Processing Modülleri

Cuckoo'nun işleme modülleri, sandbox'ın ürettiği raw sonuçları özelleştirmenize ve daha sonra imza ve raporlama modülleri tarafından kullanılacak genel bir konteynere bilgi eklemenize olanak tanıyan Python betikleridir.

Birçok modül oluşturabilirsiniz, ancak bu modüllerin önceden belirlenmiş bir yapıyı takip etmeleri gerekmektedir ki bunu bu bölümde sunacağız.

Global Konteyner

Bir analiz tamamlandığında, Cuckoo, `cuckoo/processing/` dizininde bulunan tüm işleme modüllerini çağıracaktır; bunların tümü cuckoo.processing modülü altındadır. Oluşturmayı seçtiğiniz herhangi bir ek modülü, bu dizin içine yerleştirmeniz gerekmektedir.

Her modülün ayrıca `$CWD/conf/processing.conf` dosyasında özel bir bölümü olmalıdır: örneğin, cuckoo/processing/foobar.py adında bir modül oluşturursanız, `$CWD/conf/processing.conf` dosyasına aşağıdaki bölümü eklemeniz gerekecektir:

```
[foobar]
enabled = yes
```

Ardından her modül başlatılacak ve çalıştırılacak, dönen veri global bir konteyner olarak adlandıracağımız bir veri yapısına eklenir.

Bu konteyner, sadece tüm modüller tarafından üretilen sonuçları kimlik anahtarlarına göre sınıflandırılmış bir Python sözlüğüdür.

Cuckoo zaten bir dizi varsayılan modül sağlar ve bu modüller, standart bir global konteyner oluşturacaktır. Bu varsayılan modüllerin değiştirilmemesi önemlidir, aksi takdirde oluşan global konteyner yapısı değişir ve raporlama modülleri bu yapısını tanıyamaz ve nihai raporları oluşturmak için kullanılan bilgileri çıkaramaz.

Mevcut varsayılan işleme modülleri şunlardır:

- AnalysisInfo (`cuckoo/processing/analysisinfo.py`) - mevcut analiz hakkında bazı temel bilgiler oluşturur, zaman damgaları, Cuckoo sürümü vb.
- ApkInfo (`cuckoo/processing/apkinfo.py`) - mevcut APK analizi (Android analizi) hakkında bazı temel bilgiler oluşturur.
- Baseline (`cuckoo/processing/baseline.py`) - toplanan bilgilerden temel çizgileri oluşturur.

- BehaviorAnalysis (`cuckoo/processing/behavior.py`) - ham davranış günlüklerini ayrıştırır ve bazı ilk dönüşümler ve yorumlamalar yapar, bunlar arasında tam süreç izleme, davranış özeti ve süreç ağacı bulunur.
- Buffer (`cuckoo/processing/buffer.py`) - bırakılan tampon analizi.
- Debug (`cuckoo/processing/debug.py`) - analizci tarafından oluşturulan hataları ve analysis.log dosyasını içerir.
- Droidmon (`cuckoo/processing/droidmon.py`) - Droidmon günlüklerinden Dinamik API çağrıları Bilgisi çıkarır.
- Dropped (`cuckoo/processing/dropped.py`) - zararlı tarafından bırakılan ve Cuckoo tarafından dökülen dosyalar hakkında bilgi içerir.
- DumpTls (`cuckoo/processing/dumptls.py`) - monitörden çıkarılan TLS anahtar sırlarını ve PCAP'dan çıkarılan anahtar bilgilerini çapraz referanslamak için bir anahtar sırları dosyasını dökmek üzere.
- GooglePlay (`cuckoo/processing/googleplay.py`) - analiz oturumu hakkında Google Play bilgisi.
- Irma (`cuckoo/processing/irma.py`) - IRMA bağlayıcısı.
- Memory (`cuckoo/processing/memory.py`) - tam bir bellek dökümünde Volatility'yi yürütür.
- Misp (`cuckoo/processing/misp.py`) - MISP bağlayıcısı.
- NetworkAnalysis (`cuckoo/processing/network.py`) - PCAP dosyasını ayrıştırır ve DNS trafiği, alanlar, IP'ler, HTTP istekleri, IRC ve SMTP trafiği gibi bazı ağ bilgilerini çıkarır.
- ProcMemory (`cuckoo/processing/procmemory.py`) - süreç bellek dökümü analizi yapar. Not: modül, data/yara/memory/index_memory.yar'daki kullanıcı tanımlı Yara kurallarını işlemek için yeteneklidir. Yara kurallarınızı eklemek için bu dosyayı düzenleyin.
- ProcMon (`cuckoo/processing/procmon.py`) - procmon.exe çıkışından olayları çıkarır.
- Screenshots (`cuckoo/processing/screenshots.py`) - ekran görüntüsü ve OCR analizi.
- Snort (`cuckoo/processing/snort.py`) - Snort işleme modülü.
- StaticAnalysis (`cuckoo/processing/static.py`) - PE32 dosyalarının bazı statik analizlerini gerçekleştirir.
- Strings (`cuckoo/processing/strings.py`) - analiz edilen ikili dosyadan dizgileri çıkarır.
- Suricata (`cuckoo/processing/suricata.py`) - Suricata işleme modülü.
- TargetInfo (`cuckoo/processing/targetinfo.py`) - analiz edilen dosya hakkında bilgi içerir, örneğin karma değerleri.
- VirusTotal (`cuckoo/processing/virustotal.py`) - analiz edilen dosyanın antivirüs imzaları için VirusTotal.com'da arama yapar. Not: Dosya VirusTotal.com'a yüklenmez, dosya daha önce web sitesine yüklenmediyse sonuçlar alınamaz.

Başlarken

Onları Cuckoo'ya kullanılabilir hale getirmek için tüm işleme modülleri `cuckoo/processing/` dizinine yerleştirilmelidir.

Temel bir işleme modülü şöyle görünebilir:

```
from cuckoo.common.abstracts import Processing
```

```
class MyModule(Processing):

    def run(self):
        self.key = "key"
        data = do_something()
        return data
```

Her işleme modülü şunları içermelidir:

- `Processing` sınıfından türetilmiş bir sınıf.
- Bir `run()` fonksiyonu.
- Dönen veriler için alt konteyner olarak kullanılacak adı tanımlayan `self.key` özniteliği.
- Global konteynere eklenen bir veri seti (liste, sözlük, dize vb.).

Ayrıca bir sıra değeri belirleyebilirsiniz, bu size kullanılabilir işleme modüllerini sıralı bir dizide çalıştırma olanağı sağlar. Varsayılan olarak, tüm modüllerin sıra değeri 1 olarak ayarlanmıştır ve alfabetik sırayla çalıştırılır.

Bu değeri değiştirmek istiyorsanız, modülünüz şöyle görünecektir:

```
from cuckoo.common.abstracts import Processing

class MyModule(Processing):
    order = 2

    def run(self):
        self.key = "key"
        data = do_something()
        return data
```

Ayrıca bir işleme modülünü manuel olarak devre dışı bırakabilirsiniz, bunun için `enabled` özniteliğini `False` olarak ayarlamanız gerekmektedir:

```
from cuckoo.common.abstracts import Processing

class MyModule(Processing):
    enabled = False

    def run(self):
        self.key = "key"
        data = do_something()
```

```
        return data
```

İşleme modülleri, verilen analiz için ham sonuçlara erişmek için kullanılabilecek bazı özniteliklerle sağlanır:

- `self.analysis_path`: sonuçları içeren klasörün yolunu belirtir (örneğin, `$CWD/storage/analysis/1`)
- `self.log_path`: `analysis.log` dosyasının yolunu belirtir.
- `self.file_path`: analiz edilen dosyanın yolunu belirtir.
- `self.dropped_path`: bırakılan dosyaları içeren klasörün yolunu belirtir.
- `self.logs_path`: raw davranış günlüklerini içeren klasörün yolunu belirtir.
- `self.shots_path`: ekran görüntülerini içeren klasörün yolunu belirtir.
- `self.pcap_path`: ağ pcap dökümünün yolunu belirtir.
- `self.memory_path`: eğer oluşturulmuşsa tam bellek dökümünün yolunu belirtir.
- `self.pmemory_path`: eğer oluşturulmuşsa süreç belleği dökümlerinin yolunu belirtir.

Bu özniteliklerle Cuckoo tarafından depolanan tüm ham sonuçlara kolayca erişebilmeli ve bunlar üzerinde analitik işlemlerinizi gerçekleştirebilmelisiniz.

Son bir not olarak, modülün bir sorunla karşılaştığında Cuckoo'ya bildirmek istediğiniz bir durumda CuckooProcessingError istisnasını kullanmak iyi bir uygulamadır. Bu, şu şekilde sınıfı içe aktararak yapılabilir:

```
from cuckoo.common.exceptions import CuckooProcessingError
from cuckoo.common.abstracts import Processing

class MyModule(Processing):

    def run(self):
        self.key = "key"

        try:
            data = do_something()
        except SomethingFailed:
            raise CuckooProcessingError("Failed")

        return data
```

Revision #1
Created 19 January 2024 11:53:47 by Ertan Sözer
Updated 19 January 2024 11:56:01 by Ertan Sözer