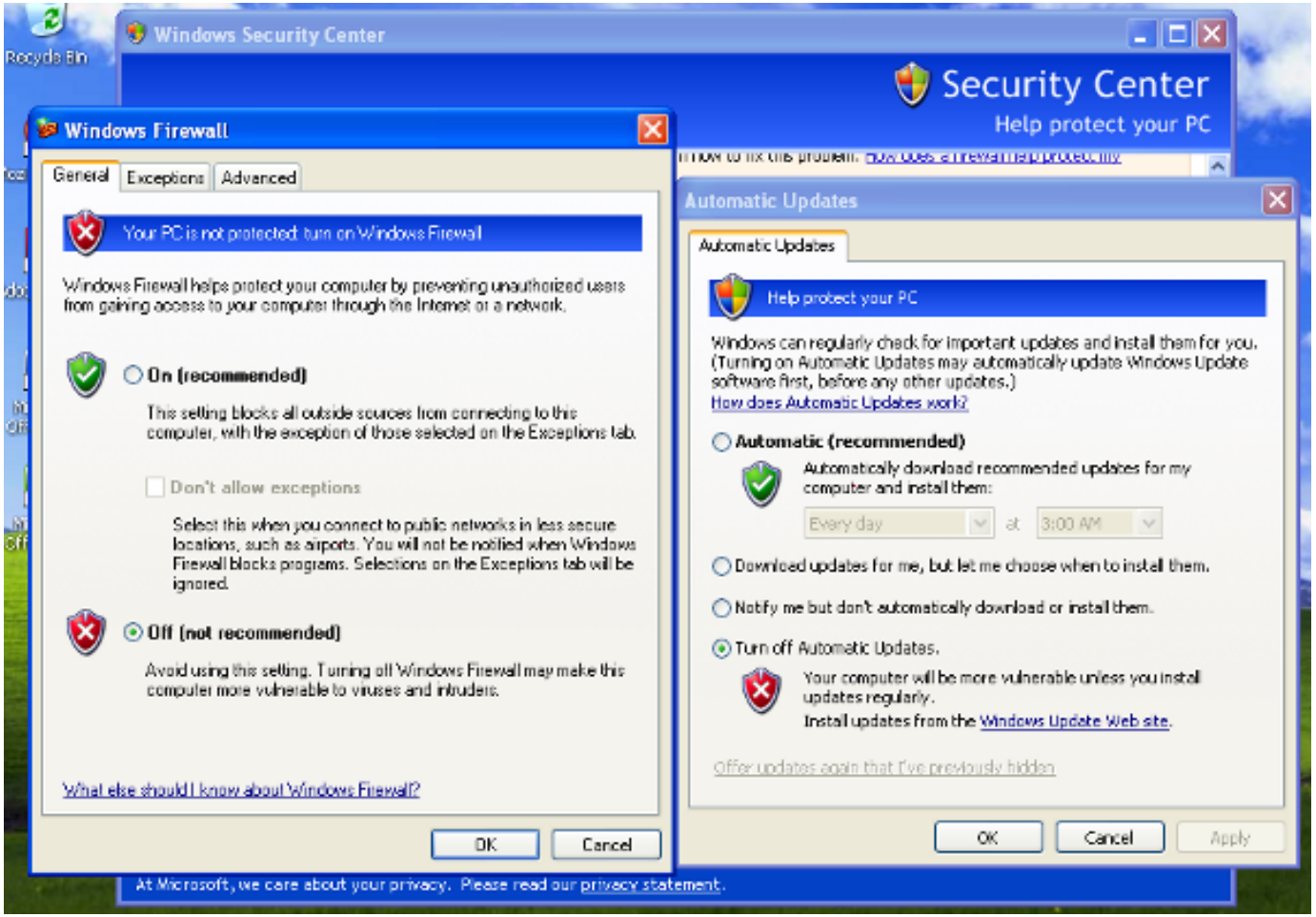
Şimdi sanal makineniz için ağ kurma zamanı geldi.

## Windows Ayarları

Sanal makinenin temel ağ yapılandırmasını yapılandırmadan önce, Windows içinde bazı ayarları düzenlemek isteyebilirsiniz.

Yapılması gereken en önemli şeylerden biri, Windows Güvenlik Duvarı'nı ve Otomatik Güncellemeler'i devre dışı bırakmaktır. Bunun nedeni, bunların normal koşullar altında kötü amaçlı yazılımın davranışını etkileyebileceği ve Cuckoo tarafından gerçekleştirilen ağ analizini, bağlantıları bırakarak veya gereksiz istekleri içerecek şekilde kirletebileceğidir.

Bu ayarları Windows'un Denetim Masası'ndan aşağıdaki resimde gösterildiği gibi yapabilirsiniz:



## Virtual Networking

Şimdi sanal makinenizin İnternet'e veya yerel ağınıza erişebilmesi için nasıl yapılandırılacağına karar vermeniz gerekiyor.

Önceki sürümlerde Cuckoo, Ana Bilgisayar ve Konuklar arasında veri alışverişi için paylaşılan klasörleri kullanırken, 0.4 sürümünden itibaren basit bir XMLRPC protokolü kullanarak ağ üzerinden çalışan özel bir ajansı benimsemektedir.

Bu düzgün çalışabilmesi için makinenizin ağını öyle yapılandırmanız gerekecek ki Host ve Guest iletişim kurabilsin. Sanal ağın doğru bir şekilde kurulup kurulmadığını kontrol etmek için bir konuğu ping ile test etmek iyi bir uygulamadır. Konuğunuz için yalnızca statik IP adresleri kullanın, çünkü Cuckoo DHCP'yi desteklemez ve kullanmak kurulumunuzu bozacaktır.

Bu aşama, kendi gereksinimlerinize ve sanallaştırma yazılımınızın özelliklerine çok bağlıdır.

Sanal ağ, Cuckoo için hayati bir bileşendir, ana bilgisayar ile konuk arasında bağlantı almak için gerçekten emin olmalısınız. Kullanıcılar tarafından bildirilen sorunların çoğu, sanal ağlarının yanlış yapılandırılmasıyla ilgilidir. Eğer bundan emin değilseniz, sanallaştırma

yazılımınızın dokümanını kontrol edin ve bağlantıyı ping ve telnet ile test edin.

Tavsiye edilen kurulum, doğru yönlendirme ile bir Host-Only ağ düzeni kullanmaktır. Bu tür ağ yönlendirmeleri hakkında daha fazla bilgi, ana makine kurulumunun bir parçası olan [Per-Analysis Network Routing](#) bölümünde bulunabilir.

# Agent Kurulumu

Sürüm 0.4'ten itibaren Cuckoo, Gast içinde çalışan ve iletişimi ve veri alışverişini Host ile yöneten özel bir ajan kullanır. Bu ajanın çeşitli platformlarda çalışabilmesi için tasarlanmış olması, bu nedenle Windows, Android, Linux ve Mac OS X üzerinde kullanabilirsiniz. Cuckoo'nun düzgün çalışabilmesi için bu ajanı kurmanız ve çalıştırmanız gerekecek.

Oldukça basit.

`$CWD/agent/` dizininde `agent.py` dosyasını bulacaksınız. Bu dosyayı Guest işletim sistemine kopyalayın (istediğiniz şekilde, belki geçici bir paylaşılan klasör veya ana bilgisayardaki bir web sunucusundan indirerek, bu sonuncusunu önerilir) ve çalıştırın. Ajan, ana bilgisayarın iletişim kurabileceği küçük bir API sunucusunu başlatacaktır.

Windows üzerinde basitçe komut dosyasını başlatmak, aynı zamanda bir Python penceresi de başlatacaktır; eğer bunu gizlemek istiyorsanız dosyayı `agent.py`'den `agent.pyw` olarak yeniden adlandırabilirsiniz, bu da konsol penceresinin başlamasını engeller.

Eğer Windows'un başlangıcında betiğin başlatılmasını istiyorsanız, dosyayı Başlangıç klasörüne yerleştirin.

# Sanal Makineyi Kaydetmek

Şimdi sanal makinenizi bir anlık görüntü durumuna kaydetmeye hazır olmalısınız.

Bunu yapmadan önce, onu nazikçe yeniden başlattığınızdan ve şu anda çalıştığından, Cuckoo'nun ajanının çalıştığından ve Windows'un tamamen başladığından emin olun.

Şimdi makineyi kaydetmeye devam edebilirsiniz. Bunun nasıl yapılacağı açıkça kullandığınız sanallaştırma yazılımına bağlıdır.

Eğer aşağıdaki adımları düzgün bir şekilde takip ederseniz, sanal makineniz Cuckoo tarafından kullanılmaya hazır olacaktır.

## Virtualbox

VirtualBox'u seçiyorsanız, anlık görüntüyü grafik arayüzden veya komut satırından alabilirsiniz:

```
$ VBoxManage snapshot "<Name of VM>" take "<Name of snapshot>" --pause
```

Anlık görüntü oluşturma işlemi tamamlandıktan sonra makineyi kapatıp geri yükleyebilirsiniz:

```
$ VBoxManage controlvm "<Name of VM>" poweroff  
$ VBoxManage snapshot "<Name of VM>" restorecurrent
```

## KVM

Eğer KVM'i kullanmaya karar verdiyseniz, öncelikle sanal makineleriniz için anlık görüntüler destekleyen bir disk formatı kullandığınızdan emin olmalısınız. Libvirt araçları varsayılan olarak RAW sanal diskler oluşturur, ve çünkü bizim anlık görüntülere ihtiyacımız var, QCOW2 veya LVM kullanmanız gerekecek. Bu kılavuzun kapsamı için QCOW2'yi benimsemekteyiz, ki bu LVM'den daha kolay kurulumdur.

Bu tür bir sanal diski doğru bir şekilde oluşturmanın en kolay yolu, libvirt paketi tarafından sağlanan araçları kullanmaktır. Virsh'i tercih ediyorsanız komut satırı arayüzünü veya güzel bir GUI için virt-manager'ı kullanabilirsiniz. QCOW2 formatında doğrudan oluşturabilirsiniz, ancak RAW bir diskiniz varsa bunu şu şekilde dönüştürebilirsiniz:

```
$ cd /your/disk/image/path  
$ qemu-img convert -O qcow2 your_disk.raw your_disk.qcow2
```

Şimdi VM tanımınızı aşağıdaki gibi düzenlemeniz gerekiyor:

```
$ virsh edit "<Name of VM>"
```

Disk bölümünü bulun, şu şekilde görünüyor:

```
<disk type='file' device='disk'>  
  <driver name='qemu' type='raw'/>  
  <source file='/your/disk/image/path/your_disk.raw'/>  
  <target dev='hda' bus='ide'/>  
  <address type='drive' controller='0' bus='0' unit='0'/>  
</disk>
```

“type”ı qcow2 olarak değiştirin ve “source file”ı qcow2 disk görüntünüze değiştirin, şu şekilde:

```
<disk type='file' device='disk'>  
  <driver name='qemu' type='qcow2'/>  
  <source file='/your/disk/image/path/your_disk.qcow2'/>  
  <target dev='hda' bus='ide'/>  
  <address type='drive' controller='0' bus='0' unit='0'/>  
</disk>
```

Şimdi sanal makinenizi test edin, eğer her şey çalışıyorsa Cuckoo Agent çalıştırırken sanal makineyi anlık olarak hazırlayın. Bu, sanal makineyi anlık alırken çalışır durumda olması gerektiği anlamına gelir. Ardından, onu kapatabilirsiniz. Aşağıdaki komutla nihayet bir anlık görüntü alabilirsiniz:

```
$ virsh snapshot-create "<Name of VM>"
```



Birkaç anlık görüntü almak hatalara neden olabilir:

```
ERROR: No snapshot found for virtual machine VM-Name
```

VM anlık görüntüleri aşağıdaki komutlarla yönetilebilir:

```
$ virsh snapshot-list "VM-Name"  
$ virsh snapshot-delete "VM-Name" 1234567890
```

## Vmware Workstation

Eğer VMware Workstation kullanmaya karar verdiyseniz, anlık görüntüyü grafik kullanıcı arayüzünden veya komut satırından alabilirsiniz:

```
$ vmrun snapshot "/your/disk/image/path/wmware_image_name.vmx"  
your_snapshot_name
```

Burada your\_snapshot\_name, anlık görüntü için seçtiğiniz addır. Ardından makineyi GUI veya komut satırından kapatın:

```
$ vmrun stop "/your/disk/image/path/wmware_image_name.vmx" hard
```

# Sanal Makinenin Klonlanması

Eğer birden fazla sanal makine kullanmayı planlıyorsanız, şimdiye kadar yapılan tüm adımları tekrarlamaya gerek yok: onu klonlayabilirsiniz. Böylece, tüm gereksinimleri zaten yüklenmiş olan orijinal sanal Windows'un bir kopyasına sahip olacaksınız.

Yeni sanal makine aynı zamanda orijinalin tüm ayarlarını içerecektir, ki bu da iyi değildir. Şimdi, bu yeni makine için [Network Konfigürasyonu](#), [Agent Kurulumu](#) ve [Sanal Makineyi Kaydetmek](#) adımlarını tekrarlayarak devam etmeniz gerekiyor.

# Linux Host Kurulumu

Öncelikle, makine platformunuz için ana bilgisayar tarafındaki ağ yapılandırmasını hazırlayın. Örneğin, VirtualBox'ı host-only arabirimleriyle kullanıyorsanız ve vboxnet0 arabiriminiz varsa, ek bağımlılıkları kurmanıza gerek yoktur.

QEMU kullanıyorsanız, ana bilgisayarda ek bağımlılıkları kurmanız gerekebilir:

```
$ sudo apt install uml-utilities bridge-utils
```

Ardından, arabirimini yapılandırmak için sanal makinelerin listesini conf/qemu.conf'dan alın. Örneğin, ubuntu\_x32, ubuntu\_x64, ubuntu\_arm, ubuntu\_mips, ubuntu\_mipsel vb. Her bir VM için, kök olarak başlatmak zorunda kalmamak için ana bilgisayarda bir ağ tap arabirimi önceden yapılandırın, örneğin:

```
$ sudo tuncctl -b -u cuckoo -t tap_ubuntu_x32
$ sudo ip link set tap_ubuntu_x32 master br0
$ sudo ip link set dev tap_ubuntu_x32 up
$ sudo ip link set dev br0 up

$ sudo tuncctl -b -u cuckoo -t tap_ubuntu_x64
$ sudo ip link set tap_ubuntu_x64 master br0
$ sudo ip link set dev tap_ubuntu_x64 up
$ sudo ip link set dev br0 up
```

Cuckoo'yu farklı bir kullanıcı olarak çalıştırıyorsanız, -u'dan sonraki "cuckoo" ifadesini kendi kullanıcı adınızla değiştirin.

## x32/x64 Ubuntu 18.04 Linux Guest'i Hazırlamak

Agent'ın otomatik olarak başladığından emin olun. En kolay yol, crontab'a eklemektir:

```
$ sudo crontab -e
@reboot python /path/to/agent.py
```

Sanal makine içerisinde bağımlılıkları yükleyin:

```
$ sudo apt-get install systemtap gcc patch linux-headers-$(uname -r)
```

Kernel debugging symbols kurun:

```
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
C8CAB6595FDFF622

$ codename=$(lsb_release -cs)
$ sudo tee /etc/apt/sources.list.d/ddebs.list << EOF
deb http://ddebs.ubuntu.com/ ${codename}          main restricted universe
multiverse
#deb http://ddebs.ubuntu.com/ ${codename}-security main restricted
universe multiverse
deb http://ddebs.ubuntu.com/ ${codename}-updates  main restricted universe
multiverse
deb http://ddebs.ubuntu.com/ ${codename}-proposed main restricted universe
multiverse
EOF

$ sudo apt-get update
$ sudo apt-get install linux-image-$(uname -r)-dbgsym
```

(Debian 9 amd64 için) Kernel debugging symbols kurun:

```
$ sudo apt-get install linux-image-$(uname -r)-dbg
```

SystemTap tapset'i düzeltin, böylece Cuckoo analyzer çıktıyı düzgün bir şekilde çözebilir:

```
$ wget
https://raw.githubusercontent.com/cuckoosandbox/cuckoo/master/stuff/systemt
ap/expand_execve_envp.patch
```

```
$ wget
https://raw.githubusercontent.com/cuckoosandbox/cuckoo/master/stuff/systemtap/escape_delimiters.patch
$ sudo patch /usr/share/systemtap/tapset/linux/sysc_execve.stp <
expand_execve_envp.patch
$ sudo patch /usr/share/systemtap/tapset/uconversions.stp <
escape_delimiters.patch
```

Kernel extension'ı derleyin:

```
$ wget
https://raw.githubusercontent.com/cuckoosandbox/cuckoo/master/stuff/systemtap/strace.stp
$ sudo stap -p4 -r $(uname -r) strace.stp -m stap_ -v
```

Derleme işlemi tamamlandığında aynı klasörde stap\_.ko dosyasını görmelisiniz. Şimdi STAP kernel extension'ı aşağıdaki gibi test edebilirsiniz:

```
$ sudo staprun -v ./stap_.ko
```

Çıktı aşağıdaki gibi olmalıdır:

```
staprun:insert_module:x Module stap_ inserted from file path_to_stap_.ko
```

stap\_.ko dosyasını /root/.cuckoo dizinine yerleştirmelisiniz:

```
$ sudo mkdir /root/.cuckoo
$ sudo mv stap_.ko /root/.cuckoo/
```

Eğer varsa, sanal makinedeki güvenlik duvarını devre dışı bırakın:

```
$ sudo ufw disable
```

Sanal makine içinde NTP'yi devre dışı bırakın:

```
$ sudo timedatectl set-ntp off
```

İsteğe bağlı - önceden yüklenmiş yazılım ve konfigürasyonları kaldırın:

```
$ sudo apt-get purge update-notifier update-manager update-manager-core  
ubuntu-release-upgrader-core  
$ sudo apt-get purge whoopsie ntpdate cups-daemon avahi-autoipd avahi-  
daemon avahi-utils  
$ sudo apt-get purge account-plugin-salut libnss-mdns telepathy-salut
```

Linux guest sistemini statik IP adresleriyle yapılandırmak önerilir. Konfigürasyondaki makine girişinin doğru IP adresine sahip olduğundan ve platform değişkeninin linux olarak ayarlandığından emin olun. VM yapılandırıldıktan sonra bir anlık görüntü oluşturun. Şimdi analiz için hazırız.