Part 1 Cuckoo宿主机运行过程解构

@click.group(invoke\_without\_command=True)  
@click.option("-d", "--debug", is\_flag=True, help="Enable verbose logging")  
@click.option("-q", "--quiet", is\_flag=True, help="Only log warnings and critical messages")  
@click.option("--nolog", is\_flag=True, help="Don't log to file.")  
@click.option("-m", "--maxcount", default=0, help="Maximum number of analyses to process")  
@click.option("--user", help="Drop privileges to this user")  
@click.option("--cwd", help="Cuckoo Working Directory")  
@click.pass\_context  
def main(ctx, debug, quiet, nolog, maxcount, user, cwd):  
 *"""Invokes the Cuckoo daemon or one of its subcommands.*

通过click库解析拼接命令，解析命令行参数。每个选项后面的东西会被传到相应选项里。

decide\_cwd(cwd)

1. 确定cwd （223上是/home/cuckoo）如果输入的路径不正确，默认为 ~/.cuckoo。这个目录是存放signatures和各种配置的地方

if quiet:  
 level = logging.WARN  
elif debug:  
 level = logging.DEBUG  
else:  
 level = logging.INFO  
  
ctx.level = level

设定日志写多少

if ctx.invoked\_subcommand:  
 return  
  
try:  
 cuckoo\_init(level, ctx)  
 cuckoo\_main(maxcount)

如果采用启动了子命令（例如: clean、api），就返回

两个主线程，先看cuckoo\_init

def cuckoo\_init(level, ctx, cfg=None):  
 *"""Initialize Cuckoo configuration.  
 @param quiet: enable quiet mode.  
 """* logo()

打印标识

if not os.path.isdir(cwd()) or not os.listdir(cwd()):  
 cuckoo\_create(ctx.user, cfg)  
 sys.exit(0)  
  
# Determine if this is a proper CWD.  
if not os.path.exists(cwd(".cwd")):  
 sys.exit(  
 "No proper Cuckoo Working Directory was identified, did you pass "  
 "along the correct directory? For new installations please use a "  
 "non-existant directory to build up the CWD! You can craft a CWD "  
 "manually, but keep in mind that the CWD layout may change along "  
 "with Cuckoo releases (and don't forget to fill out '$CWD/.cwd')!"  
 )

创建cwd

init\_console\_logging(level)  
  
check\_configs()

开始打印命令行日志

检查配置文件 check\_config

check\_config:

def check\_configs():  
 *"""Checks if config files exist.  
 @raise CuckooStartupError: if config files do not exist.  
 """* configs = (  
 "auxiliary", "cuckoo", "memory", "processing", "reporting", "routing",  
 )  
  
 for filename in configs:  
 if not os.path.exists(cwd("conf", "%s.conf" % filename)):  
 raise CuckooStartupError(  
 "Config file does not exist at path: %s" %  
 cwd("conf", "%s.conf" % filename)  
 )  
  
 check\_specific\_config(filename)

到configs文件夹下获得各部分的配置文件，每个配置项采用以下格式显示

文件夹：配置项：配置

machinery = config("cuckoo:cuckoo:machinery")  
 if machinery not in Config.configuration:  
 raise CuckooStartupError(  
 "An unknown machinery has been chosen (machinery=%s)!" % machinery  
 )  
  
 check\_specific\_config(machinery)

对应cuckoo.conf的machinery即所用的虚拟机（Virtualbox vmware）

# If Cuckoo Feedback is enabled, ensure its configuration is valid.  
 feedback\_enabled = (  
 config("cuckoo:feedback:enabled") or  
 config("reporting:feedback:enabled")  
 )  
 if feedback\_enabled:  
 try:  
 CuckooFeedbackObject(  
 name=config("cuckoo:feedback:name"),  
 email=config("cuckoo:feedback:email"),  
 company=config("cuckoo:feedback:company"),  
 message="startup"  
 ).validate()  
 except CuckooFeedbackError as e:  
 raise CuckooStartupError(  
 "You have filled out the Cuckoo Feedback configuration, but "  
 "there's an error in it: %s" % e  
 )  
 return True

这部分Feedback的，Feedback会把报告等相关资源打包以json格式发送到配置文件指定的端口去。

返回到主线程cuckoo\_init继续

Database().connect()

连接沙箱指定的数据库，根据cuckoo.conf的Database设置来定（MySQL, Postgresql, sqlite3）

load\_signatures()

def load\_signatures():  
 *"""Loads additional Signatures from the Cuckoo Working Directory.  
  
 This method is quite hacky in the sense that it magically imports  
 Signatures from an arbitrary directory - one that doesn't belong to the  
 Cuckoo package directly.  
  
 Furthermore this method provides backwards compatibility with older  
 Signatures which rely on the "lib.cuckoo.common.abstracts" import, one  
 that can now be accessed as "cuckoo.common.abstracts".  
 """* # Forward everything from lib.cuckoo to "our" cuckoo module.  
 sys.modules["lib"] = types.ModuleType("lib")  
 sys.modules["lib.cuckoo"] = sys.modules["cuckoo"]  
  
 # Import this here in order to avoid recursive import statements.  
 from common.abstracts import Signature  
  
 # Define Signature in such a way that it is equal to "our" Signature.  
 sys.modules["lib.cuckoo.common.abstracts"] = types.ModuleType(  
 "lib.cuckoo.common.abstracts"  
 )  
 sys.modules["lib.cuckoo.common.abstracts"].Signature = Signature  
  
 # Don't clobber the Cuckoo Working Directory with .pyc files.  
 dont\_write\_bytecode = sys.dont\_write\_bytecode  
 sys.dont\_write\_bytecode = True  
  
 # Trigger an import on $CWD/signatures. This will automatically import  
 # recursively down the various directories through the use of  
 # enumerate\_plugins(), which the Cuckoo Community adheres to. For this to  
 # work we temporarily insert the CWD in Python's path.  
 sys.path.insert(0, cwd())  
 mod = importlib.import\_module("signatures")  
 sys.path.pop(0)  
  
 # Restore bytecode option.  
 sys.dont\_write\_bytecode = dont\_write\_bytecode  
  
 # Index all of the available Signatures that have been located.  
 for key, value in sorted(mod.\_\_dict\_\_.items()):  
 if not key.startswith("\_") and hasattr(value, "plugins"):  
 cuckoo.signatures.extend(value.plugins)

加载可用的signatures（存放在/home/cuckoo/signatures下）

init\_modules()

def init\_modules():  
 *"""Initializes plugins."""* log.debug("Imported modules...")  
  
 categories = (  
 "auxiliary", "machinery", "processing", "signatures", "reporting",  
 )  
  
 # Call the init\_once() static method of each plugin/module. If an exception  
 # is thrown in that initialization call, then a hard error is appropriate.  
 for category in categories:  
 for module in cuckoo.plugins[category]:  
 module.init\_once()  
  
 for category in categories:  
 log.debug("Imported \"%s\" modules:", category)  
  
 entries = cuckoo.plugins[category]  
 for entry in entries:  
 if entry == entries[-1]:  
 log.debug("\t `-- %s", entry.\_\_name\_\_)  
 else:  
 log.debug("\t |-- %s", entry.\_\_name\_\_)  
  
 # Initialize the RunSignatures module with all available Signatures.  
 RunSignatures.init\_once()

对  
plugins = {  
 "auxiliary": auxiliary.plugins,  
 "machinery": machinery.plugins.values(),  
 "processing": processing.plugins,  
 "reporting": reporting.plugins,  
 "signatures": signatures,  
}

这些对象执行其init\_once方法

加载的模块有以下这些

2018-01-22 12:15:03,692 [cuckoo.core.startup] DEBUG: Imported "auxiliary" modules:

2018-01-22 12:15:03,692 [cuckoo.core.startup] DEBUG: |-- MITM

2018-01-22 12:15:03,692 [cuckoo.core.startup] DEBUG: |-- Reboot

2018-01-22 12:15:03,692 [cuckoo.core.startup] DEBUG: |-- Services

2018-01-22 12:15:03,693 [cuckoo.core.startup] DEBUG: `-- Sniffer

2018-01-22 12:15:03,693 [cuckoo.core.startup] DEBUG: Imported "machinery" modules:

2018-01-22 12:15:03,693 [cuckoo.core.startup] DEBUG: |-- vSphere

2018-01-22 12:15:03,693 [cuckoo.core.startup] DEBUG: |-- KVM

2018-01-22 12:15:03,693 [cuckoo.core.startup] DEBUG: |-- ESX

2018-01-22 12:15:03,694 [cuckoo.core.startup] DEBUG: |-- XenServer

2018-01-22 12:15:03,694 [cuckoo.core.startup] DEBUG: |-- VirtualBox

2018-01-22 12:15:03,694 [cuckoo.core.startup] DEBUG: |-- Avd

2018-01-22 12:15:03,694 [cuckoo.core.startup] DEBUG: |-- QEMU

2018-01-22 12:15:03,695 [cuckoo.core.startup] DEBUG: |-- VMware

2018-01-22 12:15:03,695 [cuckoo.core.startup] DEBUG: `-- Physical

2018-01-22 12:15:03,695 [cuckoo.core.startup] DEBUG: Imported "processing" modules:

2018-01-22 12:15:03,695 [cuckoo.core.startup] DEBUG: |-- AnalysisInfo

2018-01-22 12:15:03,695 [cuckoo.core.startup] DEBUG: |-- ApkInfo

2018-01-22 12:15:03,696 [cuckoo.core.startup] DEBUG: |-- Baseline

2018-01-22 12:15:03,696 [cuckoo.core.startup] DEBUG: |-- BehaviorAnalysis

2018-01-22 12:15:03,696 [cuckoo.core.startup] DEBUG: |-- Debug

2018-01-22 12:15:03,696 [cuckoo.core.startup] DEBUG: |-- Droidmon

2018-01-22 12:15:03,696 [cuckoo.core.startup] DEBUG: |-- Dropped

2018-01-22 12:15:03,697 [cuckoo.core.startup] DEBUG: |-- DroppedBuffer

2018-01-22 12:15:03,697 [cuckoo.core.startup] DEBUG: |-- Extracted

2018-01-22 12:15:03,697 [cuckoo.core.startup] DEBUG: |-- GooglePlay

2018-01-22 12:15:03,697 [cuckoo.core.startup] DEBUG: |-- Irma

2018-01-22 12:15:03,697 [cuckoo.core.startup] DEBUG: |-- Memory

2018-01-22 12:15:03,698 [cuckoo.core.startup] DEBUG: |-- MetaInfo

2018-01-22 12:15:03,698 [cuckoo.core.startup] DEBUG: |-- MISP

2018-01-22 12:15:03,698 [cuckoo.core.startup] DEBUG: |-- NetworkAnalysis

2018-01-22 12:15:03,698 [cuckoo.core.startup] DEBUG: |-- ProcessMemory

2018-01-22 12:15:03,698 [cuckoo.core.startup] DEBUG: |-- Procmon

2018-01-22 12:15:03,699 [cuckoo.core.startup] DEBUG: |-- Screenshots

2018-01-22 12:15:03,699 [cuckoo.core.startup] DEBUG: |-- Snort

2018-01-22 12:15:03,699 [cuckoo.core.startup] DEBUG: |-- Static

2018-01-22 12:15:03,699 [cuckoo.core.startup] DEBUG: |-- Strings

2018-01-22 12:15:03,699 [cuckoo.core.startup] DEBUG: |-- Suricata

2018-01-22 12:15:03,700 [cuckoo.core.startup] DEBUG: |-- TargetInfo

2018-01-22 12:15:03,700 [cuckoo.core.startup] DEBUG: |-- TLSMasterSecrets

2018-01-22 12:15:03,700 [cuckoo.core.startup] DEBUG: `-- VirusTotal

2018-01-22 12:15:03,700 [cuckoo.core.startup] DEBUG: Imported "signatures" modules:

2018-01-22 12:15:03,700 [cuckoo.core.startup] DEBUG: |-- AntiAnalysisJavascript

2018-01-22 12:15:03,701 [cuckoo.core.startup] DEBUG: |-- DumpedBuffer

2018-01-22 12:15:03,701 [cuckoo.core.startup] DEBUG: |-- DumpedBuffer2

2018-01-22 12:15:03,701 [cuckoo.core.startup] DEBUG: |-- EncryptionKeys

2018-01-22 12:15:03,701 [cuckoo.core.startup] DEBUG: |-- EvalJS

2018-01-22 12:15:03,701 [cuckoo.core.startup] DEBUG: |-- Exploit\_zteF460F660

2018-01-22 12:15:03,702 [cuckoo.core.startup] DEBUG: |-- HtmlFlash

2018-01-22 12:15:03,702 [cuckoo.core.startup] DEBUG: |-- JsIframe

2018-01-22 12:15:03,702 [cuckoo.core.startup] DEBUG: |-- PDFAttachments

2018-01-22 12:15:03,702 [cuckoo.core.startup] DEBUG: |-- PDFJavaScript

2018-01-22 12:15:03,702 [cuckoo.core.startup] DEBUG: |-- PDFOpenAction

2018-01-22 12:15:03,703 [cuckoo.core.startup] DEBUG: |-- PDFOpenActionJS

2018-01-22 12:15:03,703 [cuckoo.core.startup] DEBUG: |-- SuspiciousJavascript

2018-01-22 12:15:03,703 [cuckoo.core.startup] DEBUG: |-- DeadHost

2018-01-22 12:15:03,703 [cuckoo.core.startup] DEBUG: |-- NetworkBIND

2018-01-22 12:15:03,703 [cuckoo.core.startup] DEBUG: |-- NetworkDynDNS

2018-01-22 12:15:03,704 [cuckoo.core.startup] DEBUG: |-- NetworkHTTP

2018-01-22 12:15:03,704 [cuckoo.core.startup] DEBUG: |-- NetworkICMP

2018-01-22 12:15:03,704 [cuckoo.core.startup] DEBUG: |-- NetworkIRC

2018-01-22 12:15:03,704 [cuckoo.core.startup] DEBUG: |-- NetworkSMTP

2018-01-22 12:15:03,704 [cuckoo.core.startup] DEBUG: |-- SnortAlert

2018-01-22 12:15:03,705 [cuckoo.core.startup] DEBUG: |-- SuricataAlert

2018-01-22 12:15:03,705 [cuckoo.core.startup] DEBUG: |-- Suspicious\_TLD

2018-01-22 12:15:03,705 [cuckoo.core.startup] DEBUG: |-- TorGateway

2018-01-22 12:15:03,705 [cuckoo.core.startup] DEBUG: |-- WscriptDownloader

2018-01-22 12:15:03,705 [cuckoo.core.startup] DEBUG: |-- ADS

2018-01-22 12:15:03,706 [cuckoo.core.startup] DEBUG: |-- Adzok

2018-01-22 12:15:03,706 [cuckoo.core.startup] DEBUG: |-- AlinaFile

2018-01-22 12:15:03,706 [cuckoo.core.startup] DEBUG: |-- AlineURL

2018-01-22 12:15:03,706 [cuckoo.core.startup] DEBUG: |-- AllocatesRWX

2018-01-22 12:15:03,706 [cuckoo.core.startup] DEBUG: |-- AmsiBypass

2018-01-22 12:15:03,707 [cuckoo.core.startup] DEBUG: |-- Andromeda

2018-01-22 12:15:03,707 [cuckoo.core.startup] DEBUG: |-- AntiAnalysisDetectFile

2018-01-22 12:15:03,707 [cuckoo.core.startup] DEBUG: |-- AntiAVDetectFile

2018-01-22 12:15:03,707 [cuckoo.core.startup] DEBUG: |-- AntiAVDetectReg

2018-01-22 12:15:03,708 [cuckoo.core.startup] DEBUG: |-- AntiAVSRP

2018-01-22 12:15:03,708 [cuckoo.core.startup] DEBUG: |-- AntiDBGDevices

2018-01-22 12:15:03,708 [cuckoo.core.startup] DEBUG: |-- AntiDBGWindows

2018-01-22 12:15:03,708 [cuckoo.core.startup] DEBUG: |-- AntiSandboxFile

2018-01-22 12:15:03,708 [cuckoo.core.startup] DEBUG: |-- AntiSandboxForegroundWindow

2018-01-22 12:15:03,709 [cuckoo.core.startup] DEBUG: |-- AntiSandboxIdleTime

2018-01-22 12:15:03,709 [cuckoo.core.startup] DEBUG: |-- AntiSandboxRestart

2018-01-22 12:15:03,709 [cuckoo.core.startup] DEBUG: |-- AntiSandboxSleep

2018-01-22 12:15:03,709 [cuckoo.core.startup] DEBUG: |-- AntiVMBios

2018-01-22 12:15:03,709 [cuckoo.core.startup] DEBUG: |-- AntiVMComputernameQuery

2018-01-22 12:15:03,710 [cuckoo.core.startup] DEBUG: |-- AntiVMCPU

2018-01-22 12:15:03,710 [cuckoo.core.startup] DEBUG: |-- AntiVMDiskSize

2018-01-22 12:15:03,710 [cuckoo.core.startup] DEBUG: |-- AntiVMIDE

2018-01-22 12:15:03,710 [cuckoo.core.startup] DEBUG: |-- AntiVMSCSI

2018-01-22 12:15:03,710 [cuckoo.core.startup] DEBUG: |-- AntiVMServices

2018-01-22 12:15:03,711 [cuckoo.core.startup] DEBUG: |-- AntiVMSharedDevice

2018-01-22 12:15:03,711 [cuckoo.core.startup] DEBUG: |-- AppLockerBypass

2018-01-22 12:15:03,711 [cuckoo.core.startup] DEBUG: |-- APT\_Carbunak

2018-01-22 12:15:03,711 [cuckoo.core.startup] DEBUG: |-- APT\_CloudAtlas

2018-01-22 12:15:03,711 [cuckoo.core.startup] DEBUG: |-- apt\_sandworm\_ip

2018-01-22 12:15:03,712 [cuckoo.core.startup] DEBUG: |-- apt\_sandworm\_url

2018-01-22 12:15:03,712 [cuckoo.core.startup] DEBUG: |-- ArdamaxMutexes

2018-01-22 12:15:03,712 [cuckoo.core.startup] DEBUG: |-- AthenaHttp

2018-01-22 12:15:03,712 [cuckoo.core.startup] DEBUG: |-- AthenaURL

2018-01-22 12:15:03,712 [cuckoo.core.startup] DEBUG: |-- Autorun

2018-01-22 12:15:03,713 [cuckoo.core.startup] DEBUG: |-- AvastDetectLibs

2018-01-22 12:15:03,713 [cuckoo.core.startup] DEBUG: |-- AVDetectionChinaKey

2018-01-22 12:15:03,713 [cuckoo.core.startup] DEBUG: |-- BadCerts

2018-01-22 12:15:03,713 [cuckoo.core.startup] DEBUG: |-- Bagle

2018-01-22 12:15:03,713 [cuckoo.core.startup] DEBUG: |-- Bandook

2018-01-22 12:15:03,714 [cuckoo.core.startup] DEBUG: |-- banker\_bancos

2018-01-22 12:15:03,714 [cuckoo.core.startup] DEBUG: |-- BankingMutexes

2018-01-22 12:15:03,714 [cuckoo.core.startup] DEBUG: |-- Banload

2018-01-22 12:15:03,714 [cuckoo.core.startup] DEBUG: |-- Beastdoor

2018-01-22 12:15:03,714 [cuckoo.core.startup] DEBUG: |-- BeebusMutexes

2018-01-22 12:15:03,715 [cuckoo.core.startup] DEBUG: |-- BegseabugTDMutexes

2018-01-22 12:15:03,715 [cuckoo.core.startup] DEBUG: |-- BetabotURL

2018-01-22 12:15:03,715 [cuckoo.core.startup] DEBUG: |-- Bifrose

2018-01-22 12:15:03,715 [cuckoo.core.startup] DEBUG: |-- BitcoinOpenCL

2018-01-22 12:15:03,715 [cuckoo.core.startup] DEBUG: |-- BitcoinWallet

2018-01-22 12:15:03,716 [cuckoo.core.startup] DEBUG: |-- BitdefenderDetectLibs

2018-01-22 12:15:03,716 [cuckoo.core.startup] DEBUG: |-- BlackEnergyMutexes

2018-01-22 12:15:03,716 [cuckoo.core.startup] DEBUG: |-- Blackhole

2018-01-22 12:15:03,716 [cuckoo.core.startup] DEBUG: |-- BlackholeURL

2018-01-22 12:15:03,717 [cuckoo.core.startup] DEBUG: |-- Blackice

2018-01-22 12:15:03,717 [cuckoo.core.startup] DEBUG: |-- BlackposURL

2018-01-22 12:15:03,717 [cuckoo.core.startup] DEBUG: |-- BlackRevMutexes

2018-01-22 12:15:03,717 [cuckoo.core.startup] DEBUG: |-- Blackshades

2018-01-22 12:15:03,718 [cuckoo.core.startup] DEBUG: |-- BladabindiMutexes

2018-01-22 12:15:03,718 [cuckoo.core.startup] DEBUG: |-- BochsDetectKeys

2018-01-22 12:15:03,718 [cuckoo.core.startup] DEBUG: |-- Bootkit

2018-01-22 12:15:03,718 [cuckoo.core.startup] DEBUG: |-- Bottilda

2018-01-22 12:15:03,718 [cuckoo.core.startup] DEBUG: |-- BozokKey

2018-01-22 12:15:03,719 [cuckoo.core.startup] DEBUG: |-- browser\_startpage

2018-01-22 12:15:03,719 [cuckoo.core.startup] DEBUG: |-- BrowserSecurity

2018-01-22 12:15:03,719 [cuckoo.core.startup] DEBUG: |-- BrowserStealer

2018-01-22 12:15:03,719 [cuckoo.core.startup] DEBUG: |-- Btcbotnet

2018-01-22 12:15:03,719 [cuckoo.core.startup] DEBUG: |-- Bublik

2018-01-22 12:15:03,719 [cuckoo.core.startup] DEBUG: |-- BuildLangID

2018-01-22 12:15:03,719 [cuckoo.core.startup] DEBUG: |-- BuzusMutexes

2018-01-22 12:15:03,719 [cuckoo.core.startup] DEBUG: |-- BypassFirewall

2018-01-22 12:15:03,720 [cuckoo.core.startup] DEBUG: |-- c24URL

2018-01-22 12:15:03,720 [cuckoo.core.startup] DEBUG: |-- CarberpMutexes

2018-01-22 12:15:03,720 [cuckoo.core.startup] DEBUG: |-- Ceatrg

2018-01-22 12:15:03,720 [cuckoo.core.startup] DEBUG: |-- ChanitorMutexes

2018-01-22 12:15:03,720 [cuckoo.core.startup] DEBUG: |-- CheckIP

2018-01-22 12:15:03,720 [cuckoo.core.startup] DEBUG: |-- cloud\_mediafire

2018-01-22 12:15:03,720 [cuckoo.core.startup] DEBUG: |-- cloud\_wetransfer

2018-01-22 12:15:03,721 [cuckoo.core.startup] DEBUG: |-- CloudFlare

2018-01-22 12:15:03,721 [cuckoo.core.startup] DEBUG: |-- CloudGoogle

2018-01-22 12:15:03,721 [cuckoo.core.startup] DEBUG: |-- CoinminerMutexes

2018-01-22 12:15:03,721 [cuckoo.core.startup] DEBUG: |-- ComRAT

2018-01-22 12:15:03,721 [cuckoo.core.startup] DEBUG: |-- Crash

2018-01-22 12:15:03,721 [cuckoo.core.startup] DEBUG: |-- CreatesAutorunInf

2018-01-22 12:15:03,721 [cuckoo.core.startup] DEBUG: |-- CreatesDocument

2018-01-22 12:15:03,722 [cuckoo.core.startup] DEBUG: |-- CreatesExe

2018-01-22 12:15:03,722 [cuckoo.core.startup] DEBUG: |-- CreatesLargeKey

2018-01-22 12:15:03,722 [cuckoo.core.startup] DEBUG: |-- CreatesService

2018-01-22 12:15:03,722 [cuckoo.core.startup] DEBUG: |-- CreatesShortcut

2018-01-22 12:15:03,722 [cuckoo.core.startup] DEBUG: |-- CreatesSuspiciousProcess

2018-01-22 12:15:03,722 [cuckoo.core.startup] DEBUG: |-- Cridex

2018-01-22 12:15:03,722 [cuckoo.core.startup] DEBUG: |-- CryptGenKey

2018-01-22 12:15:03,722 [cuckoo.core.startup] DEBUG: |-- Cryptolocker

2018-01-22 12:15:03,723 [cuckoo.core.startup] DEBUG: |-- CuckooDetectFiles

2018-01-22 12:15:03,723 [cuckoo.core.startup] DEBUG: |-- Cybergate

2018-01-22 12:15:03,723 [cuckoo.core.startup] DEBUG: |-- Dapato

2018-01-22 12:15:03,723 [cuckoo.core.startup] DEBUG: |-- Darkcloud

2018-01-22 12:15:03,723 [cuckoo.core.startup] DEBUG: |-- DarkddosMutexes

2018-01-22 12:15:03,723 [cuckoo.core.startup] DEBUG: |-- Darkshell

2018-01-22 12:15:03,723 [cuckoo.core.startup] DEBUG: |-- Ddos556

2018-01-22 12:15:03,724 [cuckoo.core.startup] DEBUG: |-- Decay

2018-01-22 12:15:03,724 [cuckoo.core.startup] DEBUG: |-- DecebalMutexes

2018-01-22 12:15:03,724 [cuckoo.core.startup] DEBUG: |-- DeletesSelf

2018-01-22 12:15:03,724 [cuckoo.core.startup] DEBUG: |-- DelfTrojan

2018-01-22 12:15:03,724 [cuckoo.core.startup] DEBUG: |-- DEPHeapBypass

2018-01-22 12:15:03,724 [cuckoo.core.startup] DEBUG: |-- DEPStackBypass

2018-01-22 12:15:03,724 [cuckoo.core.startup] DEBUG: |-- DerusbiMutexes

2018-01-22 12:15:03,725 [cuckoo.core.startup] DEBUG: |-- Dexter

2018-01-22 12:15:03,725 [cuckoo.core.startup] DEBUG: |-- Dibik

2018-01-22 12:15:03,725 [cuckoo.core.startup] DEBUG: |-- DirtJumper

2018-01-22 12:15:03,725 [cuckoo.core.startup] DEBUG: |-- DisableCmd

2018-01-22 12:15:03,725 [cuckoo.core.startup] DEBUG: |-- DisableRegedit

2018-01-22 12:15:03,725 [cuckoo.core.startup] DEBUG: |-- DisablesAppLaunch

2018-01-22 12:15:03,725 [cuckoo.core.startup] DEBUG: |-- DisablesBrowserWarn

2018-01-22 12:15:03,725 [cuckoo.core.startup] DEBUG: |-- DisablesSecurity

2018-01-22 12:15:03,726 [cuckoo.core.startup] DEBUG: |-- DisablesSPDY

2018-01-22 12:15:03,726 [cuckoo.core.startup] DEBUG: |-- DisablesSystemRestore

2018-01-22 12:15:03,726 [cuckoo.core.startup] DEBUG: |-- DisablesWER

2018-01-22 12:15:03,726 [cuckoo.core.startup] DEBUG: |-- DisablesWindowsUpdate

2018-01-22 12:15:03,726 [cuckoo.core.startup] DEBUG: |-- DisableTaskMgr

2018-01-22 12:15:03,726 [cuckoo.core.startup] DEBUG: |-- DiskInformation

2018-01-22 12:15:03,726 [cuckoo.core.startup] DEBUG: |-- DisplaysHTA

2018-01-22 12:15:03,727 [cuckoo.core.startup] DEBUG: |-- Dns\_Freehosting\_Domain

2018-01-22 12:15:03,727 [cuckoo.core.startup] DEBUG: |-- dnsserver\_dynamic

2018-01-22 12:15:03,727 [cuckoo.core.startup] DEBUG: |-- DocumentClose

2018-01-22 12:15:03,727 [cuckoo.core.startup] DEBUG: |-- DocumentOpen

2018-01-22 12:15:03,727 [cuckoo.core.startup] DEBUG: |-- DoFoil

2018-01-22 12:15:03,727 [cuckoo.core.startup] DEBUG: |-- DownloaderCabby

2018-01-22 12:15:03,727 [cuckoo.core.startup] DEBUG: |-- Dridex\_APIs

2018-01-22 12:15:03,727 [cuckoo.core.startup] DEBUG: |-- Drive

2018-01-22 12:15:03,728 [cuckoo.core.startup] DEBUG: |-- Drive2

2018-01-22 12:15:03,728 [cuckoo.core.startup] DEBUG: |-- DriverLoad

2018-01-22 12:15:03,728 [cuckoo.core.startup] DEBUG: |-- DropBox

2018-01-22 12:15:03,728 [cuckoo.core.startup] DEBUG: |-- Dropper

2018-01-22 12:15:03,728 [cuckoo.core.startup] DEBUG: |-- Dyreza

2018-01-22 12:15:03,728 [cuckoo.core.startup] DEBUG: |-- EclipseMutexes

2018-01-22 12:15:03,728 [cuckoo.core.startup] DEBUG: |-- Emotet

2018-01-22 12:15:03,729 [cuckoo.core.startup] DEBUG: |-- Emotet\_APIs

2018-01-22 12:15:03,729 [cuckoo.core.startup] DEBUG: |-- Evilbot

2018-01-22 12:15:03,729 [cuckoo.core.startup] DEBUG: |-- ExecBitsAdmin

2018-01-22 12:15:03,729 [cuckoo.core.startup] DEBUG: |-- ExecWaitFor

2018-01-22 12:15:03,729 [cuckoo.core.startup] DEBUG: |-- exp\_3322\_dom

2018-01-22 12:15:03,729 [cuckoo.core.startup] DEBUG: |-- Expiro

2018-01-22 12:15:03,729 [cuckoo.core.startup] DEBUG: |-- ExploitHeapspray

2018-01-22 12:15:03,729 [cuckoo.core.startup] DEBUG: |-- ExploitKitMutexes

2018-01-22 12:15:03,730 [cuckoo.core.startup] DEBUG: |-- FakeAVMutexes

2018-01-22 12:15:03,730 [cuckoo.core.startup] DEBUG: |-- FakeAVMutexes

2018-01-22 12:15:03,730 [cuckoo.core.startup] DEBUG: |-- FakeRean

2018-01-22 12:15:03,730 [cuckoo.core.startup] DEBUG: |-- FarFli

2018-01-22 12:15:03,730 [cuckoo.core.startup] DEBUG: |-- FesberMutexes

2018-01-22 12:15:03,730 [cuckoo.core.startup] DEBUG: |-- Fingerprint

2018-01-22 12:15:03,730 [cuckoo.core.startup] DEBUG: |-- Flame

2018-01-22 12:15:03,731 [cuckoo.core.startup] DEBUG: |-- Flystudio

2018-01-22 12:15:03,731 [cuckoo.core.startup] DEBUG: |-- FortinetDetectFiles

2018-01-22 12:15:03,731 [cuckoo.core.startup] DEBUG: |-- FTPStealer

2018-01-22 12:15:03,731 [cuckoo.core.startup] DEBUG: |-- Fynloski

2018-01-22 12:15:03,731 [cuckoo.core.startup] DEBUG: |-- Gaelicum

2018-01-22 12:15:03,731 [cuckoo.core.startup] DEBUG: |-- Ghostbot

2018-01-22 12:15:03,731 [cuckoo.core.startup] DEBUG: |-- HasAuthenticode

2018-01-22 12:15:03,732 [cuckoo.core.startup] DEBUG: |-- HasOfficeEps

2018-01-22 12:15:03,732 [cuckoo.core.startup] DEBUG: |-- HasPdb

2018-01-22 12:15:03,732 [cuckoo.core.startup] DEBUG: |-- HasWMI

2018-01-22 12:15:03,732 [cuckoo.core.startup] DEBUG: |-- Hesperbot

2018-01-22 12:15:03,732 [cuckoo.core.startup] DEBUG: |-- Hidden\_Window

2018-01-22 12:15:03,732 [cuckoo.core.startup] DEBUG: |-- Hikit

2018-01-22 12:15:03,732 [cuckoo.core.startup] DEBUG: |-- HookMouse

2018-01-22 12:15:03,732 [cuckoo.core.startup] DEBUG: |-- Hupigon

2018-01-22 12:15:03,733 [cuckoo.core.startup] DEBUG: |-- HyperVDetectKeys

2018-01-22 12:15:03,733 [cuckoo.core.startup] DEBUG: |-- IcePoint

2018-01-22 12:15:03,733 [cuckoo.core.startup] DEBUG: |-- im\_btb

2018-01-22 12:15:03,733 [cuckoo.core.startup] DEBUG: |-- im\_qq

2018-01-22 12:15:03,733 [cuckoo.core.startup] DEBUG: |-- IMStealer

2018-01-22 12:15:03,733 [cuckoo.core.startup] DEBUG: |-- InceptionAPT

2018-01-22 12:15:03,733 [cuckoo.core.startup] DEBUG: |-- Infinity

2018-01-22 12:15:03,734 [cuckoo.core.startup] DEBUG: |-- InjectionRunPE

2018-01-22 12:15:03,734 [cuckoo.core.startup] DEBUG: |-- InjectionThread

2018-01-22 12:15:03,734 [cuckoo.core.startup] DEBUG: |-- InstalledApps

2018-01-22 12:15:03,734 [cuckoo.core.startup] DEBUG: |-- InstallsAppInit

2018-01-22 12:15:03,734 [cuckoo.core.startup] DEBUG: |-- InstallsBHO

2018-01-22 12:15:03,734 [cuckoo.core.startup] DEBUG: |-- InstallsWinpcap

2018-01-22 12:15:03,734 [cuckoo.core.startup] DEBUG: |-- IPKillerMutexes

2018-01-22 12:15:03,735 [cuckoo.core.startup] DEBUG: |-- Ircbrute

2018-01-22 12:15:03,735 [cuckoo.core.startup] DEBUG: |-- ISRstealerURL

2018-01-22 12:15:03,735 [cuckoo.core.startup] DEBUG: |-- iStealerURL

2018-01-22 12:15:03,735 [cuckoo.core.startup] DEBUG: |-- JackPOSFile

2018-01-22 12:15:03,735 [cuckoo.core.startup] DEBUG: |-- JackposURL

2018-01-22 12:15:03,735 [cuckoo.core.startup] DEBUG: |-- JeefoMutexes

2018-01-22 12:15:03,735 [cuckoo.core.startup] DEBUG: |-- Jewdo

2018-01-22 12:15:03,735 [cuckoo.core.startup] DEBUG: |-- JintorMutexes

2018-01-22 12:15:03,736 [cuckoo.core.startup] DEBUG: |-- JorikTrojan

2018-01-22 12:15:03,736 [cuckoo.core.startup] DEBUG: |-- Karagany

2018-01-22 12:15:03,736 [cuckoo.core.startup] DEBUG: |-- Karakum

2018-01-22 12:15:03,736 [cuckoo.core.startup] DEBUG: |-- Katusha

2018-01-22 12:15:03,736 [cuckoo.core.startup] DEBUG: |-- KelihosBot

2018-01-22 12:15:03,736 [cuckoo.core.startup] DEBUG: |-- Keylogger

2018-01-22 12:15:03,736 [cuckoo.core.startup] DEBUG: |-- Kilim

2018-01-22 12:15:03,737 [cuckoo.core.startup] DEBUG: |-- Killdisk

2018-01-22 12:15:03,737 [cuckoo.core.startup] DEBUG: |-- KnownVirustotal

2018-01-22 12:15:03,737 [cuckoo.core.startup] DEBUG: |-- Koobface

2018-01-22 12:15:03,737 [cuckoo.core.startup] DEBUG: |-- Koutodoor

2018-01-22 12:15:03,737 [cuckoo.core.startup] DEBUG: |-- KovterBot

2018-01-22 12:15:03,737 [cuckoo.core.startup] DEBUG: |-- KrepperMutexes

2018-01-22 12:15:03,737 [cuckoo.core.startup] DEBUG: |-- KuluozMutexes

2018-01-22 12:15:03,738 [cuckoo.core.startup] DEBUG: |-- Likseput

2018-01-22 12:15:03,738 [cuckoo.core.startup] DEBUG: |-- LocatesBrowser

2018-01-22 12:15:03,738 [cuckoo.core.startup] DEBUG: |-- LocatesSniffer

2018-01-22 12:15:03,738 [cuckoo.core.startup] DEBUG: |-- Lockscreen

2018-01-22 12:15:03,738 [cuckoo.core.startup] DEBUG: |-- LolBot

2018-01-22 12:15:03,738 [cuckoo.core.startup] DEBUG: |-- Luder

2018-01-22 12:15:03,738 [cuckoo.core.startup] DEBUG: |-- Madness

2018-01-22 12:15:03,738 [cuckoo.core.startup] DEBUG: |-- Madness

2018-01-22 12:15:03,739 [cuckoo.core.startup] DEBUG: |-- MadnessURL

2018-01-22 12:15:03,739 [cuckoo.core.startup] DEBUG: |-- MaganiaMutexes

2018-01-22 12:15:03,739 [cuckoo.core.startup] DEBUG: |-- MailStealer

2018-01-22 12:15:03,739 [cuckoo.core.startup] DEBUG: |-- MaliciousDocumentURLs

2018-01-22 12:15:03,739 [cuckoo.core.startup] DEBUG: |-- MegaUpload

2018-01-22 12:15:03,739 [cuckoo.core.startup] DEBUG: |-- MemoryAvailable

2018-01-22 12:15:03,739 [cuckoo.core.startup] DEBUG: |-- MetasploitShellcode

2018-01-22 12:15:03,740 [cuckoo.core.startup] DEBUG: |-- Minerbot

2018-01-22 12:15:03,740 [cuckoo.core.startup] DEBUG: |-- miningpool

2018-01-22 12:15:03,740 [cuckoo.core.startup] DEBUG: |-- MircFile

2018-01-22 12:15:03,740 [cuckoo.core.startup] DEBUG: |-- ModifiesBootConfig

2018-01-22 12:15:03,740 [cuckoo.core.startup] DEBUG: |-- ModifiesCertificates

2018-01-22 12:15:03,740 [cuckoo.core.startup] DEBUG: |-- ModifiesDesktopWallpaper

2018-01-22 12:15:03,740 [cuckoo.core.startup] DEBUG: |-- ModifiesUACNotify

2018-01-22 12:15:03,741 [cuckoo.core.startup] DEBUG: |-- ModifySecurityCenterWarnings

2018-01-22 12:15:03,741 [cuckoo.core.startup] DEBUG: |-- Multiple\_UA

2018-01-22 12:15:03,741 [cuckoo.core.startup] DEBUG: |-- MyBot

2018-01-22 12:15:03,741 [cuckoo.core.startup] DEBUG: |-- Nakbot

2018-01-22 12:15:03,741 [cuckoo.core.startup] DEBUG: |-- Napolar

2018-01-22 12:15:03,741 [cuckoo.core.startup] DEBUG: |-- Nebuler

2018-01-22 12:15:03,741 [cuckoo.core.startup] DEBUG: |-- Netobserve

2018-01-22 12:15:03,742 [cuckoo.core.startup] DEBUG: |-- Netshadow

2018-01-22 12:15:03,742 [cuckoo.core.startup] DEBUG: |-- Netwire

2018-01-22 12:15:03,742 [cuckoo.core.startup] DEBUG: |-- NetworkAdapters

2018-01-22 12:15:03,742 [cuckoo.core.startup] DEBUG: |-- NetworkC2Details

2018-01-22 12:15:03,742 [cuckoo.core.startup] DEBUG: |-- NetworkDocumentFile

2018-01-22 12:15:03,742 [cuckoo.core.startup] DEBUG: |-- NetworkEXE

2018-01-22 12:15:03,742 [cuckoo.core.startup] DEBUG: |-- Nitol

2018-01-22 12:15:03,742 [cuckoo.core.startup] DEBUG: |-- NjRat

2018-01-22 12:15:03,743 [cuckoo.core.startup] DEBUG: |-- ObfusMutexes

2018-01-22 12:15:03,743 [cuckoo.core.startup] DEBUG: |-- OfficeCheckName

2018-01-22 12:15:03,743 [cuckoo.core.startup] DEBUG: |-- OfficeCheckProjectName

2018-01-22 12:15:03,743 [cuckoo.core.startup] DEBUG: |-- OfficeCheckVersion

2018-01-22 12:15:03,743 [cuckoo.core.startup] DEBUG: |-- OfficeCheckWindow

2018-01-22 12:15:03,743 [cuckoo.core.startup] DEBUG: |-- OfficeCountDirectories

2018-01-22 12:15:03,743 [cuckoo.core.startup] DEBUG: |-- OfficeCreateObject

2018-01-22 12:15:03,744 [cuckoo.core.startup] DEBUG: |-- OfficeDDE

2018-01-22 12:15:03,744 [cuckoo.core.startup] DEBUG: |-- OfficeEpsStrings

2018-01-22 12:15:03,744 [cuckoo.core.startup] DEBUG: |-- OfficeHttpRequest

2018-01-22 12:15:03,744 [cuckoo.core.startup] DEBUG: |-- OfficeIndirectCall

2018-01-22 12:15:03,744 [cuckoo.core.startup] DEBUG: |-- OfficePackager

2018-01-22 12:15:03,744 [cuckoo.core.startup] DEBUG: |-- OfficePlatformDetect

2018-01-22 12:15:03,744 [cuckoo.core.startup] DEBUG: |-- OfficeRecentFiles

2018-01-22 12:15:03,744 [cuckoo.core.startup] DEBUG: |-- OfficeVulnerableGuid

2018-01-22 12:15:03,745 [cuckoo.core.startup] DEBUG: |-- OfficeVulnModules

2018-01-22 12:15:03,745 [cuckoo.core.startup] DEBUG: |-- Oldrea

2018-01-22 12:15:03,745 [cuckoo.core.startup] DEBUG: |-- OverwritesFiles

2018-01-22 12:15:03,745 [cuckoo.core.startup] DEBUG: |-- PackerEntropy

2018-01-22 12:15:03,745 [cuckoo.core.startup] DEBUG: |-- Palevo

2018-01-22 12:15:03,745 [cuckoo.core.startup] DEBUG: |-- ParallelsDetectKeys

2018-01-22 12:15:03,745 [cuckoo.core.startup] DEBUG: |-- Pasta

2018-01-22 12:15:03,746 [cuckoo.core.startup] DEBUG: |-- PcClientMutexes

2018-01-22 12:15:03,746 [cuckoo.core.startup] DEBUG: |-- PEFeatures

2018-01-22 12:15:03,746 [cuckoo.core.startup] DEBUG: |-- PEIDPacker

2018-01-22 12:15:03,746 [cuckoo.core.startup] DEBUG: |-- PerfLogger

2018-01-22 12:15:03,746 [cuckoo.core.startup] DEBUG: |-- PersistanceRegJavaScript

2018-01-22 12:15:03,746 [cuckoo.core.startup] DEBUG: |-- PersistenceBootexecute

2018-01-22 12:15:03,746 [cuckoo.core.startup] DEBUG: |-- Phorpiex

2018-01-22 12:15:03,747 [cuckoo.core.startup] DEBUG: |-- Pidief

2018-01-22 12:15:03,747 [cuckoo.core.startup] DEBUG: |-- Plugx

2018-01-22 12:15:03,747 [cuckoo.core.startup] DEBUG: |-- Poebot

2018-01-22 12:15:03,747 [cuckoo.core.startup] DEBUG: |-- PoisonIvy

2018-01-22 12:15:03,747 [cuckoo.core.startup] DEBUG: |-- Polymorphic

2018-01-22 12:15:03,747 [cuckoo.core.startup] DEBUG: |-- Ponfoy

2018-01-22 12:15:03,747 [cuckoo.core.startup] DEBUG: |-- PonyURL

2018-01-22 12:15:03,747 [cuckoo.core.startup] DEBUG: |-- PosCardStealerURL

2018-01-22 12:15:03,748 [cuckoo.core.startup] DEBUG: |-- Powerfun

2018-01-22 12:15:03,748 [cuckoo.core.startup] DEBUG: |-- PowershellBitsTransfer

2018-01-22 12:15:03,748 [cuckoo.core.startup] DEBUG: |-- PowershellCcDns

2018-01-22 12:15:03,748 [cuckoo.core.startup] DEBUG: |-- PowershellDdiRc4

2018-01-22 12:15:03,748 [cuckoo.core.startup] DEBUG: |-- PowershellDFSP

2018-01-22 12:15:03,748 [cuckoo.core.startup] DEBUG: |-- PowershellDI

2018-01-22 12:15:03,748 [cuckoo.core.startup] DEBUG: |-- PowershellDownload

2018-01-22 12:15:03,749 [cuckoo.core.startup] DEBUG: |-- PowershellEmpire

2018-01-22 12:15:03,749 [cuckoo.core.startup] DEBUG: |-- PowershellMeterpreter

2018-01-22 12:15:03,749 [cuckoo.core.startup] DEBUG: |-- PowershellRegAdd

2018-01-22 12:15:03,749 [cuckoo.core.startup] DEBUG: |-- PowershellRequest

2018-01-22 12:15:03,749 [cuckoo.core.startup] DEBUG: |-- PowershellUnicorn

2018-01-22 12:15:03,749 [cuckoo.core.startup] DEBUG: |-- Powerworm

2018-01-22 12:15:03,749 [cuckoo.core.startup] DEBUG: |-- Prinimalka

2018-01-22 12:15:03,749 [cuckoo.core.startup] DEBUG: |-- ProcessInterest

2018-01-22 12:15:03,750 [cuckoo.core.startup] DEBUG: |-- ProcessMartian

2018-01-22 12:15:03,750 [cuckoo.core.startup] DEBUG: |-- ProcessNeeded

2018-01-22 12:15:03,750 [cuckoo.core.startup] DEBUG: |-- ProcMemDumpIPURLs

2018-01-22 12:15:03,750 [cuckoo.core.startup] DEBUG: |-- ProcMemDumpTORURLs

2018-01-22 12:15:03,750 [cuckoo.core.startup] DEBUG: |-- ProcMemDumpURLs

2018-01-22 12:15:03,750 [cuckoo.core.startup] DEBUG: |-- ProcMemDumpYara

2018-01-22 12:15:03,750 [cuckoo.core.startup] DEBUG: |-- Psyokym

2018-01-22 12:15:03,751 [cuckoo.core.startup] DEBUG: |-- PuceMutexes

2018-01-22 12:15:03,751 [cuckoo.core.startup] DEBUG: |-- PutterpandaMutexes

2018-01-22 12:15:03,751 [cuckoo.core.startup] DEBUG: |-- Putty

2018-01-22 12:15:03,751 [cuckoo.core.startup] DEBUG: |-- PWDumpFile

2018-01-22 12:15:03,751 [cuckoo.core.startup] DEBUG: |-- Pykse

2018-01-22 12:15:03,751 [cuckoo.core.startup] DEBUG: |-- Qakbot

2018-01-22 12:15:03,751 [cuckoo.core.startup] DEBUG: |-- Ragebot

2018-01-22 12:15:03,752 [cuckoo.core.startup] DEBUG: |-- RaisesException

2018-01-22 12:15:03,752 [cuckoo.core.startup] DEBUG: |-- Ramnit

2018-01-22 12:15:03,752 [cuckoo.core.startup] DEBUG: |-- RamsomwareFileMoves

2018-01-22 12:15:03,752 [cuckoo.core.startup] DEBUG: |-- ransomware\_viruscoder

2018-01-22 12:15:03,752 [cuckoo.core.startup] DEBUG: |-- RansomwareAppendsExtension

2018-01-22 12:15:03,752 [cuckoo.core.startup] DEBUG: |-- RansomwareBcdedit

2018-01-22 12:15:03,752 [cuckoo.core.startup] DEBUG: |-- RansomwareDroppedFiles

2018-01-22 12:15:03,753 [cuckoo.core.startup] DEBUG: |-- RansomwareExtensions

2018-01-22 12:15:03,753 [cuckoo.core.startup] DEBUG: |-- RansomwareFiles

2018-01-22 12:15:03,753 [cuckoo.core.startup] DEBUG: |-- RansomwareMessage

2018-01-22 12:15:03,753 [cuckoo.core.startup] DEBUG: |-- RansomwareRecyclebin

2018-01-22 12:15:03,753 [cuckoo.core.startup] DEBUG: |-- RansomwareShadowcopy

2018-01-22 12:15:03,753 [cuckoo.core.startup] DEBUG: |-- RapidShare

2018-01-22 12:15:03,753 [cuckoo.core.startup] DEBUG: |-- rat\_fexel\_ip

2018-01-22 12:15:03,753 [cuckoo.core.startup] DEBUG: |-- rat\_naid\_ip

2018-01-22 12:15:03,754 [cuckoo.core.startup] DEBUG: |-- RatSiggen

2018-01-22 12:15:03,754 [cuckoo.core.startup] DEBUG: |-- RBot

2018-01-22 12:15:03,754 [cuckoo.core.startup] DEBUG: |-- RdpMutexes

2018-01-22 12:15:03,754 [cuckoo.core.startup] DEBUG: |-- Recon\_Beacon

2018-01-22 12:15:03,754 [cuckoo.core.startup] DEBUG: |-- RemovesZoneIdADS

2018-01-22 12:15:03,754 [cuckoo.core.startup] DEBUG: |-- Renocide

2018-01-22 12:15:03,754 [cuckoo.core.startup] DEBUG: |-- RenosTrojan

2018-01-22 12:15:03,755 [cuckoo.core.startup] DEBUG: |-- Rovnix

2018-01-22 12:15:03,755 [cuckoo.core.startup] DEBUG: |-- Runbu

2018-01-22 12:15:03,755 [cuckoo.core.startup] DEBUG: |-- RunouceMutexes

2018-01-22 12:15:03,755 [cuckoo.core.startup] DEBUG: |-- Ruskill

2018-01-22 12:15:03,755 [cuckoo.core.startup] DEBUG: |-- Sadbot

2018-01-22 12:15:03,755 [cuckoo.core.startup] DEBUG: |-- SandboxieDetect

2018-01-22 12:15:03,755 [cuckoo.core.startup] DEBUG: |-- SandboxJoeAnubisDetectFiles

2018-01-22 12:15:03,755 [cuckoo.core.startup] DEBUG: |-- SDBot

2018-01-22 12:15:03,756 [cuckoo.core.startup] DEBUG: |-- SelfDeleteBat

2018-01-22 12:15:03,756 [cuckoo.core.startup] DEBUG: |-- Senna

2018-01-22 12:15:03,756 [cuckoo.core.startup] DEBUG: |-- Shadowbot

2018-01-22 12:15:03,756 [cuckoo.core.startup] DEBUG: |-- SharingRGhost

2018-01-22 12:15:03,756 [cuckoo.core.startup] DEBUG: |-- SharpStealerURL

2018-01-22 12:15:03,756 [cuckoo.core.startup] DEBUG: |-- ShellcodeWriteProcessMemory

2018-01-22 12:15:03,756 [cuckoo.core.startup] DEBUG: |-- Shiz

2018-01-22 12:15:03,757 [cuckoo.core.startup] DEBUG: |-- Shylock

2018-01-22 12:15:03,757 [cuckoo.core.startup] DEBUG: |-- SipStun

2018-01-22 12:15:03,757 [cuckoo.core.startup] DEBUG: |-- Smtp\_GMail

2018-01-22 12:15:03,757 [cuckoo.core.startup] DEBUG: |-- Smtp\_Live

2018-01-22 12:15:03,757 [cuckoo.core.startup] DEBUG: |-- Smtp\_Mail\_Ru

2018-01-22 12:15:03,757 [cuckoo.core.startup] DEBUG: |-- Smtp\_Yahoo

2018-01-22 12:15:03,757 [cuckoo.core.startup] DEBUG: |-- SolarURL

2018-01-22 12:15:03,757 [cuckoo.core.startup] DEBUG: |-- SpyEyeMutexes

2018-01-22 12:15:03,758 [cuckoo.core.startup] DEBUG: |-- SpyeyeURL

2018-01-22 12:15:03,758 [cuckoo.core.startup] DEBUG: |-- SpynetRat

2018-01-22 12:15:03,758 [cuckoo.core.startup] DEBUG: |-- Spyrecorder

2018-01-22 12:15:03,758 [cuckoo.core.startup] DEBUG: |-- StackPivot

2018-01-22 12:15:03,758 [cuckoo.core.startup] DEBUG: |-- StackPivotDllLoad

2018-01-22 12:15:03,758 [cuckoo.core.startup] DEBUG: |-- Staser

2018-01-22 12:15:03,758 [cuckoo.core.startup] DEBUG: |-- StealthChildProc

2018-01-22 12:15:03,759 [cuckoo.core.startup] DEBUG: |-- StealthHiddenExtension

2018-01-22 12:15:03,759 [cuckoo.core.startup] DEBUG: |-- StealthHiddenFile

2018-01-22 12:15:03,759 [cuckoo.core.startup] DEBUG: |-- StealthHiddenIcons

2018-01-22 12:15:03,759 [cuckoo.core.startup] DEBUG: |-- StopsService

2018-01-22 12:15:03,759 [cuckoo.core.startup] DEBUG: |-- SunbeltDetectFiles

2018-01-22 12:15:03,759 [cuckoo.core.startup] DEBUG: |-- SunBeltSandboxDetect

2018-01-22 12:15:03,759 [cuckoo.core.startup] DEBUG: |-- SuspiciousCommandTools

2018-01-22 12:15:03,760 [cuckoo.core.startup] DEBUG: |-- SuspiciousPowershell

2018-01-22 12:15:03,760 [cuckoo.core.startup] DEBUG: |-- SuspiciousWriteEXE

2018-01-22 12:15:03,760 [cuckoo.core.startup] DEBUG: |-- SweetorangeMutexes

2018-01-22 12:15:03,760 [cuckoo.core.startup] DEBUG: |-- Swrort

2018-01-22 12:15:03,760 [cuckoo.core.startup] DEBUG: |-- SystemInfo

2018-01-22 12:15:03,760 [cuckoo.core.startup] DEBUG: |-- SystemMetrics

2018-01-22 12:15:03,760 [cuckoo.core.startup] DEBUG: |-- TapiDpMutexes

2018-01-22 12:15:03,760 [cuckoo.core.startup] DEBUG: |-- TDSSBackdoor

2018-01-22 12:15:03,761 [cuckoo.core.startup] DEBUG: |-- TeamviewerRat

2018-01-22 12:15:03,761 [cuckoo.core.startup] DEBUG: |-- ThreatTrackDetectFiles

2018-01-22 12:15:03,761 [cuckoo.core.startup] DEBUG: |-- TinbaMutexes

2018-01-22 12:15:03,761 [cuckoo.core.startup] DEBUG: |-- TnegaMutexes

2018-01-22 12:15:03,761 [cuckoo.core.startup] DEBUG: |-- Tor

2018-01-22 12:15:03,761 [cuckoo.core.startup] DEBUG: |-- TorHiddenService

2018-01-22 12:15:03,761 [cuckoo.core.startup] DEBUG: |-- Travnet

2018-01-22 12:15:03,762 [cuckoo.core.startup] DEBUG: |-- Trogbot

2018-01-22 12:15:03,762 [cuckoo.core.startup] DEBUG: |-- TrojanJorik

2018-01-22 12:15:03,762 [cuckoo.core.startup] DEBUG: |-- TrojanLethic

2018-01-22 12:15:03,762 [cuckoo.core.startup] DEBUG: |-- TrojanLethic

2018-01-22 12:15:03,762 [cuckoo.core.startup] DEBUG: |-- trojanmrblack

2018-01-22 12:15:03,762 [cuckoo.core.startup] DEBUG: |-- TrojanRedosru

2018-01-22 12:15:03,762 [cuckoo.core.startup] DEBUG: |-- TrojanSysn

2018-01-22 12:15:03,762 [cuckoo.core.startup] DEBUG: |-- trojanyoddos

2018-01-22 12:15:03,763 [cuckoo.core.startup] DEBUG: |-- TufikMutexes

2018-01-22 12:15:03,763 [cuckoo.core.startup] DEBUG: |-- Turkojan

2018-01-22 12:15:03,763 [cuckoo.core.startup] DEBUG: |-- TurlaCarbon

2018-01-22 12:15:03,763 [cuckoo.core.startup] DEBUG: |-- UFRStealer

2018-01-22 12:15:03,763 [cuckoo.core.startup] DEBUG: |-- Unhook

2018-01-22 12:15:03,763 [cuckoo.core.startup] DEBUG: |-- Upatre

2018-01-22 12:15:03,763 [cuckoo.core.startup] DEBUG: |-- UpatreTDMutexes

2018-01-22 12:15:03,764 [cuckoo.core.startup] DEBUG: |-- UPXCompressed

2018-01-22 12:15:03,764 [cuckoo.core.startup] DEBUG: |-- UrkShortCN

2018-01-22 12:15:03,764 [cuckoo.core.startup] DEBUG: |-- URLSpy

2018-01-22 12:15:03,764 [cuckoo.core.startup] DEBUG: |-- UroburosFile

2018-01-22 12:15:03,764 [cuckoo.core.startup] DEBUG: |-- UroburosMutexes

2018-01-22 12:15:03,764 [cuckoo.core.startup] DEBUG: |-- Urxbot

2018-01-22 12:15:03,764 [cuckoo.core.startup] DEBUG: |-- UsesWindowsUtilities

2018-01-22 12:15:03,764 [cuckoo.core.startup] DEBUG: |-- Vanbot

2018-01-22 12:15:03,765 [cuckoo.core.startup] DEBUG: |-- VBInject

2018-01-22 12:15:03,765 [cuckoo.core.startup] DEBUG: |-- VBoxDetectACPI

2018-01-22 12:15:03,765 [cuckoo.core.startup] DEBUG: |-- VBoxDetectDevices

2018-01-22 12:15:03,765 [cuckoo.core.startup] DEBUG: |-- VBoxDetectFiles

2018-01-22 12:15:03,765 [cuckoo.core.startup] DEBUG: |-- VBoxDetectKeys

2018-01-22 12:15:03,765 [cuckoo.core.startup] DEBUG: |-- VBoxDetectProvname

2018-01-22 12:15:03,765 [cuckoo.core.startup] DEBUG: |-- VBoxDetectWindow

2018-01-22 12:15:03,765 [cuckoo.core.startup] DEBUG: |-- Vertex

2018-01-22 12:15:03,766 [cuckoo.core.startup] DEBUG: |-- VertexSolarURL

2018-01-22 12:15:03,766 [cuckoo.core.startup] DEBUG: |-- VirtualPCDetect

2018-01-22 12:15:03,766 [cuckoo.core.startup] DEBUG: |-- VirtualPCIllegalInstruction

2018-01-22 12:15:03,766 [cuckoo.core.startup] DEBUG: |-- Virut

2018-01-22 12:15:03,766 [cuckoo.core.startup] DEBUG: |-- VMFirmware

2018-01-22 12:15:03,766 [cuckoo.core.startup] DEBUG: |-- VMPPacked

2018-01-22 12:15:03,766 [cuckoo.core.startup] DEBUG: |-- VMWareDetectFiles

2018-01-22 12:15:03,767 [cuckoo.core.startup] DEBUG: |-- VMWareDetectKeys

2018-01-22 12:15:03,767 [cuckoo.core.startup] DEBUG: |-- VMWareInInstruction

2018-01-22 12:15:03,767 [cuckoo.core.startup] DEBUG: |-- VncMutexes

2018-01-22 12:15:03,767 [cuckoo.core.startup] DEBUG: |-- VNLoaderURL

2018-01-22 12:15:03,767 [cuckoo.core.startup] DEBUG: |-- VolDevicetree1

2018-01-22 12:15:03,767 [cuckoo.core.startup] DEBUG: |-- VolHandles1

2018-01-22 12:15:03,767 [cuckoo.core.startup] DEBUG: |-- VolLdrModules1

2018-01-22 12:15:03,767 [cuckoo.core.startup] DEBUG: |-- VolLdrModules2

2018-01-22 12:15:03,768 [cuckoo.core.startup] DEBUG: |-- VolMalfind1

2018-01-22 12:15:03,768 [cuckoo.core.startup] DEBUG: |-- VolModscan1

2018-01-22 12:15:03,768 [cuckoo.core.startup] DEBUG: |-- VolSvcscan1

2018-01-22 12:15:03,768 [cuckoo.core.startup] DEBUG: |-- VolSvcscan2

2018-01-22 12:15:03,768 [cuckoo.core.startup] DEBUG: |-- VolSvcscan3

2018-01-22 12:15:03,768 [cuckoo.core.startup] DEBUG: |-- VPCDetectKeys

2018-01-22 12:15:03,768 [cuckoo.core.startup] DEBUG: |-- Wakbot

2018-01-22 12:15:03,769 [cuckoo.core.startup] DEBUG: |-- WarbotURL

2018-01-22 12:15:03,769 [cuckoo.core.startup] DEBUG: |-- Whimoo

2018-01-22 12:15:03,769 [cuckoo.core.startup] DEBUG: |-- Win32ProcessCreate

2018-01-22 12:15:03,769 [cuckoo.core.startup] DEBUG: |-- WineDetect

2018-01-22 12:15:03,769 [cuckoo.core.startup] DEBUG: |-- WinSCP

2018-01-22 12:15:03,769 [cuckoo.core.startup] DEBUG: |-- WinSxsBot

2018-01-22 12:15:03,769 [cuckoo.core.startup] DEBUG: |-- WMIAntiVM

2018-01-22 12:15:03,769 [cuckoo.core.startup] DEBUG: |-- WormAllaple

2018-01-22 12:15:03,770 [cuckoo.core.startup] DEBUG: |-- WormKolabc

2018-01-22 12:15:03,770 [cuckoo.core.startup] DEBUG: |-- XenDetectKeys

2018-01-22 12:15:03,770 [cuckoo.core.startup] DEBUG: |-- XtremeRAT

2018-01-22 12:15:03,770 [cuckoo.core.startup] DEBUG: |-- Xworm

2018-01-22 12:15:03,770 [cuckoo.core.startup] DEBUG: |-- Zegost

2018-01-22 12:15:03,770 [cuckoo.core.startup] DEBUG: |-- ZeusMutexes

2018-01-22 12:15:03,770 [cuckoo.core.startup] DEBUG: |-- ZeusP2P

2018-01-22 12:15:03,771 [cuckoo.core.startup] DEBUG: |-- ZeusURL

2018-01-22 12:15:03,771 [cuckoo.core.startup] DEBUG: `-- ZoneID

2018-01-22 12:15:03,771 [cuckoo.core.startup] DEBUG: Imported "reporting" modules:

2018-01-22 12:15:03,771 [cuckoo.core.startup] DEBUG: |-- ElasticSearch

2018-01-22 12:15:03,771 [cuckoo.core.startup] DEBUG: |-- Feedback

2018-01-22 12:15:03,771 [cuckoo.core.startup] DEBUG: |-- JsonDump

2018-01-22 12:15:03,771 [cuckoo.core.startup] DEBUG: |-- Mattermost

2018-01-22 12:15:03,771 [cuckoo.core.startup] DEBUG: |-- MISP

2018-01-22 12:15:03,772 [cuckoo.core.startup] DEBUG: |-- Moloch

2018-01-22 12:15:03,772 [cuckoo.core.startup] DEBUG: |-- MongoDB

2018-01-22 12:15:03,772 [cuckoo.core.startup] DEBUG: |-- Notification

2018-01-22 12:15:03,772 [cuckoo.core.startup] DEBUG: Initializing Yara...

2018-01-22 12:15:03,772 [cuckoo.core.startup] DEBUG: |-- binaries embedded.yar

2018-01-22 12:15:04,876 [cuckoo.core.startup] DEBUG: |-- binaries shellcodes.yar

2018-01-22 12:15:04,877 [cuckoo.core.startup] DEBUG: |-- binaries vmdetect.yar

def init\_modules():  
 *"""Initializes plugins."""* ... 上面的代码之后  
 RunSignatures.init\_once()

@classmethod  
def init\_once(cls):  
 cls.available\_signatures = []  
  
 # Gather all enabled & up-to-date Signatures.  
 for signature in cuckoo.signatures:  
 if cls.should\_load\_signature(signature):  
 cls.available\_signatures.append(signature)  
  
 # Sort Signatures by their order.  
 cls.available\_signatures.sort(key=lambda sig: sig.order)

判断signatures是否有效，然后放进这个类的avilable\_signatures属性中

@classmethod  
def should\_load\_signature(cls, signature):  
 *"""Should the given signature be enabled for this analysis?"""* if not signature.enabled or signature.name is None:  
 return False  
  
 if not cls.check\_signature\_version(signature):  
 return False  
  
 if hasattr(signature, "enable") and callable(signature.enable):  
 if not signature.enable():  
 return False  
  
 return True

有效的条件为signature的enable为True，signature的 name存在且signature的version在

signature.minimum 和signature.maximum 之间

回到主进程

init\_tasks()

def init\_tasks():  
 *"""Check tasks and reschedule uncompleted ones."""* db = Database()  
  
 log.debug("Checking for locked tasks..")  
 for task in db.list\_tasks(status=TASK\_RUNNING):  
 if config("cuckoo:cuckoo:reschedule"):  
 task\_id = db.reschedule(task.id)  
 log.info(  
 "Rescheduled task with ID %s and target %s: task #%s",  
 task.id, task.target, task\_id  
 )  
 else:  
 db.set\_status(task.id, TASK\_FAILED\_ANALYSIS)  
 log.info(  
 "Updated running task ID %s status to failed\_analysis",  
 task.id  
 )  
  
 log.debug("Checking for pending service tasks..")  
 for task in db.list\_tasks(status=TASK\_PENDING, category="service"):  
 db.set\_status(task.id, TASK\_FAILED\_ANALYSIS)

这部分读取cuckoo:cuckoo:reschedule的配置，通过数据库获取running（这是关了cuckoo再启动时的操作，一般来说不会存在这个状态）的task\_id，如果cuckoo:cuckoo:reschedule是yes，就会将用新的task\_id 覆盖旧id，并将其状态设为"recovered"。

然后将原来为"pending"的task设为failed。（记住，这是关了之后再重启时的pending）

init\_yara()  
init\_binaries()  
init\_rooter()  
init\_routing()

以上函数都是与插件和路由相关的，暂且不表

signatures = 0  
for sig in cuckoo.signatures:  
 if not sig.enabled:  
 continue  
 signatures += 1

确定signatures数量

回到cuckoo\_main

try:  
 cuckoo\_init(level, ctx)  
 cuckoo\_main(maxcount)

def cuckoo\_main(max\_analysis\_count=0):  
 *"""Cuckoo main loop.  
 @param max\_analysis\_count: kill cuckoo after this number of analyses  
 """* try:  
 ResultServer()  
 sched = Scheduler(max\_analysis\_count)  
 sched.start()  
 except KeyboardInterrupt:  
 sched.stop()

这部分主要有ResultServer()和Scheduler()

ResultServer()

class ResultServer(SocketServer.ThreadingTCPServer, object):  
 *"""Result server. Singleton!  
  
 This class handles results coming back from the analysis machines.  
 """* \_\_metaclass\_\_ = Singleton

class Singleton(type):  
 *"""Singleton.  
 @see: http://stackoverflow.com/questions/6760685/creating-a-singleton-in-python  
 """* \_instances = {}  
  
 def \_\_call\_\_(cls, \*args, \*\*kwargs):  
 if cls not in cls.\_instances:  
 cls.\_instances[cls] = super(Singleton, cls).\_\_call\_\_(\*args, \*\*kwargs)  
 return cls.\_instances[cls]

首先，他是个单例

self.servethread = threading.Thread(target=self.serve\_forever)  
self.servethread.setDaemon(True)  
self.servethread.start()  
break

主要是根据以下ip，端口来

self.ip = config("cuckoo:resultserver:ip")  
self.port = config("cuckoo:resultserver:port")

创建一个HTTPServer来显示分析结果

返回cuckoo\_main

try:  
 ResultServer()  
 sched = Scheduler(max\_analysis\_count)  
 sched.start()

Scheduler()就是task的调度器，直接跳到了start() 方法

def start(self):  
 *"""Start scheduler."""* self.initialize()

先执行initialize()

def initialize(self):  
 *"""Initialize the machine manager."""* global machinery, machine\_lock  
  
 machinery\_name = self.cfg.cuckoo.machinery  
  
 max\_vmstartup\_count = self.cfg.cuckoo.max\_vmstartup\_count  
 if max\_vmstartup\_count:  
 machine\_lock = threading.Semaphore(max\_vmstartup\_count)  
 else:  
 machine\_lock = threading.Lock()  
  
 log.info("Using \"%s\" as machine manager", machinery\_name, extra={  
 "action": "init.machinery",  
 "status": "success",  
 "machinery": machinery\_name,  
 })

如果在传入参数时设置了max\_vmstartup\_count，就会给予相应的信号量，如果有5个线程，max\_vmstartup\_count为4的话，第五个就会等待至有线程空位时才会触发。Python的多线程机制并非真正的多线程，而是不断在线程之间切换。但是，目前我们采用的是supervisord运行，是将每个任务当作进程来完成的。

machinery = cuckoo.machinery.plugins[machinery\_name]()  
  
# Provide a dictionary with the configuration options to the  
# machine manager instance.  
machinery.set\_options(Config(machinery\_name))  
  
# Initialize the machine manager.  
try:  
 machinery.initialize(machinery\_name)

具体根据config所选的引擎（Virtualbox）对machinery 进行initialize(machinery\_name)操作

据此跳到继承了Machinery的Virtualbox类下

class VirtualBox(Machinery):  
 *"""Virtualization layer for VirtualBox."""*

class Machinery(object):  
 *"""Base abstract class for machinery modules."""*

下的initialize方法

def initialize(self, module\_name):  
 *"""Read, load, and verify machines configuration.  
 @param module\_name: module name.  
 """* # Load.  
 self.\_initialize(module\_name)  
  
 # Run initialization checks.  
 self.\_initialize\_check()

执行了两个方法

def \_initialize(self, module\_name):  
 *"""Read configuration.  
 @param module\_name: module name.  
 """* machinery = self.options.get(module\_name)  
 for vmname in machinery["machines"]:  
 options = self.options.get(vmname)  
  
 # If configured, use specific network interface for this  
 # machine, else use the default value.  
 if options.get("interface"):  
 interface = options["interface"]  
 else:  
 interface = machinery.get("interface")  
  
 if options.get("resultserver\_ip"):  
 ip = options["resultserver\_ip"]  
 else:  
 ip = config("cuckoo:resultserver:ip")  
  
 if options.get("resultserver\_port"):  
 port = options["resultserver\_port"]  
 else:  
 # The ResultServer port might have been dynamically changed,  
 # get it from the ResultServer singleton. Also avoid import  
 # recursion issues by importing ResultServer here.  
 from cuckoo.core.resultserver import ResultServer  
 port = ResultServer().port

这部分主要是确定每个虚拟机报告resultserver的ip以及虚拟机本机的Ip

self.db.add\_machine(  
 name=vmname,  
 label=options[self.LABEL],  
 ip=options.ip,  
 platform=options.platform,  
 options=options.get("options", ""),  
 tags=options.tags,  
 interface=interface,  
 snapshot=options.snapshot,  
 resultserver\_ip=ip,  
 resultserver\_port=port  
 )

看Database的add\_machine方法

@classlock  
def add\_machine(self, name, label, ip, platform, options, tags, interface,  
 snapshot, resultserver\_ip, resultserver\_port):  
 *"""Add a guest machine.  
 @param name: machine id  
 @param label: machine label  
 @param ip: machine IP address  
 @param platform: machine supported platform  
 @param tags: list of comma separated tags  
 @param interface: sniffing interface for this machine  
 @param snapshot: snapshot name to use instead of the current one, if configured  
 @param resultserver\_ip: IP address of the Result Server  
 @param resultserver\_port: port of the Result Server  
 """* if options is None:  
 options = []  
 if not isinstance(options, (tuple, list)):  
 options = options.split()  
  
 session = self.Session()  
 machine = Machine(name=name,  
 label=label,  
 ip=ip,  
 platform=platform,  
 options=options,  
 interface=interface,  
 snapshot=snapshot,  
 resultserver\_ip=resultserver\_ip,  
 resultserver\_port=resultserver\_port)  
  
 # Deal with tags format (i.e., foo,bar,baz)  
 if tags:  
 for tag in tags.split(","):  
 if tag.strip():  
 tag = self.\_get\_or\_create(session, Tag, name=tag.strip())  
 machine.tags.append(tag)  
 session.add(machine)  
  
 try:  
 session.commit()  
 except SQLAlchemyError as e:  
 log.debug("Database error adding machine: {0}".format(e))  
 session.rollback()  
 finally:  
 session.close()

class Machine(Base):  
 *"""Configured virtual machines to be used as guests."""* \_\_tablename\_\_ = "machines"  
  
 id = Column(Integer(), primary\_key=True)  
 name = Column(String(255), nullable=False)  
 label = Column(String(255), nullable=False)  
 ip = Column(String(255), nullable=False)  
 platform = Column(String(255), nullable=False)  
 tags = relationship("Tag", secondary=machines\_tags, single\_parent=True,  
 backref="machine")  
 options = Column(JsonTypeList255(), nullable=True)  
 interface = Column(String(255), nullable=True)  
 snapshot = Column(String(255), nullable=True)  
 locked = Column(Boolean(), nullable=False, default=False)  
 locked\_changed\_on = Column(DateTime(timezone=False), nullable=True)  
 status = Column(String(255), nullable=True)  
 status\_changed\_on = Column(DateTime(timezone=False), nullable=True)  
 resultserver\_ip = Column(String(255), nullable=False)  
 resultserver\_port = Column(Integer(), nullable=False)

def \_\_init\_\_(self, name, label, ip, platform, options, interface,  
 snapshot, resultserver\_ip, resultserver\_port):  
 self.name = name  
 self.label = label  
 self.ip = ip  
 self.platform = platform  
 self.options = options  
 self.interface = interface  
 self.snapshot = snapshot  
 self.resultserver\_ip = resultserver\_ip  
 self.resultserver\_port = resultserver\_port

再看看Machine类，主要负责将每个虚拟机都添加到数据库中，并与数据库中的数据建立ORM映射联系

返回initialize的

self.\_initialize(module\_name)  
  
# Run initialization checks.  
self.\_initialize\_check()

下一个方法\_initialize\_check()

def \_initialize\_check(self):  
 *"""Runs all checks when a machine manager is initialized.  
 @raise CuckooMachineError: if VBoxManage is not found.  
 """* if not self.options.virtualbox.path:  
 raise CuckooCriticalError(  
 "VirtualBox VBoxManage path is missing, please add it to the "  
 "virtualbox.conf configuration file!"  
 )  
  
 if not os.path.exists(self.options.virtualbox.path):  
 raise CuckooCriticalError(  
 "VirtualBox' VBoxManage not found at specified path \"%s\" "  
 "(as specified in virtualbox.conf). Did you properly install "  
 "VirtualBox and configure Cuckoo to use it?"  
 % self.options.virtualbox.path  
 )  
  
 if self.options.virtualbox.mode not in ("gui", "headless"):  
 raise CuckooCriticalError(  
 "VirtualBox has been configured to run in a non-supported "  
 "mode: %s. Please upgrade your configuration to reflect "  
 "either 'gui' or 'headless' mode!" %  
 self.options.virtualbox.mode  
 )

先确认配置存在

然后执行父类的initalize\_check()

super(VirtualBox, self).\_initialize\_check()

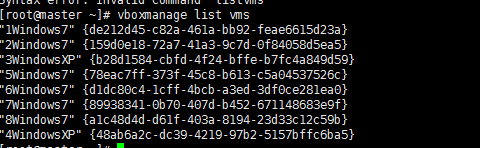
def \_initialize\_check(self):  
 *"""Runs checks against virtualization software when a machine manager  
 is initialized.  
 @note: in machine manager modules you may override or superclass  
 his method.  
 @raise CuckooMachineError: if a misconfiguration or a unkown vm state  
 is found.  
 """* try:  
 configured\_vms = self.\_list()  
 except NotImplementedError:  
 return  
  
 for machine in self.machines():  
 # If this machine is already in the "correct" state, then we  
 # go on to the next machine.  
 if machine.label in configured\_vms and \  
 self.\_status(machine.label) in [self.POWEROFF, self.ABORTED]:  
 continue  
  
 # This machine is currently not in its correct state, we're going  
 # to try to shut it down. If that works, then the machine is fine.  
 try:  
 self.stop(machine.label)

找出可配置的虚拟机，然后对它进行stop方法

machines = self.\_list()  
for machine in self.machines():  
 if machine.label not in machines:  
 continue  
  
 self.restore(machine.label, machine)

def \_list(self):  
 *"""Lists virtual machines installed.  
 @return: virtual machine names list.  
 """* try:  
 args = [  
 self.options.virtualbox.path, "list", "vms"  
 ]  
 output, \_ = Popen(  
 args, stdout=subprocess.PIPE, stderr=subprocess.PIPE,  
 close\_fds=True  
 ).communicate()  
 except OSError as e:  
 raise CuckooMachineError(  
 "VBoxManage error listing installed machines: %s" % e  
 )

通过命令VBoxManage list vms



获取可用的虚拟机，放进列表并返回

machines = []  
 for line in output.split("\n"):  
 if '"' not in line:  
 continue  
  
 label = line.split('"')[1]  
 if label == "<inaccessible>":  
 log.warning(  
 "Found an inaccessible virtual machine, please check "  
 "its state."  
 )  
 continue  
  
 machines.append(label)  
 return machines

def stop(self, label):  
 *"""Stops a virtual machine.  
 @param label: virtual machine name.  
 @raise CuckooMachineError: if unable to stop.  
 """* log.debug("Stopping vm %s" % label)  
  
 status = self.\_status(label)  
  
 # The VM has already been restored, don't shut it down again. This  
 # appears to be a VirtualBox-specific state though, hence we handle  
 # it here rather than in Machinery.\_initialize\_check().  
 if status == self.SAVED:  
 return  
  
 if status == self.POWEROFF or status == self.ABORTED:  
 raise CuckooMachineError(  
 "Trying to stop an already stopped VM: %s" % label  
 )  
  
 vm\_state\_timeout = config("cuckoo:timeouts:vm\_state")  
  
 try:  
 args = [  
 self.options.virtualbox.path, "controlvm", label, "poweroff"  
 ]  
 proc = Popen(  
 args, stdout=subprocess.PIPE, stderr=subprocess.PIPE,  
 close\_fds=True  
 )  
  
 # Sometimes VBoxManage stucks when stopping vm so we needed  
 # to add a timeout and kill it after that.  
 stop\_me = 0  
 while proc.poll() is None:  
 if stop\_me < vm\_state\_timeout:  
 time.sleep(1)  
 stop\_me += 1  
 else:  
 log.debug("Stopping vm %s timeouted. Killing" % label)  
 proc.terminate()  
  
 if proc.returncode != 0 and stop\_me < vm\_state\_timeout:  
 log.debug(  
 "VBoxManage exited with error powering off the machine"  
 )  
 except OSError as e:  
 raise CuckooMachineError(  
 "VBoxManage failed powering off the machine: %s" % e  
 )  
  
 self.\_wait\_status(label, self.POWEROFF, self.ABORTED, self.SAVED)

调整状态至关闭 VBoxManage controlvm 虚拟机名称 poweroff

返回\_initialize\_check()

def \_initialize\_check(self):

......

super(VirtualBox, self).\_initialize\_check()

machines = self.\_list()  
 for machine in self.machines():  
 if machine.label not in machines:  
 continue  
  
 self.restore(machine.label, machine)

对所有machines进行restore()方法

def restore(self, label, machine):  
 *"""Restore a VM to its snapshot."""* args = [  
 self.options.virtualbox.path, "snapshot", label  
 ]  
  
 if machine.snapshot:  
 log.debug(  
 "Restoring virtual machine %s to %s",  
 label, machine.snapshot  
 )  
 args.extend(["restore", machine.snapshot])  
 else:  
 log.debug(  
 "Restoring virtual machine %s to its current snapshot",  
 label  
 )  
 args.append("restorecurrent")

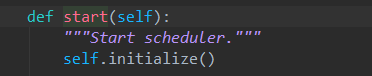
对所有机器获取快照snapshot，如果有指定就使用那个快照，如果没有就采用最近的快照

def restore(self, label, machine):  
 *......*

try:  
 p = Popen(  
 args, stdout=subprocess.PIPE, stderr=subprocess.PIPE,  
 close\_fds=True  
 )  
 \_, err = p.communicate()  
 if p.returncode:  
 raise OSError("error code %d: %s" % (p.returncode, err))  
 except OSError as e:  
 raise CuckooMachineSnapshotError(  
 "VBoxManage failed trying to restore the snapshot of "  
 "machine '%s' (this most likely means there is no snapshot, "  
 "please refer to our documentation for more information on "  
 "how to setup a snapshot for your VM): %s" % (label, e)  
 )

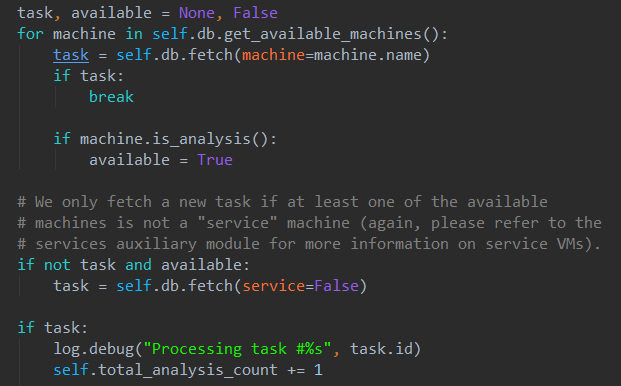
vboxmanage snapshot 虚拟机名字 restore 快照名字 | restorecurrent，采用这条命令将虚拟机回复到之前的状态

initialize()完毕后回到

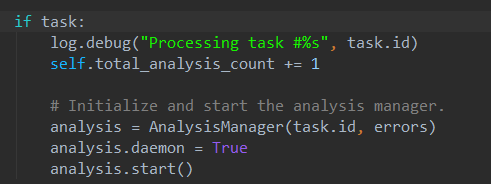




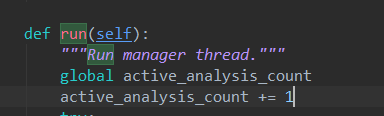
这里创建了一个死循环，等待任务进入



从数据库中获取可用的虚拟机，如果有，再获取要处理的task

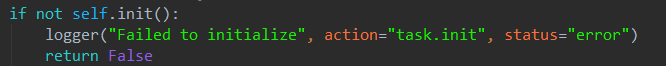


将需要处理的task 放进AnalysisManager里运行



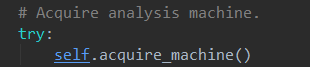


真正运行是这个函数

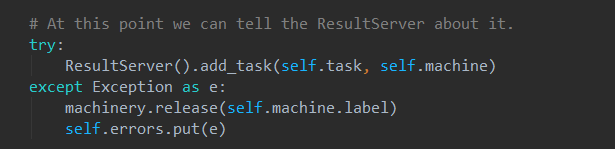


首先执行init()，这里执行了一些任务，比如创建相应的文件夹，对比文件提交前后的sha256值，复制文件等等。

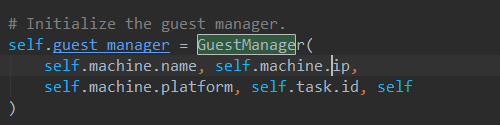
然后执行



往数据库中写入虚拟机的状态为locked，将self.machine设为空闲的machine.

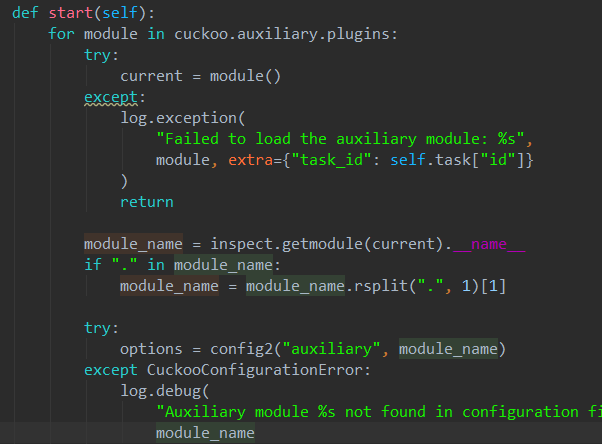


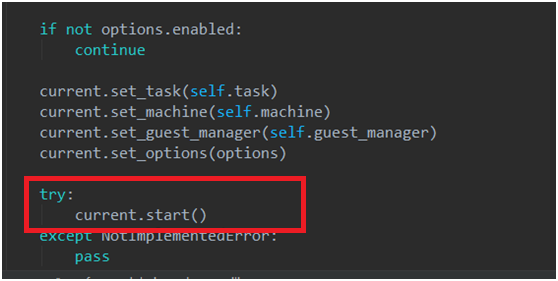
在Resultserver上注册task和对应的machine



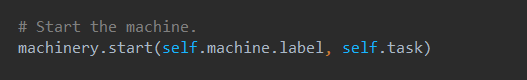
初始化GuestManager，以便控制guest（虚拟机）

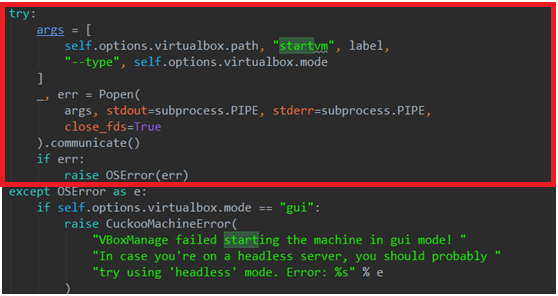






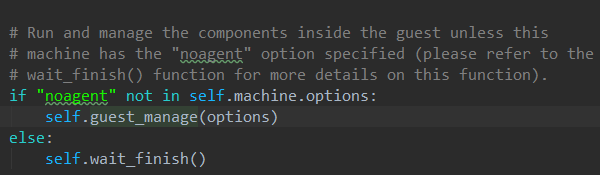
对上面那个长长的pluginlist运行其run（start()）方法，由于这些都是auxilary模块，先不表。



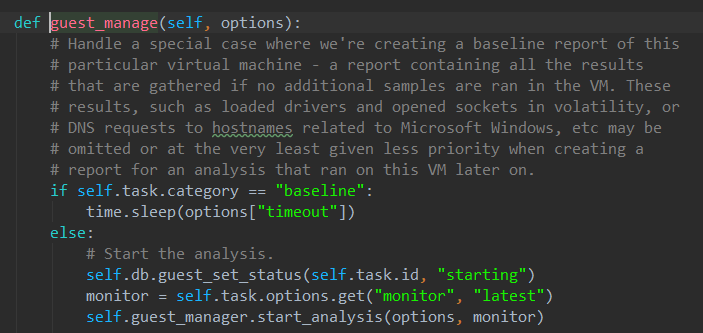


利用VBoxManage startvm 虚拟机名称 --type 启动方式（有界面启动或无界面启动）

开启虚拟机后，



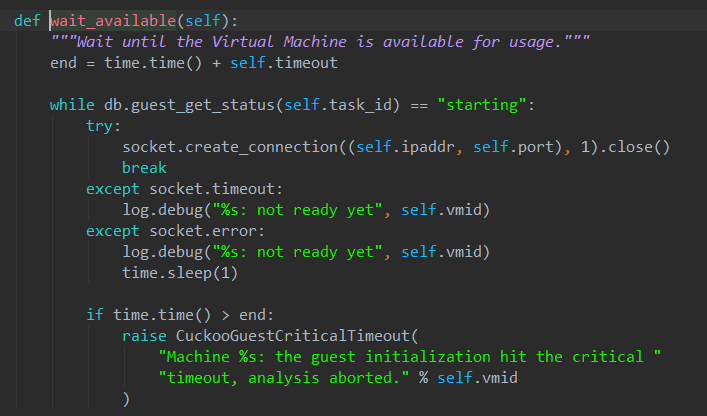
把option传进guest\_manage()运行



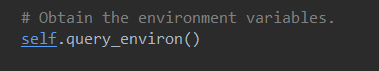
一般来说，会进入else那里，运行GuestManager的start\_analysis()方法

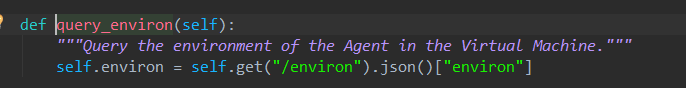


由于在虚拟机了安装了agent.py，cuckoo依靠两者socket通讯来传递信息，这个函数就是等待agent.py服务器可用，然后建立连接



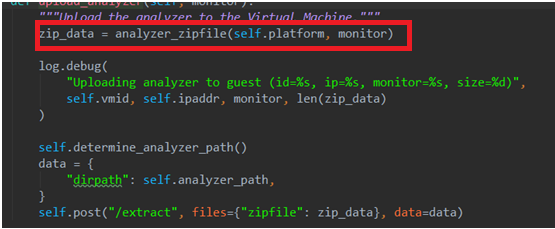
随后如果不退出的话，

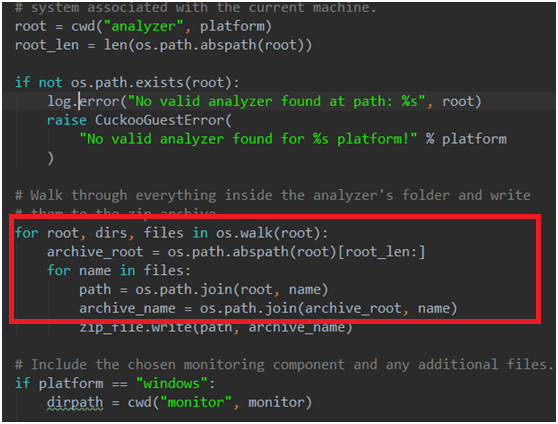




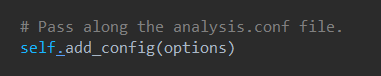
通过/environ请求获得环境变量

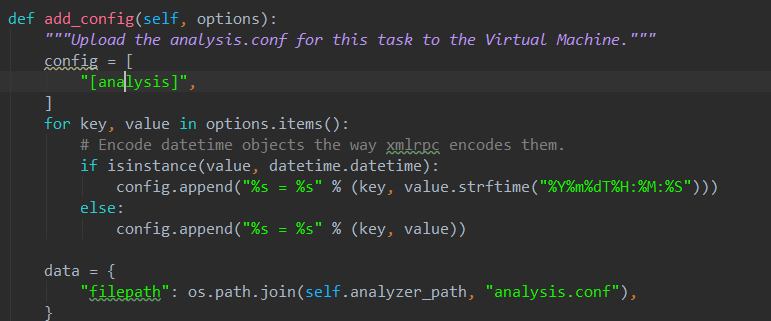




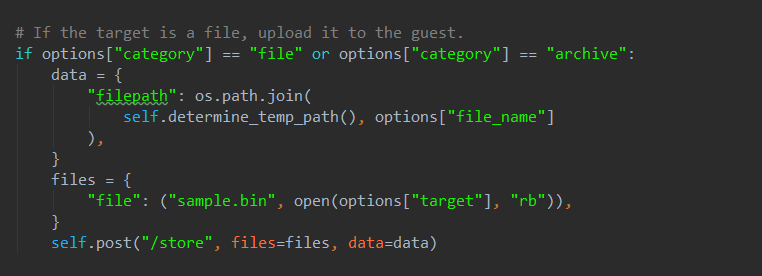


获得环境变量的系统后，通过post方法，将对应的analyzer（windows）打包，上传到/extract去。analyzer的目录是/data/analyzer

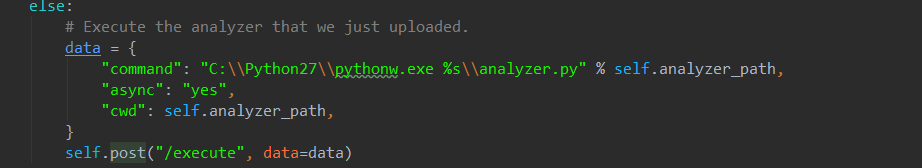




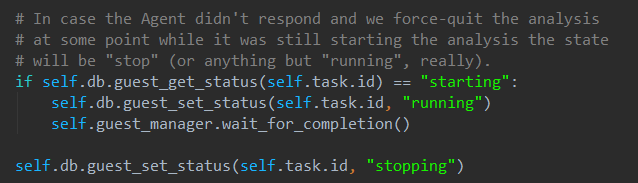
把配置文件上传到虚拟机



把要分析的文件上传虚拟机



执行刚刚上传的analyzer.py



将状态从starting变成running然后

def wait\_for\_completion(self):  
 if self.is\_old:  
 self.old.wait\_for\_completion()  
 return  
  
 end = time.time() + self.timeout  
  
 while db.guest\_get\_status(self.task\_id) == "running":  
 log.debug("%s: analysis still processing", self.vmid)  
  
 time.sleep(1)  
  
 # If the analysis hits the critical timeout, just return straight  
 # away and try to recover the analysis results from the guest.  
 if time.time() > end:  
 log.info("%s: end of analysis reached!", self.vmid)  
 return  
  
 try:  
 status = self.get("/status", timeout=5).json()  
 except Exception as e:  
 log.info("Virtual Machine /status failed (%r)", e)  
 # this might fail due to timeouts or just temporary network issues  
 # thus we don't want to abort the analysis just yet and wait for things to  
 # recover  
 continue  
  
 if status["status"] == "complete":  
 log.info("%s: analysis completed successfully", self.vmid)  
 return  
 elif status["status"] == "exception":  
 log.warning(  
 "%s: analysis caught an exception\n%s",  
 self.vmid, status["description"]  
 )  
 return

等待执行完毕，注意这里是agent.py执行写的操作，AnalysisManager执行读的操作，读到状态不为是"complete"或"exception"就跳出循环

之后的代码就是退出和执行清理工作了

def run(self):  
 *"""Run manager thread."""* global active\_analysis\_count  
 active\_analysis\_count += 1  
 try:  
 self.launch\_analysis()  
  
 self.db.set\_status(self.task.id, TASK\_COMPLETED)

回到run()可以发现，如果成功结束的话，会执行store\_task\_info()和process\_results()

def store\_task\_info(self):  
 *"""grab latest task from db (if available) and update self.task"""* dbtask = self.db.view\_task(self.task.id)  
 self.task = dbtask.to\_dict()  
  
 task\_info\_path = os.path.join(self.storage, "task.json")  
 open(task\_info\_path, "w").write(dbtask.to\_json())

第一个函数主要把任务结果写进task.json

def process\_results(self):  
 *"""Process the analysis results and generate the enabled reports."""* logger(  
 "Starting task reporting",  
 action="task.report", status="pending"  
 )  
  
 # *TODO Refactor this function as currently "cuckoo process" has a 1:1* # copy of its code. *TODO Also remove "archive" files.* results = RunProcessing(task=self.task).run()  
 RunSignatures(results=results).run()  
 RunReporting(task=self.task, results=results).run()

第二个函数主要进行了RunProcessing, RunSIgnatures, RunReporting 的run方法

class RunProcessing(object):

def run(self):  
 *"""Run all processing modules and all signatures.  
 @return: processing results.  
 """* # This is the results container. It's what will be used by all the  
 # reporting modules to make it consumable by humans and machines.  
 # It will contain all the results generated by every processing  
 # module available. Its structure can be observed through the JSON  
 # dump in the analysis' reports folder. (If jsondump is enabled.)  
 # We friendly call this "fat dict".  
 results = {  
 "\_temp": {},  
 }  
  
 # Uses plain machine configuration as input.  
 self.populate\_machine\_info()  
  
 # Order modules using the user-defined sequence number.  
 # If none is specified for the modules, they are selected in  
 # alphabetical order.  
 processing\_list = cuckoo.processing.plugins  
  
 # If no modules are loaded, return an empty dictionary.  
 if processing\_list:  
 processing\_list.sort(key=lambda module: module.order)  
  
 # Run every loaded processing module.  
 for module in processing\_list:  
 key, result = self.process(module, results)  
  
 # If the module provided results, append it to the fat dict.  
 if key and result:  
 results[key] = result  
 else:  
 log.info("No processing modules loaded")  
  
 results.pop("\_temp", None)  
  
 # Return the fat dict.  
 return results

class RunSignatures(object):  
 *"""Run Signatures."""*

...的

def run(self):  
 *"""Run signatures."""*

*......*

# Allow signatures to initialize themselves.  
 score, configuration = 0, []  
 for signature in self.signatures:  
 if signature.matched:  
 log.debug(  
 "Analysis matched signature: %s", signature.name, extra={  
 "action": "signature.match", "status": "success",  
 "signature": signature.name,  
 "severity": signature.severity,  
 }  
 )  
 self.matched.append(signature.results())  
 score += signature.severity  
  
 for mark in signature.marks:  
 if mark["type"] == "config":  
 configuration.append(mark["config"])  
  
 # Sort the matched signatures by their severity level and put them  
 # into the results dictionary.  
 self.matched.sort(key=lambda key: key["severity"])  
 self.results["signatures"] = self.matched  
 if "info" in self.results:  
 self.results["info"]["score"] = score / 5.0  
  
 # If malware configuration has been extracted, simplify its  
 # accessibility in the analysis report.  
 if configuration:  
 # *TODO Should this be included elsewhere?* if "metadata" in self.results:  
 self.results["metadata"]["cfgextr"] = configuration  
 if "info" in self.results:  
 self.results["info"]["score"] = 10  
 if self.results["info"]["score"] > 10:  
 self.results["info"]["score"] = 10

这一块就是最后的得分，就是上次报告呈现的部分

class RunReporting(object):

def run(self):  
 *"""Generates all reports.  
 @raise CuckooReportError: if a report module fails.  
 """* # In every reporting module you can specify a numeric value that  
 # represents at which position that module should be executed among  
 # all the available ones. It can be used in the case where a  
 # module requires another one to be already executed beforehand.  
 reporting\_list = cuckoo.reporting.plugins  
  
 # Return if no reporting modules are loaded.  
 if reporting\_list:  
 reporting\_list.sort(key=lambda module: module.order)  
  
 # Run every loaded reporting module.  
 for module in reporting\_list:  
 self.process(module)  
 else:  
 log.info("No reporting modules loaded")

这是把报告结果报告给其他平台的地方

第一和第三个方法都是基于各个Signature和Reporting模块来定义的，也就是说这是一种多态的编程手段。

余下的代码是执行一些清理工作。到此，沙箱处理文件的流程就已经完毕。

关于调度的问题。沙箱源码中，默认每隔一秒去查可用的沙箱，可用的标志是数据库中该沙箱的状态值是否locked，然后从数据库中去查pending的任务，找到之后就放到可用的沙箱里去检测。

我之前设计的调度也是这样的，只不过在外层再加了一层队列来处理提交，将提交分成提交到队列和真实提交两个过程。事实证明，源码也是这样操作。此意见我大概在2017年11月就已提出，然而，直到最近才被开发出来。

这样做的合理性，就是提高了沙箱的利用率。现行生产上出现分数不一致的问题，也是因为如此，个人认为应该可以通过配置默认沙箱系统、摘要算法识别同一文件的做法（源码也是这样做的）来确保同一个文件不会被提交两次，从而产生的分数统一。

如果用户实在有需要，可以将文件手动提交到其他系统的沙箱，或者多勾选沙箱，来进行其他系统上的跑分。这时就必须要接受跑得慢的事实。沙箱是通过虚拟机跑起来的，每跑一个虚拟机，就要在系统里面开一个进程（直接cuckoomain用的还是恶名昭著的Python线程）运行一个操作系统，监听文件在里面运行的行为，开销其实是很大的。

用户必须接受的一个事实，就是不同系统上跑的结果不同。因为沙箱是一种动态分析工具，所谓的动态，就是要实时抓取文件运行的情况。为了提高速度，可以将第一次运行的结果保存，以后遇到同一文件都采用这个结果，但是如果不同系统上跑的结果是一样的，那么分系统的做法也就毫无意义了。

关于多沙箱并行跑分的做法，个人认为在目前的机器配置下并不现实。多沙箱设计的本意其实是为了尽快跑完多个文件，而不是对它进行同步多操作系统运行，这样得出的行为结果也难以整合。

反正这是一个两难的问题：牺牲速度来换多样性（还不是准确性而只是多样性），还是牺牲多样性来换取速度。当然，升级机器配置可以解决大部分问题，但这样做成本就增加了。

Part 2 Cuckoo 虚拟机运行过程解构

沙箱运行前，必须把虚拟机回复到一个之前的状态，而这个状态有一个条件，就是agent.py要开始运行。由前文可知，agent.py是一个TCP服务器。从入口开始看起。

if \_\_name\_\_ == "\_\_main\_\_":  
 parser = argparse.ArgumentParser()  
 parser.add\_argument("host", nargs="?", default="0.0.0.0")  
 parser.add\_argument("port", nargs="?", default="8000")  
 args = parser.parse\_args()  
  
 app.run(host=args.host, port=int(args.port))

运行了app.run方法

app = MiniHTTPServer()  
state = {}

class MiniHTTPServer(object):  
 def \_\_init\_\_(self):  
 self.handler = MiniHTTPRequestHandler  
  
 # Reference back to the server.  
 self.handler.httpd = self  
  
 self.routes = {  
 "GET": [],  
 "POST": [],  
 }  
  
 def run(self, host="0.0.0.0", port=8000):  
 self.s = SocketServer.TCPServer((host, port), self.handler)  
 self.s.allow\_reuse\_address = True  
 self.s.serve\_forever()

run方法很简单，就是创建了一个TCP服务器，然后让它不断地运行下去

else:  
 # Execute the analyzer that we just uploaded.  
 data = {  
 "command": "C:\\Python27\\pythonw.exe %s\\analyzer.py" % self.analyzer\_path,  
 "async": "yes",  
 "cwd": self.analyzer\_path,  
 }  
 self.post("/execute", data=data)

回到上面调用analyzer.py的进程代码中，可发现宿主机（客户端）向虚拟机（服务器）发送了一个post请求"/execute"将analyzer.py的以表单的形式上传，此时再看看agent.py的回应：

@app.route("/execute", methods=["POST"])  
def do\_execute():  
 if "command" not in request.form:  
 return json\_error(400, "No command has been provided")  
  
 # Execute the command asynchronously? As a shell command?  
 async = "async" in request.form  
 shell = "shell" in request.form  
  
 cwd = request.form.get("cwd")  
 stdout = stderr = None  
  
 try:  
 if async:  
 subprocess.Popen(request.form["command"], shell=shell, cwd=cwd)  
 else:  
 p = subprocess.Popen(  
 request.form["command"], shell=shell, cwd=cwd,  
 stdout=subprocess.PIPE, stderr=subprocess.PIPE  
 )  
 stdout, stderr = p.communicate()  
 except:  
 return json\_exception("Error executing command")  
  
 return json\_success("Successfully executed command",  
 stdout=stdout, stderr=stderr)

先看看函数头的装饰器app.route

def route(self, path, methods=["GET"]):  
 def register(fn):  
 for method in methods:  
 self.routes[method].append((re.compile(path + "$"), fn))  
 return fn  
  
 return register

负责将self.route字典的"GET"或"POST"键列表中放入一个正则表达式对象

然后看函数主体

cwd 是在命令行中执行的当前目录

request.form["command"]就是命令，通常是C:\Python27\pythonw.exe 平台所属analyer路径\analyzer.py，在这个情况下是\data\analyzer\windows\analyzer.py

就是用python去执行它，所以跳到analyzer.py的入口。

if \_\_name\_\_ == "\_\_main\_\_":  
 success = False  
 error = ""  
  
 try:  
 # Initialize the main analyzer class.  
 analyzer = Analyzer()  
  
 # Run it and wait for the response.  
 success = analyzer.run()  
  
 data = {  
 "status": "complete",  
 "description": success,  
 }  
 # This is not likely to happen.  
 except KeyboardInterrupt:  
 error = "Keyboard Interrupt"

执行了analyzer.run()方法

def run(self):  
 *"""Run analysis.  
 @return: operation status.  
 """* self.prepare()   
 self.path = os.getcwd()

首先执行prepare()方法

def prepare(self):  
 *"""Prepare env for analysis."""* # Get SeDebugPrivilege for the Python process. It will be needed in  
 # order to perform the injections.  
 grant\_privilege("SeDebugPrivilege")  
 grant\_privilege("SeLoadDriverPrivilege")

def grant\_privilege(privilege):

...  
 *"""Grant debug privileges.  
 @param pid: PID.  
 @return: operation status.  
 """* ADVAPI32.OpenProcessToken.argtypes = (wintypes.HANDLE,  
 wintypes.DWORD,  
 POINTER(wintypes.HANDLE))  
  
 ADVAPI32.LookupPrivilegeValueW.argtypes = (wintypes.LPWSTR,  
 wintypes.LPWSTR,  
 POINTER(LUID))  
  
 ADVAPI32.AdjustTokenPrivileges.argtypes = (wintypes.HANDLE,  
 wintypes.BOOL,  
 POINTER(TOKEN\_PRIVILEGES),  
 wintypes.DWORD,  
 POINTER(TOKEN\_PRIVILEGES),  
 POINTER(wintypes.DWORD))  
  
 h\_process = KERNEL32.GetCurrentProcess()  
  
 h\_current\_token = wintypes.HANDLE()  
 if not ADVAPI32.OpenProcessToken(h\_process,  
 TOKEN\_ALL\_ACCESS,  
 h\_current\_token):  
 return False  
  
 se\_original\_luid = LUID()  
 if not ADVAPI32.LookupPrivilegeValueW(None, privilege, se\_original\_luid):  
 return False  
  
 luid\_attributes = LUID\_AND\_ATTRIBUTES()  
 luid\_attributes.Luid = se\_original\_luid  
 luid\_attributes.Attributes = SE\_PRIVILEGE\_ENABLED  
 token\_privs = TOKEN\_PRIVILEGES()  
 token\_privs.PrivilegeCount = 1  
 token\_privs.Privileges = luid\_attributes  
  
 if not ADVAPI32.AdjustTokenPrivileges(h\_current\_token, False, token\_privs,  
 0, None, None):  
 return False  
  
 KERNEL32.CloseHandle(h\_current\_token)  
 KERNEL32.CloseHandle(h\_process)  
 return True

这个grant\_privilege()比较纠结……就是调用了windows的一些API

KERNEL32 = windll.kernel32  
ADVAPI32 = windll.advapi32

def grant\_privilege(privilege):

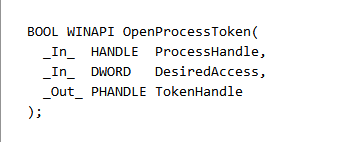
...

h\_process = KERNEL32.GetCurrentProcess()  
  
 h\_current\_token = wintypes.HANDLE()  
 if not ADVAPI32.OpenProcessToken(h\_process,  
 TOKEN\_ALL\_ACCESS,  
 h\_current\_token):  
 return False

第一句使用GetCurrentProcess()函数获得当前进程（analyzer.py）

这里进行了一个提权操作，将当前进程的ProcessToken的所有权限开放，包括

|  |  |
| --- | --- |
| **Value** | **Meaning** |
| TOKEN\_ADJUST\_DEFAULT | Required to change the default owner, primary group, or DACL of an access token. |
| TOKEN\_ADJUST\_GROUPS | Required to adjust the attributes of the groups in an access token. |
| TOKEN\_ADJUST\_PRIVILEGES | Required to enable or disable the privileges in an access token. |
| TOKEN\_ADJUST\_SESSIONID | Required to adjust the session ID of an access token. The SE\_TCB\_NAME privilege is required. |
| TOKEN\_ASSIGN\_PRIMARY | Required to attach a [*primary token*](https://msdn.microsoft.com/en-us/library/windows/desktop/ms721603(v=vs.85).aspx#_security_primary_token_gly) to a [*process*](https://msdn.microsoft.com/en-us/library/windows/desktop/ms721603(v=vs.85).aspx#_security_process_gly). The SE\_ASSIGNPRIMARYTOKEN\_NAME privilege is also required to accomplish this task. |
| TOKEN\_DUPLICATE | Required to duplicate an access token. |
| TOKEN\_EXECUTE | Combines STANDARD\_RIGHTS\_EXECUTE and TOKEN\_IMPERSONATE. |
| TOKEN\_IMPERSONATE | Required to attach an impersonation access token to a process. |
| TOKEN\_QUERY | Required to query an access token. |
| TOKEN\_QUERY\_SOURCE | Required to query the source of an access token. |
| TOKEN\_READ | Combines STANDARD\_RIGHTS\_READ and TOKEN\_QUERY. |
| TOKEN\_WRITE | Combines STANDARD\_RIGHTS\_WRITE, TOKEN\_ADJUST\_PRIVILEGES, TOKEN\_ADJUST\_GROUPS, and TOKEN\_ADJUST\_DEFAULT. |
| TOKEN\_ALL\_ACCESS | Combines all possible access rights for a token. |



根据OpenProcessToken定义，传入的三个参数分别是要修改访问权限的进程句柄（无类型指针）、要进行的操作类型或者说要获得的访问权限（无符号长整形），访问令牌的指针（无类型指针的指针），这个在上文定义了。

ADVAPI32.OpenProcessToken.argtypes = (wintypes.HANDLE,  
 wintypes.DWORD,  
 POINTER(wintypes.HANDLE))

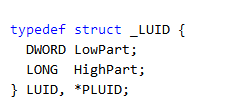
这个提权操作应该能让当前进程打开与进程相关联的访问令牌。如无法获得则返回。

def grant\_privilege(privilege):

...

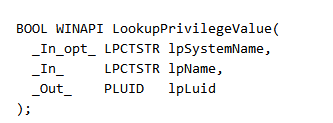
se\_original\_luid = LUID()  
 if not ADVAPI32.LookupPrivilegeValueW(None, privilege, se\_original\_luid):  
 return False

class LUID(Structure):  
 \_fields\_ = [  
 ("LowPart", DWORD),  
 ("HighPart", LONG),  
 ]



LUID局部唯一识别符：每个系统生成的，在系统中独一无二的标识符，其独特性保持到系统重启。

然后，通过定义并实例化一个LUID（由两个长整型组成的结构体），调用LookupPrivilegeValueW。



此函数第一个参数表示所要查看的系统，本地系统直接用NULL（Python用None表示空指针），第二个参数指向一个以零结尾的字符串，指定特权的名称（https://msdn.microsoft.com/en-us/library/windows/desktop/bb530716(v=vs.85).aspx 可以获得所有可能的项）。第三个参数用来接收所返回的制定特权名称的信息（LUID）。

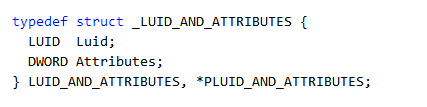
def grant\_privilege(privilege):

...

luid\_attributes = LUID\_AND\_ATTRIBUTES()   
 luid\_attributes.Luid = se\_original\_luid  
 luid\_attributes.Attributes = SE\_PRIVILEGE\_ENABLED  
 token\_privs = TOKEN\_PRIVILEGES()  
 token\_privs.PrivilegeCount = 1   
 token\_privs.Privileges = luid\_attributes

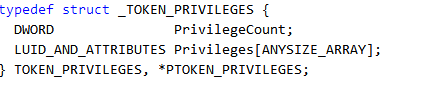
回到grant\_privilege。这里实例化了一个luid\_attributes， 生成结构体（LUID, 长整型）把之前的LUID、se\_privilege\_enabled（启用此特权）传进去。

class LUID\_AND\_ATTRIBUTES(Structure):  
 \_fields\_ = [  
 ("Luid", LUID),  
 ("Attributes", DWORD),  
 ]



SE\_PRIVILEGE\_ENABLED = 0x00000002

然后，



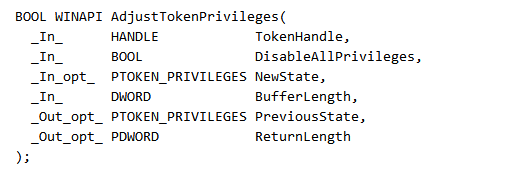
class TOKEN\_PRIVILEGES(Structure):  
 \_fields\_ = [  
 ("PrivilegeCount", DWORD),  
 ("Privileges", LUID\_AND\_ATTRIBUTES),  
 ]

实例化TOKEN\_PRIVILEGES，其属性PrivilegeCount = 1，将luid\_and\_attributes传进去

def grant\_privilege(privilege):

...  
 if not ADVAPI32.AdjustTokenPrivileges(h\_current\_token, False, token\_privs,  
 0, None, None):  
 return False

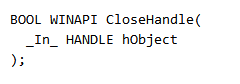
调用AdjustTokenPriviliges函数，将h\_current\_token所载的TokenHandle变成token\_privs



def grant\_privilege(privilege):

KERNEL32.CloseHandle(h\_current\_token)  
 KERNEL32.CloseHandle(h\_process)  
 return True

最后再关闭进程句和令牌句柄



整个过程就是

定义参数类型 -> 获得当前进程句柄 -> 找出参数privilege字符串对应的权限 -> 通过令牌句柄将权限写入进程句柄

至此完成了提权的全部操作。

回到prepare

def prepare(self):  
 *"""Prepare env for analysis."""* # Get SeDebugPrivilege for the Python process. It will be needed in  
 # order to perform the injections.  
 grant\_privilege("SeDebugPrivilege")  
 grant\_privilege("SeLoadDriverPrivilege")

使用grant\_privilege获得“修改进程拥有账户内存的权限”和“获取驱动器”的权限

def prepare(self):  
  
 init\_logging()  
  
 # Parse the analysis configuration file generated by the agent.  
 self.config = Config(cfg="analysis.conf")  
  
 # Pass the configuration through to the Process class.  
 Process.set\_config(self.config)  
  
 # Set virtual machine clock.  
 set\_clock(datetime.datetime.strptime(  
 self.config.clock, "%Y%m%dT%H:%M:%S"  
 ))  
  
 # Set the default DLL to be used for this analysis.  
 self.default\_dll = self.config.options.get("dll")  
  
 # If a pipe name has not set, then generate a random one.  
 self.config.pipe = self.get\_pipe\_path(  
 self.config.options.get("pipe", random\_string(16, 32))  
 )  
  
 # Generate a random name for the logging pipe server.  
 self.config.logpipe = self.get\_pipe\_path(random\_string(16, 32))  
  
 # Initialize and start the Command Handler pipe server. This is going  
 # to be used for communicating with the monitored processes.  
 self.command\_pipe = PipeServer(  
 PipeDispatcher, self.config.pipe, message=True,  
 dispatcher=CommandPipeHandler(self)  
 )  
 self.command\_pipe.daemon = True  
 self.command\_pipe.start()  
  
 # Initialize and start the Log Pipe Server - the log pipe server will  
 # open up a pipe that monitored processes will use to send logs to  
 # before they head off to the host machine.  
 destination = self.config.ip, self.config.port

然后如果有指定文件分析时的配置项，则根据它来设置通讯管道名称、使用的DLL和管道路径

self.log\_pipe\_server = PipeServer(  
 PipeForwarder, self.config.logpipe, destination=destination  
 )  
 self.log\_pipe\_server.daemon = True  
 self.log\_pipe\_server.start()  
  
 # We update the target according to its category. If it's a file, then  
 # we store the target path.  
 if self.config.category == "file":  
 self.target = os.path.join(  
 os.environ["TEMP"], self.config.file\_name  
 )  
 elif self.config.category == "archive":  
 zip\_path = os.path.join(os.environ["TEMP"], self.config.file\_name)  
 zipfile.ZipFile(zip\_path).extractall(os.environ["TEMP"])  
 self.target = os.path.join(  
 os.environ["TEMP"], self.config.options["filename"]  
 )  
 # If it's a URL, well.. we store the URL.  
 else:  
 self.target = self.config.target

创建一个PipeServer来专门处理命令。

看一下这个PipeServer

class PipeServer(threading.Thread):  
 *"""The Pipe Server accepts incoming pipe handlers and initializes  
 them in a new thread."""* def \_\_init\_\_(self, pipe\_handler, pipe\_name, message=False, \*\*kwargs):  
 threading.Thread.\_\_init\_\_(self)  
 self.pipe\_handler = pipe\_handler  
 self.pipe\_name = pipe\_name  
 self.message = message  
 self.kwargs = kwargs  
 self.do\_run = True  
  
 def run(self):  
 while self.do\_run:  
 flags = FILE\_FLAG\_WRITE\_THROUGH  
 if self.message:  
 pipe\_handle = KERNEL32.CreateNamedPipeA(  
 self.pipe\_name, PIPE\_ACCESS\_DUPLEX | flags,  
 PIPE\_TYPE\_MESSAGE | PIPE\_READMODE\_MESSAGE | PIPE\_WAIT,  
 PIPE\_UNLIMITED\_INSTANCES, BUFSIZE, BUFSIZE, 0, None  
 )  
 else:  
 pipe\_handle = KERNEL32.CreateNamedPipeA(  
 self.pipe\_name, PIPE\_ACCESS\_INBOUND | flags,  
 PIPE\_TYPE\_BYTE | PIPE\_READMODE\_BYTE | PIPE\_WAIT,  
 PIPE\_UNLIMITED\_INSTANCES, 0, BUFSIZE, 0, None  
 )  
  
 if pipe\_handle == INVALID\_HANDLE\_VALUE:  
 log.warning("Error opening logging pipe server.")  
 continue  
  
 if KERNEL32.ConnectNamedPipe(pipe\_handle, None) or \  
 KERNEL32.GetLastError() == ERROR\_PIPE\_CONNECTED:  
 handler = self.pipe\_handler(pipe\_handle, \*\*self.kwargs)  
 handler.daemon = True  
 handler.start()  
 else:  
 KERNEL32.CloseHandle(pipe\_handle)  
  
 def stop(self):  
 self.do\_run = False

PIPE\_ACCESS\_INBOUND = 0x00000001  
PIPE\_ACCESS\_DUPLEX = 0x00000003  
PIPE\_TYPE\_MESSAGE = 0x00000004  
PIPE\_READMODE\_MESSAGE = 0x00000002  
PIPE\_WAIT = 0x00000000  
PIPE\_UNLIMITED\_INSTANCES = 0x000000ff  
PIPE\_TYPE\_BYTE = 0x00000000  
PIPE\_READMODE\_BYTE = 0x00000000  
FILE\_FLAG\_WRITE\_THROUGH = 0x80000000  
INVALID\_HANDLE\_VALUE = 0xffffffff  
ERROR\_BROKEN\_PIPE = 0x0000006d  
ERROR\_MORE\_DATA = 0x000000EA  
ERROR\_PIPE\_CONNECTED = 0x00000217

定义了与Windows PIPE相关的一些变量

|  |  |
| --- | --- |
| **PIPE\_ACCESS\_DUPLEX**  0x00000003 | The pipe is bi-directional; both server and client processes can read from and write to the pipe. This mode gives the server the equivalent of **GENERIC\_READ** and **GENERIC\_WRITE** access to the pipe. The client can specify **GENERIC\_READ** or **GENERIC\_WRITE**, or both, when it connects to the pipe using the [**CreateFile**](https://msdn.microsoft.com/en-us/library/aa363858(v=vs.85).aspx) function. |
| **PIPE\_ACCESS\_INBOUND**  0x00000001 | The flow of data in the pipe goes from client to server only. This mode gives the server the equivalent of **GENERIC\_READ** access to the pipe. The client must specify **GENERIC\_WRITE** access when connecting to the pipe. If the client must read pipe settings by calling the [**GetNamedPipeInfo**](https://msdn.microsoft.com/en-us/library/aa365445(v=vs.85).aspx) or [**GetNamedPipeHandleState**](https://msdn.microsoft.com/en-us/library/aa365443(v=vs.85).aspx) functions, the client must specify **GENERIC\_WRITE** and **FILE\_READ\_ATTRIBUTES** access when connecting to the pipe. |
| **PIPE\_TYPE\_MESSAGE**  **0x00000004** | Data is written to the pipe as a stream of messages. The pipe treats the bytes written during each write operation as a message unit. The [GetLastError](https://msdn.microsoft.com/en-us/library/ms679360(v=vs.85).aspx) function returns ERROR\_MORE\_DATA when a message is not read completely. This mode can be used with either PIPE\_READMODE\_MESSAGE or PIPE\_READMODE\_BYTE. |
| **PIPE\_READMODE\_MESSAGE**  **0x00000002** | Data is read from the pipe as a stream of messages. This mode can be only used if PIPE\_TYPE\_MESSAGE is also specified. |
|  |  |
|  |  |
| **PIPE\_WAIT**  **0x00000000** | Blocking mode is enabled. When the pipe handle is specified in the [ReadFile](https://msdn.microsoft.com/en-us/library/aa365467(v=vs.85).aspx), [WriteFile](https://msdn.microsoft.com/en-us/library/aa365747(v=vs.85).aspx), or [ConnectNamedPipe](https://msdn.microsoft.com/en-us/library/aa365146(v=vs.85).aspx) function, the operations are not completed until there is data to read, all data is written, or a client is connected. Use of this mode can mean waiting indefinitely in some situations for a client process to perform an action. |
| **PIPE\_TYPE\_BYTE**  0x00000000 | Data is written to the pipe as a stream of bytes. This mode cannot be used with PIPE\_READMODE\_MESSAGE. The pipe does not distinguish bytes written during different write operations. |
| **PIPE\_READMODE\_BYTE**  **0x00000000** | Data is read from the pipe as a stream of bytes. This mode can be used with either PIPE\_TYPE\_MESSAGE or PIPE\_TYPE\_BYTE. |
| **FILE\_FLAG\_WRITE\_THROUGH**  **0x80000000** | Write-through mode is enabled. This mode affects only write operations on byte-type pipes and, then, only when the client and server processes are on different computers. If this mode is enabled, functions writing to a named pipe do not return until the data written is transmitted across the network and is in the pipe's buffer on the remote computer. If this mode is not enabled, the system enhances the efficiency of network operations by buffering data until a minimum number of bytes accumulate or until a maximum time elapses. |

**ERROR\_BROKEN\_PIPE**

109 (0x6D)

The pipe has been ended.

**ERROR\_MORE\_DATA**

234 (0xEA)

More data is available.

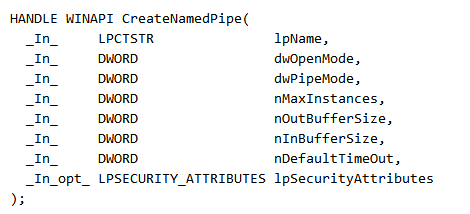
**ERROR\_PIPE\_CONNECTED**

535 (0x217)

There is a process on other end of the pipe.

*nMaxInstances* [in]

The maximum number of instances that can be created for this pipe. The first instance of the pipe can specify this value; the same number must be specified for other instances of the pipe. Acceptable values are in the range 1 through **PIPE\_UNLIMITED\_INSTANCES** (255). 255 == ff



lpName是管道的名称（\\.\pipe\管道名）

dwOpenMode打开模式，在代码中看到通过self.message参数指定创建了两种管道，第一种是双向的（PIPE\_ACCESS\_DUPLEX）；第二种是单向的（PIPE\_ACCESS\_INBOUND），从客户端到服务器，并都加上了FILE\_FLAG\_WRITE\_THROUGH这个flag，使得操作系统不得推迟对文件的写操作 。

dwPipeMode：

PIPE\_TYPE\_MESSAGE | PIPE\_READMODE\_MESSAGE | PIPE\_WAIT：以信息流方式写入与读取管道，管道阻塞

PIPE\_TYPE\_MESSAGE | PIPE\_READMODE\_MESSAGE | PIPE\_WAIT：以字节方式写入与读取管道，管道阻塞

nMaxInstances: PIPE\_UNLIMITED\_INSTANCES，实例个数无限

nInBufferSize: BUFSIZE = 0x10000 或0

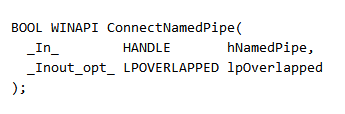
nOutBufferSize: BUFSIZE = 0x10000

剩下两个参数都是空指针

返回了pipe\_handle这个管道句柄

if KERNEL32.ConnectNamedPipe(pipe\_handle, None) or \  
 KERNEL32.GetLastError() == ERROR\_PIPE\_CONNECTED:  
 handler = self.pipe\_handler(pipe\_handle, \*\*self.kwargs)  
 handler.daemon = True  
 handler.start()

然后根据pipe\_handler开始管道线程



Pipe\_handler有两种，一种是Dispatcher，将命令通过Dispatcher发送，然后获得返回（双向管道），另一种是Pipeforwarder，只负责转发虚拟机向宿主机的转发（单向管道）。

注意第三句

handler = self.pipe\_handler(pipe\_handle, \*\*self.kwargs)

self.command\_pipe = PipeServer(  
 PipeDispatcher, self.config.pipe, message=True,  
 dispatcher=CommandPipeHandler(self)  
)

PipeServer的构造函数

def \_\_init\_\_(self, pipe\_handler, pipe\_name, message=False, \*\*kwargs):

在这个实例化的管道中，self.pipe\_handler是PipeDispatcher , pipe\_name是一个字符串，然后传入了一个字典{'dispatcher':CommandPipeHandler}

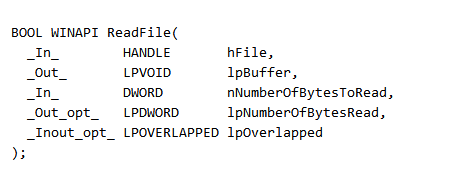
所以第三句其实是执行了PipeDispatcher(双向的pipe\_handle, dispatcher=CommandPipeHandler)，然后进行其run方法

def run(self):  
 *"""Run the pipe dispatcher."""* buf = create\_string\_buffer(BUFSIZE)  
 bytes\_written = c\_uint()  
  
 while self.do\_run:  
 message = self.\_read\_message(buf)  
 if not message:  
 break  
  
 response = self.dispatcher.dispatch(message) or "OK"  
  
 KERNEL32.WriteFile(  
 self.pipe\_handle, response, len(response),  
 byref(bytes\_written), None  
 )  
  
 KERNEL32.CloseHandle(self.pipe\_handle)

首先创建了一个缓冲区域和用来记录字节数的变量，然后不断执行\_read\_message(buf

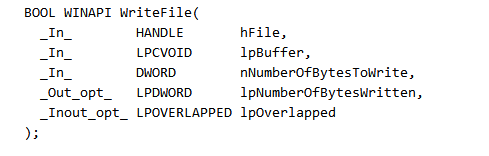
def \_read\_message(self, buf):  
 *"""Reads a message."""* bytes\_read = c\_uint()  
 ret = ""  
  
 while True:  
 success = KERNEL32.ReadFile(  
 self.pipe\_handle, byref(buf), sizeof(buf),  
 byref(bytes\_read), None  
 )  
  
 if KERNEL32.GetLastError() == ERROR\_MORE\_DATA:  
 ret += buf.raw[:bytes\_read.value]  
 elif success:  
 return ret + buf.raw[:bytes\_read.value]  
 else:  
 return

)



第一个参数是文件句柄，第二个参数缓冲指针，第三个参数是最大读取字节数，第四个参数是已读取字节的指针，第五个不用管，是空指针。

这个方法，就是不断的读取直到不返回Error\_more\_data，没有信息后退出，然后调用dispatcher（CommandPipeHandler）的dispatch()方法，同时写入宿主机的句柄文件。



def dispatch(self, data):  
 response = "NOPE"  
  
 if not data or ":" not in data:  
 log.critical("Unknown command received from the monitor: %r",  
 data.strip())  
 else:  
 # Backwards compatibility (old syntax is, e.g., "FILE\_NEW:" vs the  
 # new syntax, e.g., "1234:FILE\_NEW:").  
 if data[0].isupper():  
 command, arguments = data.strip().split(":", 1)  
 self.pid = None  
 else:  
 self.pid, command, arguments = data.strip().split(":", 2)  
  
 fn = getattr(self, "\_handle\_%s" % command.lower(), None)  
 if not fn:  
 log.critical("Unknown command received from the monitor: %r",  
 data.strip())  
 else:  
 try:  
 response = fn(arguments)  
 except:  
 log.exception(  
 "Pipe command handler exception occurred (command "  
 "%s args %r).", command, arguments  
 )  
  
 return response

而dispatch方法接收一个命令，获取对应的\_handle\_XXX方法并执行。

def \_handle\_process(self, data):  
 *"""Request for injection into a process."""* # Parse the process identifier.  
 if not data or not data.isdigit():  
 log.warning("Received PROCESS command from monitor with an "  
 "incorrect argument.")  
 return  
  
 return self.\_inject\_process(int(data), None, 0)  
  
def \_handle\_process2(self, data):  
 *"""Request for injection into a process using APC."""* # Parse the process and thread identifier.  
 if not data or data.count(",") != 2:  
 log.warning("Received PROCESS2 command from monitor with an "  
 "incorrect argument.")  
 return  
  
 pid, tid, mode = data.split(",")  
 if not pid.isdigit() or not tid.isdigit() or not mode.isdigit():  
 log.warning("Received PROCESS2 command from monitor with an "  
 "incorrect argument.")  
 return  
  
 return self.\_inject\_process(int(pid), int(tid), int(mode))  
  
def \_handle\_file\_new(self, data):  
 *"""Notification of a new dropped file."""* self.analyzer.files.add\_file(data.decode("utf8"), self.pid)  
  
def \_handle\_file\_del(self, data):  
 *"""Notification of a file being removed (if it exists) - we have to  
 dump it before it's being removed."""* filepath = data.decode("utf8")  
 if os.path.exists(filepath):  
 self.analyzer.files.delete\_file(filepath, self.pid)  
  
def \_handle\_file\_move(self, data):  
 *"""A file is being moved - track these changes."""* if "::" not in data:  
 log.warning("Received FILE\_MOVE command from monitor with an "  
 "incorrect argument.")  
 return  
  
 old\_filepath, new\_filepath = data.split("::", 1)  
 self.analyzer.files.move\_file(  
 old\_filepath.decode("utf8"), new\_filepath.decode("utf8"), self.pid  
 )  
  
def \_handle\_kill(self, data):  
 *"""A process is being killed."""* if not data.isdigit():  
 log.warning("Received KILL command with an incorrect argument.")  
 return  
  
 if self.analyzer.config.options.get("procmemdump"):  
 Process(pid=int(data)).dump\_memory()  
  
def \_handle\_dumpmem(self, data):  
 *"""Dump the memory of a process as it is right now."""* if not data.isdigit():  
 log.warning("Received DUMPMEM command with an incorrect argument.")  
 return  
  
 Process(pid=int(data)).dump\_memory()  
  
def \_handle\_dumpreqs(self, data):  
 if not data.isdigit():  
 log.warning("Received DUMPREQS command with an incorrect argument %r.", data)  
 return  
  
 pid = int(data)  
  
 if pid not in self.tracked:  
 log.warning("Received DUMPREQS command but there are no reqs for pid %d.", pid)  
 return  
  
 dumpreqs = self.tracked[pid].get("dumpreq", [])  
 for addr, length in dumpreqs:  
 log.debug("tracked dump req (%r, %r, %r)", pid, addr, length)  
  
 if not addr or not length:  
 continue  
  
 Process(pid=pid).dump\_memory\_block(int(addr), int(length))  
  
def \_handle\_track(self, data):  
 if not data.count(":") == 2:  
 log.warning("Received TRACK command with an incorrect argument %r.", data)  
 return  
  
 pid, scope, params = data.split(":", 2)  
 pid = int(pid)  
  
 paramtuple = params.split(",")  
 if pid not in self.tracked:  
 self.tracked[pid] = {}  
 if scope not in self.tracked[pid]:  
 self.tracked[pid][scope] = []  
 self.tracked[pid][scope].append(paramtuple)

def \_handle\_debug(self, data):  
 *"""Debug message from the monitor."""* log.debug(data)  
  
def \_handle\_info(self, data):  
 *"""Regular message from the monitor."""* log.info(data)  
  
def \_handle\_warning(self, data):  
 *"""Warning message from the monitor."""* log.warning(data)  
  
def \_handle\_critical(self, data):  
 *"""Critical message from the monitor."""* log.critical(data)  
  
def \_handle\_loaded(self, data):  
 *"""The monitor has loaded into a particular process."""* if not data or data.count(",") != 1:  
 log.warning("Received loaded command with incorrect parameters, "  
 "skipping it.")  
 return  
  
 pid, track = data.split(",")  
 if not pid.isdigit() or not track.isdigit():  
 log.warning("Received loaded command with incorrect parameters, "  
 "skipping it.")  
 return  
  
 self.analyzer.process\_lock.acquire()  
 self.analyzer.process\_list.add\_pid(int(pid), track=int(track))  
 self.analyzer.process\_lock.release()  
  
 log.debug("Loaded monitor into process with pid %s", pid)  
  
def \_handle\_getpids(self, data):  
 *"""Return the process identifiers of the agent and its parent  
 process."""* return struct.pack("II", self.analyzer.pid, self.analyzer.ppid)

这样每一条命令都会获得自己的属性，并被分配到不同的方法去。

再看第二种管道，这个就是日志用的单向管道。

self.log\_pipe\_server = PipeServer(  
 PipeForwarder, self.config.logpipe, destination=destination  
)

class PipeForwarder(threading.Thread):  
 *"""The Pipe Forwarder forwards all data received from a local pipe to  
 the Cuckoo server through a socket."""* sockets = {}  
 active = {}  
  
 def \_\_init\_\_(self, pipe\_handle, destination):  
 threading.Thread.\_\_init\_\_(self)  
 self.pipe\_handle = pipe\_handle  
 self.destination = destination  
  
 def run(self):  
 buf = create\_string\_buffer(BUFSIZE)  
 bytes\_read = c\_uint()  
 pid = c\_uint()  
  
 # The first four bytes indicate the process identifier. In case the  
 # pipe handle is closed in an unknown way, reopening one and  
 # specifying the same process identifier will reuse the same socket,  
 # thus making it look like as if it was never closed in the first  
 # place.  
 success = KERNEL32.ReadFile(  
 self.pipe\_handle, byref(pid), sizeof(pid),  
 byref(bytes\_read), None  
 )  
  
 ......  
 if pid.value:  
 if pid.value not in self.sockets:  
 self.sockets[pid.value] = (  
 socket.create\_connection(self.destination)  
 )  
  
 sock = self.sockets[pid.value]  
 self.active[pid.value] = True  
 else:  
 sock = socket.create\_connection(self.destination)  
  
 while True:  
 success = KERNEL32.ReadFile(  
 self.pipe\_handle, byref(buf), sizeof(buf),  
 byref(bytes\_read), None  
 )  
  
 if success or KERNEL32.GetLastError() == ERROR\_MORE\_DATA:  
 sock.sendall(buf.raw[:bytes\_read.value])  
 # If we get the broken pipe error then this pipe connection has  
 # been terminated for one reason or another. So break from the  
 # loop and make the socket "inactive", that is, another pipe  
 # connection can in theory pick it up. (This will only happen in  
 # cases where malware for some reason broke our pipe connection).  
 elif KERNEL32.GetLastError() == ERROR\_BROKEN\_PIPE:  
 break  
 else:  
 log.warning(  
 "The log pipe handler has failed, last error %d.",  
 KERNEL32.GetLastError()  
 )  
 break  
  
 if pid.value:  
 self.active[pid.value] = False

主要负责读文件（同上），然后不断通过socket发送到宿主机。这个类的执行方法是PipeForwarder

回到Analyzer的prepare():

class Analyzer(object):

def prepare(self):

.......

if self.config.category == "file":  
 self.target = os.path.join(  
 os.environ["TEMP"], self.config.file\_name  
 )  
elif self.config.category == "archive":  
 zip\_path = os.path.join(os.environ["TEMP"], self.config.file\_name)  
 zipfile.ZipFile(zip\_path).extractall(os.environ["TEMP"])  
 self.target = os.path.join(  
 os.environ["TEMP"], self.config.options["filename"]  
 )  
# If it's a URL, well.. we store the URL.  
else:  
 self.target = self.config.target

主要根据传入类型是URL、文件、文件夹类确定执行的操作（解压）。

返回上一层run():

def run(self):  
 *...*

self.prepare()  
 self.path = os.getcwd()

获得当前路径。

def run(self):

...

if not self.config.package:  
 log.debug(  
 "No analysis package specified, trying to detect "  
 "it automagically."  
 )  
  
 # If the analysis target is a file, we choose the package according  
 # to the file format.  
 if self.config.category == "file":  
 package = choose\_package(  
 self.config.file\_type, self.config.file\_name,  
 self.config.pe\_exports.split(",")  
 )  
 # If it's an URL, we'll just use the default Internet Explorer  
 # package.  
 else:  
 package = "ie"  
  
 # If we weren't able to automatically determine the proper package,  
 # we need to abort the analysis.  
 if not package:  
 raise CuckooError("No valid package available for file "  
 "type: {0}".format(self.config.file\_type))  
  
 log.info("Automatically selected analysis package \"%s\"", package)  
# Otherwise just select the specified package.  
 else:  
 package = self.config.package

如果配置文件有指定打开文件所用的Package，就从配置文件中读取Package。否则，就根据文件名、文件类型来用choose\_package确定Package

def choose\_package(file\_type, file\_name, exports):  
 *"""Choose analysis package due to file type and file extension.  
 @param file\_type: file type.  
 @param file\_name: file name.  
 @return: package name or None.  
 """* if not file\_type:  
 return None  
  
 file\_name = file\_name.lower()  
  
 if "DLL" in file\_type:  
 if file\_name.endswith(".cpl"):  
 return "cpl"  
 elif has\_com\_exports(exports):  
 return "com"  
 else:  
 return "dll"  
 elif "PE32" in file\_type or "MS-DOS" in file\_type:  
 return "exe"  
 elif "PDF" in file\_type or file\_name.endswith(".pdf"):  
 return "pdf"  
 elif file\_name.endswith(".pub"):  
 return "pub"  
 elif "Rich Text Format" in file\_type or \  
 "Microsoft Word" in file\_type or \  
 "Microsoft Office Word" in file\_type or \  
 file\_name.endswith((".doc", ".docx", ".rtf", ".docm")):  
 return "doc"  
 elif "Microsoft Office Excel" in file\_type or \  
 "Microsoft Excel" in file\_type or \  
 file\_name.endswith((".xls", ".xlsx", ".xlt", ".xlsm")):  
 return "xls"  
 elif "Microsoft PowerPoint" in file\_type or \  
 file\_name.endswith((".ppt", ".pptx", ".pps", ".ppsx", ".pptm", ".potm", ".potx", ".ppsm")):  
 return "ppt"  
 elif file\_name.endswith(".jar"):  
 return "jar"  
 elif file\_name.endswith(".hta"):  
 return "hta"  
 elif "Zip" in file\_type:  
 return "zip"  
 elif file\_name.endswith((".py", ".pyc")) or "Python script" in file\_type:  
 return "python"  
 elif file\_name.endswith(".vbs"):  
 return "vbs"  
 elif file\_name.endswith((".js", ".jse")):  
 return "js"  
 elif file\_name.endswith(".msi"):  
 return "msi"  
 elif file\_name.endswith(".ps1"):  
 return "ps1"  
 elif file\_name.endswith((".wsf", ".wsc")):  
 return "wsf"  
 elif "HTML" in file\_type or file\_name.endswith((".htm", ".html", ".hta", ".mht", ".mhtml")):  
 return "ie"  
 else:  
 return "generic"

def run(self):

...  
 package\_name = "modules.packages.%s" % package  
  
 # Try to import the analysis package.  
 try:  
 \_\_import\_\_(package\_name, globals(), locals(), ["dummy"], -1)  
 # If it fails, we need to abort the analysis.  
 except ImportError:  
 raise CuckooError("Unable to import package \"{0}\", does "  
 "not exist.".format(package\_name))  
  
 # Initialize the package parent abstract.  
 Package()  
  
 # Enumerate the abstract subclasses.  
 try:  
 package\_class = Package.\_\_subclasses\_\_()[0]  
 except IndexError as e:  
 raise CuckooError("Unable to select package class "  
 "(package={0}): {1}".format(package\_name, e))  
  
 # Initialize the analysis package.  
 self.package = package\_class(self.config.options, analyzer=self)  
  
 # Move the sample to the current working directory as provided by the  
 # task - one is able to override the starting path of the sample.  
 # E.g., for some samples it might be useful to run from %APPDATA%  
 # instead of %TEMP%.  
 if self.config.category == "file":  
 self.target = self.package.move\_curdir(self.target)

获得对应的package名字之后，就通过\_\_import\_\_函数导入对应的Package类，随后先实例化一个基类Package，获得其子类也就是需要使用的Package。然后通过move\_curdir将文件移动到cwd（）

def move\_curdir(self, filepath):  
 *"""Move a file to the current working directory so it can be executed  
 from there.  
 @param filepath: the file to be moved  
 @return: the new filepath  
 """* outpath = os.path.join(self.curdir, os.path.basename(filepath))  
 os.rename(filepath, outpath)  
 return outpath

def run(self):

...

# Initialize Auxiliary modules  
 Auxiliary()  
 prefix = auxiliary.\_\_name\_\_ + "."  
 for loader, name, ispkg in pkgutil.iter\_modules(auxiliary.\_\_path\_\_, prefix):  
 if ispkg:  
 continue  
  
 # Import the auxiliary module.  
 try:  
 \_\_import\_\_(name, globals(), locals(), ["dummy"], -1)  
 except ImportError as e:  
 log.warning("Unable to import the auxiliary module "  
 "\"%s\": %s", name, e)  
  
 # Walk through the available auxiliary modules.  
 aux\_enabled, aux\_avail = [], []  
 for module in Auxiliary.\_\_subclasses\_\_():  
 # Try to start the auxiliary module.  
 try:  
 aux = module(options=self.config.options, analyzer=self)  
 aux\_avail.append(aux)  
 aux.init()  
 aux.start()  
 except (NotImplementedError, AttributeError):  
 log.exception(  
 "Auxiliary module %s was not implemented", module.\_\_name\_\_  
 )  
 except CuckooDisableModule:  
 continue  
 except Exception as e:  
 log.exception(  
 "Cannot execute auxiliary module %s: %s",  
 module.\_\_name\_\_, e  
 )  
 else:  
 log.debug("Started auxiliary module %s",  
 module.\_\_name\_\_)  
 aux\_enabled.append(aux)

这一部分以相同方法加载辅助模块，放在列表aux\_enable中

def run(self):  
 # Forward the command pipe and logpipe names on to zer0m0n.  
 zer0m0n.cmdpipe(self.config.pipe)  
 zer0m0n.channel(self.config.logpipe)  
  
 # Initialize zer0m0n with our compiled Yara rules.  
 zer0m0n.yarald("bin/rules.yarac")

上面两句通过传入self.config.pipe使用了zer0m0n.cmdpipe方法

class Zer0m0nIoctl(Ioctl):  
 actions = [  
 "addpid",  
 "cmdpipe",  
 "channel",  
 "dumpmem",  
 "yarald",  
 ]  
  
 def invoke(self, action, buf):  
 if action not in self.actions:  
 raise RuntimeError("Invalid ioctl action: %s" % action)  
  
 return Ioctl.invoke(  
 self, CTL\_CODE\_BASE + self.actions.index(action) \* 4, buf,  
 )  
  
 def addpid(self, pid):  
 return self.invoke("addpid", struct.pack("I", pid))  
  
 def cmdpipe(self, pipe):  
 return self.invoke("cmdpipe", "\x00".join(pipe + "\x00"))  
  
 def channel(self, pipe):  
 return self.invoke("channel", "\x00".join(pipe + "\x00"))  
  
 def dumpmem(self, pid):  
 return self.invoke("dumpmem", struct.pack("I", pid))  
  
 def yarald(self, rulepath):  
 return self.invoke("yarald", open(rulepath, "rb").read())  
  
zer0m0n = Zer0m0nIoctl(driver\_name)

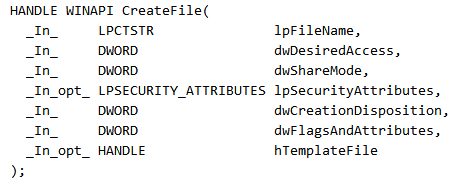
Zer0m0nIoctl继承了Ioctl，其构造方法是传入管道的路径。

完成了一系列的转化：

self.cmdpipe(self, pipe)->self.invoke("cmdpipe", "\x00".join(pipe + "\x00"))-> Ioctl.invoke(self, CTL\_CODE\_BASE + self.actions.index(action) \* 4, buf)

class Ioctl(object):  
 def \_\_init\_\_(self, pipepath):  
 self.pipepath = pipepath  
  
 def invoke(self, ctlcode, value, outlength=0x1000):  
 # *TODO Enable the kernel drivers.* return  
  
 device\_handle = KERNEL32.CreateFileA(  
 "\\\\.\\%s" % self.pipepath, GENERIC\_READ | GENERIC\_WRITE,  
 FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, None, OPEN\_EXISTING, 0, None  
 ) % 2\*\*32  
  
 if device\_handle == 0xffffffff:  
 # Only report an error if the error is not "name not found",  
 # indicating that no kernel analysis is currently taking place.  
 if KERNEL32.GetLastError() != 2:  
 log.warning(  
 "Error opening handle to driver (%s): %d!",  
 driver\_name, KERNEL32.GetLastError()  
 )  
 return False  
  
 out = ctypes.create\_string\_buffer(outlength)  
 length = ctypes.c\_uint()  
  
 ret = KERNEL32.DeviceIoControl(  
 device\_handle, ctlcode, value, len(value), out,  
 ctypes.sizeof(out), ctypes.byref(length), None  
 )  
 KERNEL32.CloseHandle(device\_handle)  
  
 if not ret:  
 log.warning(  
 "Error performing ioctl (0x%08x): %d!",  
 ctlcode, KERNEL32.GetLastError()  
 )  
 return False  
  
 return out.raw[:length.value]

首先通过KERNEL32.CreateFile创建了一个device\_handle文件句柄，其名称为self.pipepath，加上通读、通写、共享读写权限，第四个参数*pSecurityAttributes* 为空指针说明返回的句柄无法被子进程继承，与句柄相关的文件和设备都获得默认安全描述符。第五个参数*dwCreationDisposition* 是常用的OPEN\_EXISTING，表示只在文件或设备存在的时候打开文件，否则GetLastError是ERROR\_FILE\_NOT\_FOUND。dwFlagsAndAttributes指向一个SECURITY\_ATTRIBUTES结构的指针，定义了文件的安全特性（如果操作系统支持的话）。hTemplateFilehTemplateFile为一个文件或设备句柄，表示按这个参数给出的句柄为模板创建文件（就是将该句柄文件拷贝到lpFileName指定的路径，然后再打开）。它将指定该文件的属性扩展到新创建的文件上面，这个参数可用于将某个新文件的属性设置成与现有文件一样，并且这样会忽略dwAttrsAndFlags。通常这个参数设置为NULL，为空表示不使用模板，一般为空指针。



|  |  |
| --- | --- |
| GENERIC\_READ | Read access |
| GENERIC\_WRITE | Write access |
| FILE\_SHARE\_READ  0x00000001 | Enables subsequent open operations on a file or device to request read access.  Otherwise, other processes cannot open the file or device if they request read access.  If this flag is not specified, but the file or device has been opened for read access, the function fails. |
| FILE\_SHARE\_WRITE  0x00000002 | Enables subsequent open operations on a file or device to request write access.  Otherwise, other processes cannot open the file or device if they request write access.  If this flag is not specified, but the file or device has been opened for write access or has a file mapping with write access, the function fails. |
| OPEN\_EXISTING  3 | Opens a file or device, only if it exists.  If the specified file or device does not exist, the function fails and the last-error code is set toERROR\_FILE\_NOT\_FOUND (2).  For more information about devices, see the Remarks section. |

def invoke(self, ctlcode, value, outlength=0x1000):  
 ...

out = ctypes.create\_string\_buffer(outlength)  
 length = ctypes.c\_uint()  
  
 ret = KERNEL32.DeviceIoControl(  
 device\_handle, ctlcode, value, len(value), out,  
 ctypes.sizeof(out), ctypes.byref(length), None  
 )  
 KERNEL32.CloseHandle(device\_handle)

先创建出outlength个长度的字符数组指针和int类型length，然后使用DeviceIoControl控制

hDevice Long，设备句柄

dwIoControlCode Long，应用程序调用驱动程序的控制命令，就是IOCTL\_XXX IOCTLs。

lpInBuffer Any，应用程序传递给驱动程序的数据缓冲区地址。

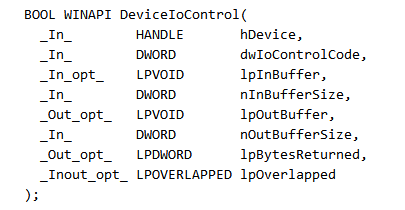
nInBufferSize Long，应用程序传递给驱动程序的数据缓冲区大小，字节数。

lpOutBuffer Any，驱动程序返回给应用程序的数据缓冲区地址。

nOutBufferSize Long，驱动程序返回给应用程序的数据缓冲区大小，字节数。

lpBytesReturned Long，驱动程序实际返回给应用程序的数据字节数地址。

lpOverlapped OVERLAPPED，这个结构用于重叠操作。针对同步操作，请用ByVal As Long传递零值。



最后获得控制结果ret，关闭句柄，返回缓冲out的前length个字节

def run(self):

...

# Start analysis package. If for any reason, the execution of the  
 # analysis package fails, we have to abort the analysis.  
 pids = self.package.start(self.target)  
  
 # If the analysis package returned a list of process identifiers, we  
 # add them to the list of monitored processes and enable the process monitor.  
 if pids:  
 self.process\_list.add\_pids(pids)  
 pid\_check = True  
  
 # If the package didn't return any process ID (for example in the case  
 # where the package isn't enabling any behavioral analysis), we don't  
 # enable the process monitor.  
 else:  
 log.info("No process IDs returned by the package, running "  
 "for the full timeout.")  
 pid\_check = False  
  
 # Check in the options if the user toggled the timeout enforce. If so,  
 # we need to override pid\_check and disable process monitor.  
 if self.config.enforce\_timeout:  
 log.info("Enabled timeout enforce, running for the full timeout.")  
 pid\_check = False

由于之前已经通过 self.package = package\_class(self.config.options, analyzer=self)获得了package的具体名称，这是就具体调用某一个package，执行其start方法。比如xls文件的话，会这样：

这是一个示例

class XLS(Package):  
 *"""Excel analysis package."""* PATHS = [  
 ("ProgramFiles", "Microsoft Office", "EXCEL.EXE"),  
 ("ProgramFiles", "Microsoft Office", "Office10", "EXCEL.EXE"),  
 ("ProgramFiles", "Microsoft Office", "Office11", "EXCEL.EXE"),  
 ("ProgramFiles", "Microsoft Office", "Office12", "EXCEL.EXE"),  
 ("ProgramFiles", "Microsoft Office", "Office14", "EXCEL.EXE"),  
 ("ProgramFiles", "Microsoft Office", "Office15", "EXCEL.EXE"),  
 ("ProgramFiles", "Microsoft Office", "Office16", "EXCEL.EXE"),  
 ("ProgramFiles", "Microsoft Office 15", "root", "office15", "EXCEL.EXE"),  
 ("ProgramFiles", "Microsoft Office", "root", "Office16", "EXCEL.EXE"),  
 ]  
  
 REGKEYS = [  
 [  
 HKEY\_CURRENT\_USER,  
 "Software\\Microsoft\\Office\\12.0\\Common\\General",  
 {  
 # "Welcome to the 2007 Microsoft Office system"  
 "ShownOptIn": 1,  
 },  
 ],  
 [  
 HKEY\_CURRENT\_USER,  
 "Software\\Microsoft\\Office\\12.0\\Excel\\Security",  
 {  
 # Enable VBA macros in Office 2007.  
 "VBAWarnings": 1,  
 "AccessVBOM": 1,  
  
 # "The file you are trying to open .xyz is in a different  
 # format than specified by the file extension. Verify the file  
 # is not corrupted and is from trusted source before opening  
 # the file. Do you want to open the file now?"  
 "ExtensionHardening": 0,  
 },  
 ],  
 [  
 HKEY\_CURRENT\_USER,  
 "Software\\Microsoft\\Office\\Common\\Security",  
 {  
 # Enable all ActiveX controls without restrictions & prompting.  
 "DisableAllActiveX": 0,  
 "UFIControls": 1,  
 },  
 ],  
 ]  
  
 def start(self, path):  
 excel = self.get\_path("Microsoft Office Excel")  
 return self.execute(  
 excel, args=[path], mode="office", trigger="file:%s" % path  
 )

def enum\_paths(self):  
 *"""Enumerate available paths."""* basepaths = {  
 "System32": [  
 os.path.join(os.getenv("SystemRoot"), "System32"),  
 os.path.join(os.getenv("SystemRoot"), "SysWOW64"),  
 ],  
 "ProgramFiles": [  
 os.getenv("ProgramFiles").replace(" (x86)", ""),  
 os.getenv("ProgramFiles(x86)"),  
 ],  
 "HomeDrive": [  
 # os.path.join() doesn't work well if you give it just "C:"  
 # so manually append a backslash.  
 os.getenv("HomeDrive") + "\\",  
 ],  
 }  
  
 for path in self.PATHS:  
 basedir = path[0]  
 for basepath in basepaths.get(basedir, [basedir]):  
 if not basepath or not os.path.isdir(basepath):  
 continue  
  
 yield os.path.join(basepath, \*path[1:])  
  
def get\_path(self, application):  
 *"""Search for the application in all available paths.  
 @param applicaiton: application executable name  
 @return: executable path  
 """* for path in self.enum\_paths():  
 if os.path.isfile(path):  
 return path  
  
 raise CuckooPackageError("Unable to find any %s executable." %  
 application)

首先会通过XLS.get\_path函数，在PATHS这个列表和环境变量中寻找虚拟机中真正执行文件的exe，然后执行父类Package的execute方法，返回了pid。

def execute(self, path, args, mode=None, maximize=False, env=None,  
 source=None, trigger=None):  
 *"""Starts an executable for analysis.  
 @param path: executable path  
 @param args: executable arguments  
 @param mode: monitor mode - which functions to instrument  
 @param maximize: whether the GUI should start maximized  
 @param env: additional environment variables  
 @param source: parent process of our process  
 @param trigger: trigger to indicate analysis start  
 @return: process pid  
 """* dll = self.options.get("dll")  
 free = self.options.get("free")  
 analysis = self.options.get("analysis")  
  
 # Kernel analysis overrides the free argument.  
 if analysis == "kernel":  
 free = True  
  
 source = source or self.options.get("from")  
 mode = mode or self.options.get("mode")  
  
 if not trigger and self.options.get("trigger"):  
 if self.options["trigger"] == "exefile":  
 trigger = "file:%s" % path  
  
 # Setup pre-defined registry keys.  
 self.init\_regkeys(self.REGKEYS)  
  
 p = Process()  
 if not p.execute(path=path, args=args, dll=dll, free=free,  
 curdir=self.curdir, source=source, mode=mode,  
 maximize=maximize, env=env, trigger=trigger):  
 raise CuckooPackageError(  
 "Unable to execute the initial process, analysis aborted."  
 )  
  
 return p.pid

在execute方法中，实际执行的又是Process的execute方法

class Process(object):

...  
 def execute(self, path, args=None, dll=None, free=False, curdir=None,  
 source=None, mode=None, maximize=False, env=None,  
 trigger=None):  
 *"""Execute sample process.  
 @param path: sample path.  
 @param args: process args.  
 @param dll: dll path.  
 @param free: do not inject our monitor.  
 @param curdir: current working directory.  
 @param source: process identifier or process name which will  
 become the parent process for the new process.  
 @param mode: monitor mode - which functions to instrument.  
 @param maximize: whether the GUI should be maximized.  
 @param env: environment variables.  
 @param trigger: trigger to indicate analysis start  
 @return: operation status.  
 """* if not os.access(path, os.X\_OK):  
 log.error(  
 "Unable to access file at path %r, execution aborted!", path  
 )  
 return False  
  
 is32bit = self.is32bit(path=path)

先通过is32bit来确定系统是否32位

class Process(object):

...  
 def is32bit(self, pid=None, process\_name=None, path=None):  
 *"""Is a PE file 32-bit or does a process identifier belong to a  
 32-bit process.  
 @param pid: process identifier.  
 @param process\_name: process name.  
 @param path: path to a PE file.  
 @return: boolean or exception.  
 """* count = (pid is None) + (process\_name is None) + (path is None)  
 if count != 2:  
 raise CuckooError("Invalid usage of is32bit, only one identifier "  
 "should be specified")  
  
 is32bit\_exe = os.path.join("bin", "is32bit.exe")  
  
 if pid:  
 args = [is32bit\_exe, "-p", "%s" % pid]  
 elif process\_name:  
 args = [is32bit\_exe, "-n", process\_name]  
  
 # If we're running a 32-bit Python in a 64-bit Windows system and the  
 # path points to System32, then we hardcode it as being a 64-bit  
 # binary. (To be fair, a 64-bit Python on 64-bit Windows would also  
 # make the System32 binary 64-bit).  
 elif os.path.isdir("C:\\Windows\\Sysnative") and \  
 path.lower().startswith("c:\\windows\\system32"):  
 return False  
 elif not os.path.exists(path):  
 raise CuckooError("File not found: %s" % path)  
 else:  
 args = [is32bit\_exe, "-f", self.shortpath(path)]  
  
 try:  
 bitsize = int(subprocess\_checkoutput(args))  
 except subprocess.CalledProcessError as e:  
 raise CuckooError("Error returned by is32bit: %s" % e.output)  
  
 return bitsize == 32

# path points to System32, then we hardcode it as being a 64-bit  
 # binary. (To be fair, a 64-bit Python on 64-bit Windows would also  
 # make the System32 binary 64-bit).  
 elif os.path.isdir("C:\\Windows\\Sysnative") and \  
 path.lower().startswith("c:\\windows\\system32"):  
 return False  
 elif not os.path.exists(path):  
 raise CuckooError("File not found: %s" % path)  
 else:  
 args = [is32bit\_exe, "-f", self.shortpath(path)]  
  
 try:  
 bitsize = int(subprocess\_checkoutput(args))  
 except subprocess.CalledProcessError as e:  
 raise CuckooError("Error returned by is32bit: %s" % e.output)  
  
 return bitsize == 32

class Process(object):

...  
 def execute(self, path, args=None, dll=None, free=False, curdir=None,  
 source=None, mode=None, maximize=False, env=None,  
 trigger=None):

...  
 if not dll:

if is32bit:  
 dll = "monitor-x86.dll"  
 else:  
 dll = "monitor-x64.dll"  
  
 dllpath = os.path.abspath(os.path.join("bin", dll))  
  
 if not os.path.exists(dllpath):  
 log.warning("No valid DLL specified to be injected, "  
 "injection aborted.")  
 return False

如果是32bit，就使用monitor-x86.dll， 否则使用monitor-x64.dll。这个是确定注入dll的使用。

class Process(object):  
 ...  
 def execute(self, path, args=None, dll=None, free=False, curdir=None,  
 source=None, mode=None, maximize=False, env=None,  
 trigger=None):  
 ...  
 if free:  
 argv.append("--free")  
 else:  
 argv += [  
 "--apc",  
 "--dll", dllpath,  
 "--config", self.drop\_config(mode=mode, trigger=trigger),  
 ]  
  
 try:  
 subprocess\_checkoutput(argv, env)  
 except subprocess.CalledProcessError as e:  
 log.error(  
 "Failed to execute process from path %r with "  
 "arguments %r (Error: %s)", path, argv, e  
 )  
 return False  
  
 log.info("Successfully executed process from path %r with "  
 "arguments %r and pid %d", path, args or "", self.pid)  
 return True

加上参数后，开始使用命令行来执行。命令大概是

bin\inject-x86.exe --app 路径 --onlystart --args参数 --curdir 当前目录 --from 程序来源 --maximize，返回成功执行与否

返回到原来的run函数

def run(self):

...  
 while self.do\_run:  
 self.time\_counter += 1  
 if self.time\_counter == int(self.config.timeout):  
 log.info("Analysis timeout hit, terminating analysis.")  
 break  
  
 # If the process lock is locked, it means that something is  
 # operating on the list of monitored processes. Therefore we  
 # cannot proceed with the checks until the lock is released.  
 if self.process\_lock.locked():  
 KERNEL32.Sleep(1000)  
 continue  
  
 try:  
 # If the process monitor is enabled we start checking whether  
 # the monitored processes are still alive.  
 if pid\_check:  
 for pid in self.process\_list.pids:  
 if not Process(pid=pid).is\_alive():  
 log.info("Process with pid %s has terminated", pid)  
 self.process\_list.remove\_pid(pid)  
  
 # If none of the monitored processes are still alive, we  
 # can terminate the analysis.  
 if not self.process\_list.pids:  
 log.info("Process list is empty, "  
 "terminating analysis.")  
 break  
  
 # Update the list of monitored processes available to the  
 # analysis package. It could be used for internal  
 # operations within the module.  
 self.package.set\_pids(self.process\_list.pids)  
  
 try:  
 # The analysis packages are provided with a function that  
 # is executed at every loop's iteration. If such function  
 # returns False, it means that it requested the analysis  
 # to be terminate.  
 if not self.package.check():  
 log.info("The analysis package requested the "  
 "termination of the analysis.")  
 break  
  
 # If the check() function of the package raised some exception  
 # we don't care, we can still proceed with the analysis but we  
 # throw a warning.  
 except Exception as e:  
 log.warning("The package \"%s\" check function raised "  
 "an exception: %s", package\_name, e)  
 finally:  
 # Zzz.  
 KERNEL32.Sleep(1000)

这些代码只是在检测进程是否在运行，如果不在运行，就从pid列表中移除，self.do\_run是程序是否在运行的信号。

def run(self):

...  
 # Create the shutdown mutex.  
 KERNEL32.CreateMutexA(None, False, SHUTDOWN\_MUTEX)  
  
 try:  
 # Before shutting down the analysis, the package can perform some  
 # final operations through the finish() function.  
 self.package.finish()  
 except Exception as e:  
 log.warning("The package \"%s\" finish function raised an "  
 "exception: %s", package\_name, e)  
  
 try:  
 # Upload files the package created to package\_files in the  
 # results folder.  
 for path, name in self.package.package\_files() or []:  
 upload\_to\_host(path, os.path.join("package\_files", name))  
 except Exception as e:  
 log.warning("The package \"%s\" package\_files function raised an "  
 "exception: %s", package\_name, e)  
  
 # Terminate the Auxiliary modules.  
 for aux in aux\_enabled:  
 try:  
 aux.stop()  
 except (NotImplementedError, AttributeError):  
 continue  
 except Exception as e:  
 log.warning("Cannot terminate auxiliary module %s: %s",  
 aux.\_\_class\_\_.\_\_name\_\_, e)  
  
 if self.config.terminate\_processes:  
 # Try to terminate remaining active processes.  
 log.info("Terminating remaining processes before shutdown.")  
  
 for pid in self.process\_list.pids:  
 proc = Process(pid=pid)  
 if proc.is\_alive():  
 try:  
 proc.terminate()  
 except:  
 continue  
  
 # Run the finish callback of every available Auxiliary module.  
 for aux in aux\_avail:  
 try:  
 aux.finish()  
 except (NotImplementedError, AttributeError):  
 continue  
 except Exception as e:  
 log.warning("Exception running finish callback of auxiliary "  
 "module %s: %s", aux.\_\_class\_\_.\_\_name\_\_, e)  
  
 # Let's invoke the completion procedure.  
 self.complete()  
 return True

最后再创建关闭程序的互斥锁，然后执行一些清理操作。

至此，沙箱虚拟机的全部操作就已经完成了。主程序中还有一些清理、报告的操作。