IDA插件之动态数据解析器（DDR）精讲

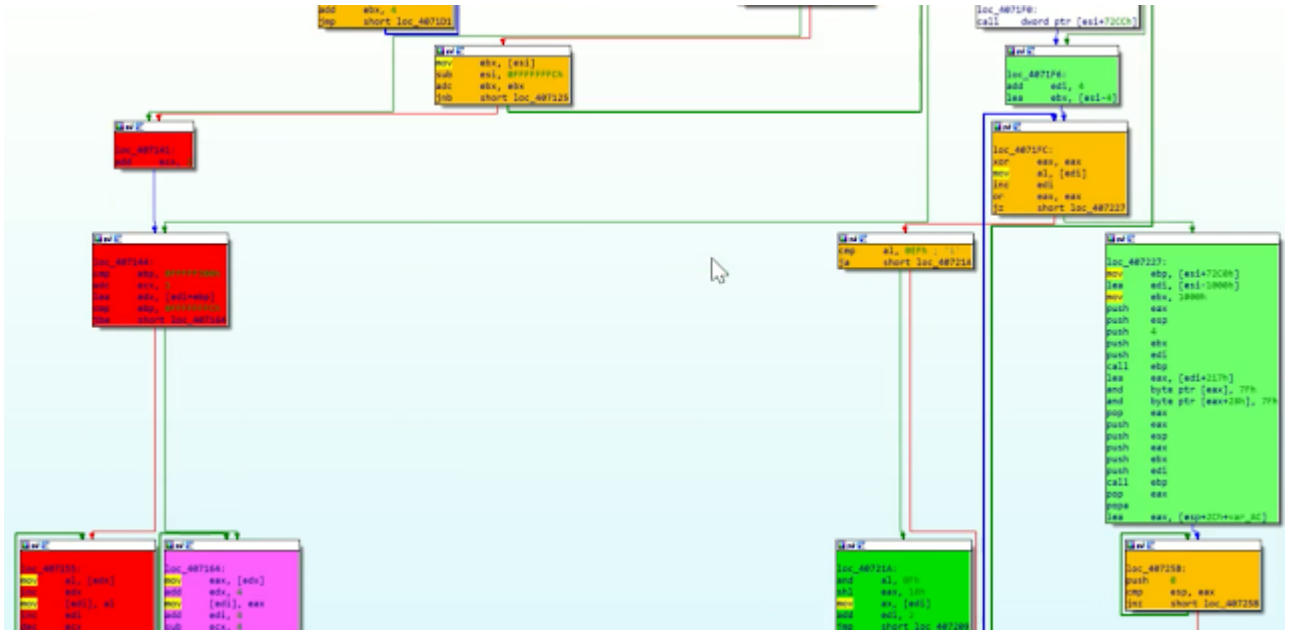
Pinging / 2019-01-21 21:37:00 / 浏览数 3826 安全工具 工具 顶(0) 踩(0)

摘要

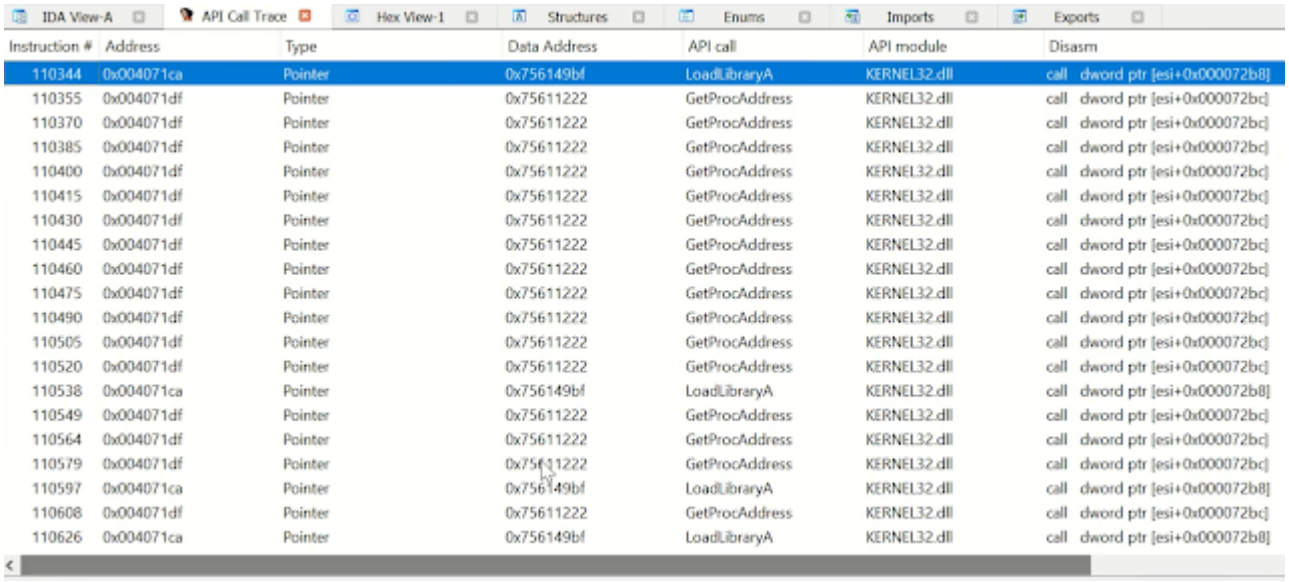
使用IDA在进行静态逆向的过程中常常会出现一些问题。由于程序中的某些值是在运行时计算的，所以这使得分析人员很难理解某些模块的具体工作。在分析恶意软件的过程

工具特征

代码流跟踪
（显示执行了大约20种不同颜色的基本块的执行次数）：

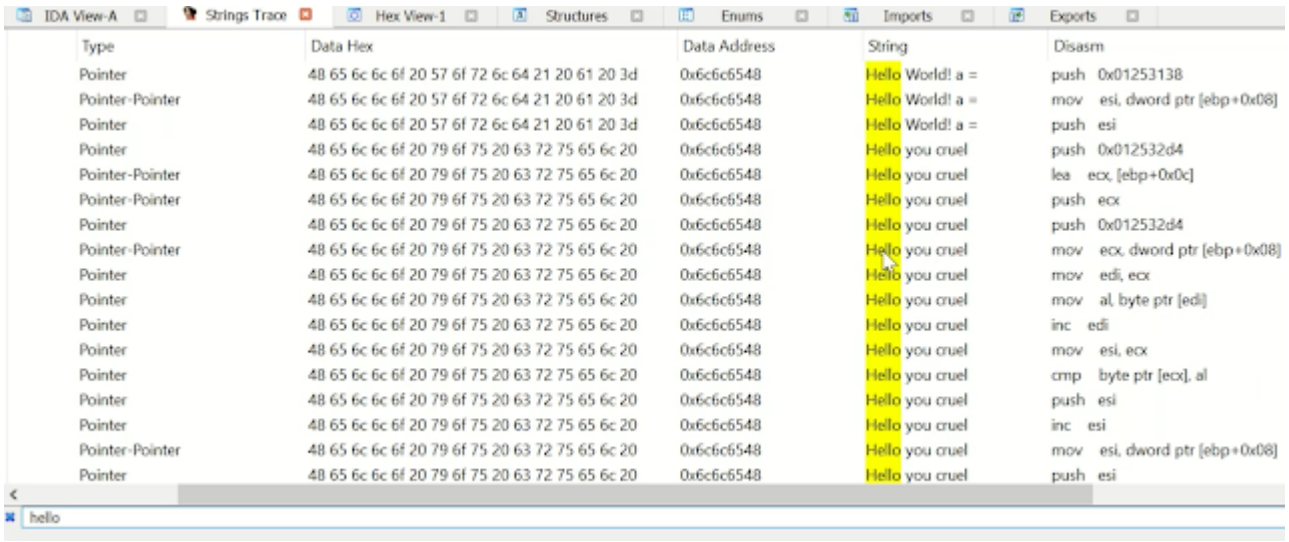


可搜索的API调用日志：
（包括所有出现的指令，如call、jxx、API地址等）



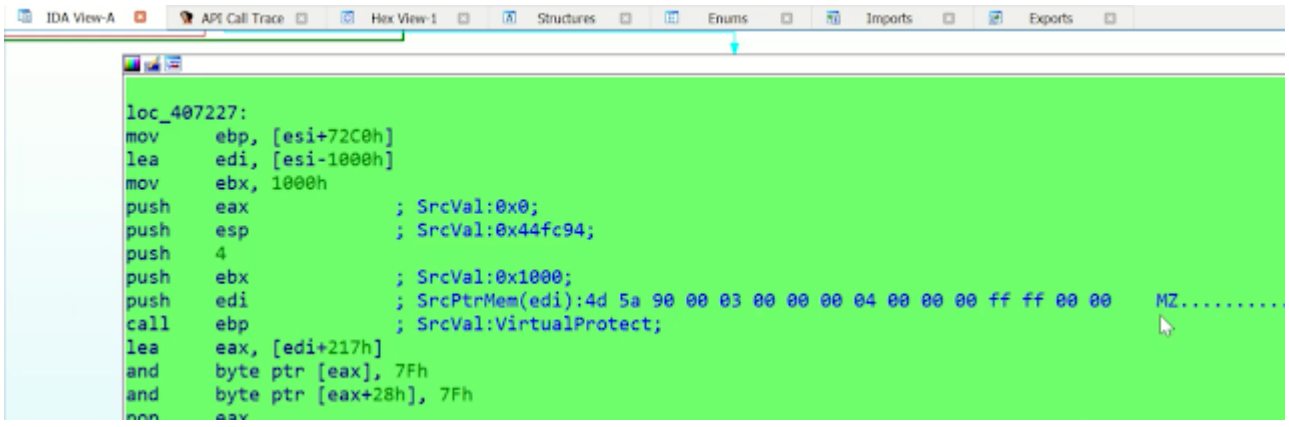
Instruction #	Address	Type	Data Address	API call	API module	Disasm
110344	0x004071ca	Pointer	0x756149bf	LoadLibraryA	KERNEL32.dll	call dword ptr [esi+0x000072b8]
110355	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110370	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110385	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110400	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110415	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110430	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110445	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110460	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110475	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110490	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110505	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110520	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110538	0x004071ca	Pointer	0x756149bf	LoadLibraryA	KERNEL32.dll	call dword ptr [esi+0x000072b8]
110549	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110564	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110579	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110597	0x004071ca	Pointer	0x756149bf	LoadLibraryA	KERNEL32.dll	call dword ptr [esi+0x000072b8]
110608	0x004071df	Pointer	0x75611222	GetProcAddress	KERNEL32.dll	call dword ptr [esi+0x000072bc]
110626	0x004071ca	Pointer	0x756149bf	LoadLibraryA	KERNEL32.dll	call dword ptr [esi+0x000072b8]

可搜索的字符串记录：



Type	Data Hex	Data Address	String	Disasm
Pointer	48 65 6c 6c 6f 20 57 6f 72 6c 64 21 20 61 20 3d	0x6c6c6548	Hello World! a =	push 0x01253138
Pointer-Pointer	48 65 6c 6c 6f 20 57 6f 72 6c 64 21 20 61 20 3d	0x6c6c6548	Hello World! a =	mov esi, dword ptr [ebp+0x08]
Pointer	48 65 6c 6c 6f 20 57 6f 72 6c 64 21 20 61 20 3d	0x6c6c6548	Hello World! a =	push esi
Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	push 0x012532d4
Pointer-Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	lea ecx, [ebp+0x0c]
Pointer-Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	push ecx
Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	push 0x012532d4
Pointer-Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	mov ecx, dword ptr [ebp+0x08]
Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	mov edi, ecx
Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	mov al, byte ptr [edi]
Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	inc edi
Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	mov esi, ecx
Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	cmp byte ptr [ecx], al
Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	push esi
Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	inc esi
Pointer-Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	mov esi, dword ptr [ebp+0x08]
Pointer	48 65 6c 6c 6f 20 79 6f 75 20 63 72 75 65 6c 20	0x6c6c6548	Hello you cruel	push esi

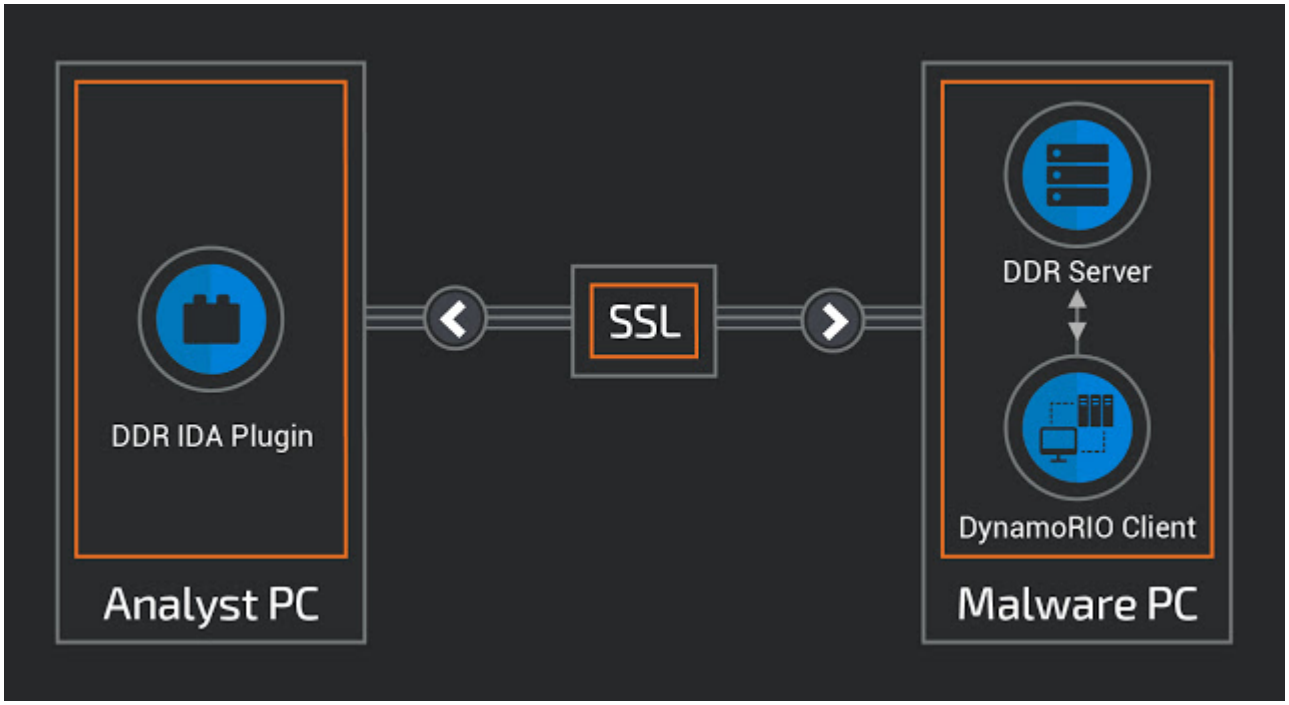
解析动态值和自动评论部分：



技术细节

结构与使用方法

DDR具有下图所示的客户端/服务器架构。.DDR IDA插件和DDR■■■均是用Python脚本编写的。
DynamoRIO客户端是用C编写的DLL，并由DynamoRIO工具调用drrun.exe执行。
此DLL使用检测技术在运行时用于分析和监视恶意软件。通常，所有进程都通过插件进行控制。
完成DynamoRIO客户端的后端分析后，其结果将发送回插件。我们选择JSON作为此数据的格式，以便用户可以轻松地读取和解析第三方Python脚本。



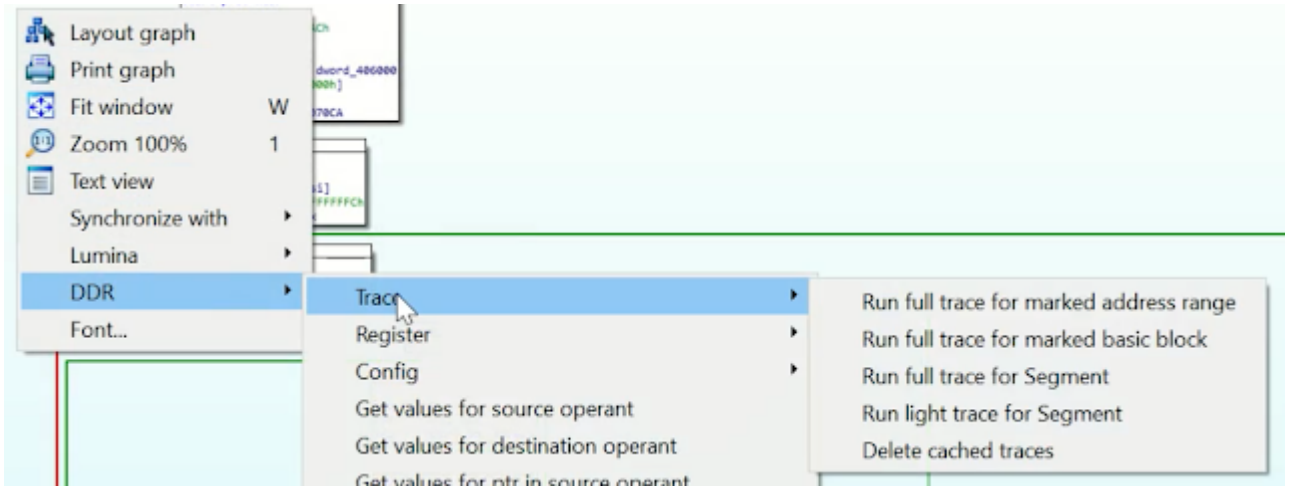
从理论上讲，用户可以在同一台PC上运行插件和服务器，但就恶意软件样本的执行情况而言，强烈建议在单独的计算机上执行此操作。

在大多数情况下，我们可以按照下面所述的插件安装，从IDA中的DDR
/Trace菜单开始分析，如果我们想在无Python环境的系统中执行此恶意软件并进行分析，那么插件会返回不支持，所以我们需要进行手动分析操作。DLL可以在命令行上执
根据示例的体系结构，语法为：

```
<DYNRIO_DIR>\bin<ARCH>\drrun.exe -c "<PATH_TO_DLL>\ddr<ARCH>.dll" -s <START_ADDR> -e <END_ADDR> -c <NUM_INSTR_TO_EXECUTE> -f "
e.g.
C:\DYNRIO_DIR\bin64\drrun.exe -c "C:\ddr\ddr64.dll" -s 0x140001000 -e 0x140002200 -c 10000 -f "C:\ddrlog\sample_log64.json" --
C:\DYNRIO_DIR\bin32\drrun.exe -c "C:\ddr\ddr32.dll" -s 0x00401000 -e 0x00402000 -c 10000 -f "C:\ddrlog\sample_log32.json" -- s
```

分析完成后需要加载<JSON_FILE_TO_LOG_TO>，例如：通过IDA中的File/Load■■/Load DynRio File菜单加载sample_log32.json。

但同样，我们没有必要这么做。 DDR中的所有功能都可通过IDAs反汇编视图中的右键单击上下文菜单进行访问。
在运行DDR功能之前，用户需要先分析样本或手动加载JSON文件，如上所述。 如果用户不想进行手动分析过程，DDR会提供多种不同的选项来运行分析。
它们都可以通过图6中所示的Trace菜单访问。



在动态分析的过程中，系统会收集更多的运行时信息。它在执行的时候会消耗更多的内容并且需要更多的执行时间。追踪过程仅执行代码覆盖的，也就是说它在运行时执行一

请记住，所有DDR功能都使用我们运行的最后一个跟踪JSON文件。
例如，如果我们刚刚运行了一个程序并尝试通过"获取源操作数的值"来解析寄存器值，那么我们可能找不到任何数据（除非它是我们所提到的控制流指令之一，如调用
，jmp等）。 当我们第一次使用DDR时，我们需要查看生成的JSON文件，用以了解不同情况下的生成数据。

跟踪在样本所在的目录中的缓存，我们发现完整路径也可以在IDA日志窗口中找到。
这意味着，如果需要记录当前未加载的JSON文件中的信息，我们可以再次选择正确的跟踪菜单选项，并加载缓存的文件。
加载和解析文件通常不会花费太多，因此我们可以快速跳转到不同的分析地址，从而无需重新运行它们。
这也意味着，如果我们确实要重新进行某些分析，则必须通过"跟踪"菜单来删除所有缓存/保存的文件，或者从samples目录中手动删除相应的文件。

声明

Talos正在发布alpha版本，其中可能包含一些bug，不过其会在未来被修复。尽管如此，我们希望这是一个有用的工具，并在社区分享。

安装手册

该插件是针对Windows x64上的IDA7.2版本构建的，但也可能适用于7.1。

首先，在此处下载DDR存储库。

安装Python模块与DynamoRIO框架。详细信息可在下面的附录中找到。

接下来要做的是根据本地设置在"DDR_server.py"脚本中配置变量。此外，请确保本地防火墙不会阻止插件和服务器之间的通信。如果启动DDR_server.py脚本并且找不到"ddr_server.crt"，分析器（IDA / DDR_plugin.py）并将DDR_plugin.py中的CA_CERT■■指向它。之后我们可以自行设置API密钥和其他变量。下面是主要变量：</confdir>

DDR_plugin.py

运行主机ddr_server.py的IP地址：

```
WEBSERVER = "192.168.100.122"
```

DDRserver.py运行的TCP端口：

```
WEBSERVER_PORT = "5000"
```

API密钥，检查ddr_server.py启动消息以及jrdd_get.py脚本生成。

```
DDR_WEBAPI_KEY = "KT5LUFCHHSO12986OPZTWEFS"
```

本地目录用于查找由DDR_server.py脚本生成的证书或手动创建的证书（用于SSL连接）。 不要忘记将证书文件复制到此位置。

```
CA_CERT = r"C:\Users\User Name\Documents\idaplugin\ddr_server.crt"
```

验证证书。 将此设置为False是不安全的，在测试时进行操作即可。

```
VERIFY_CERT = True
```

ddr_server.py机器上的目录。 服务器上的本地目录，服务器脚本可以在其中找到要分析的样本。 确保它存在以及用户已将样本复制到其中。
该插件的未来版本将自动复制该文件。

```
SERVER_LOCAL_SAMPLE_DIR = r"C:\Users\User Name\Documents\DDR_samples"
```

第一次启动时生成自签名证书的参数和本地网络设置

```
CERT_FILE = "ddr_server.crt"
KEY_FILE = "ddr_server.key"
APIKEY_FILE = "ddr_apikey.txt"
MY_IPADDR = "192.168.100.122" # Malware Host IP addr
MY_PORT = "5000"
MY_FQDN = "malwarehost.local" # Malware host FQDN
```

用于保存与加载配置文件的目录 API密钥文件，证书文件等。

```
CONFDIR = r"C:\malware\tools\DDR_Talos\IDAplugin"
```

找到x32/x64 ddrun.exe和相应的DynRIO客户端DDR.dll。

```
CFG_DYNRIO_DRRUN_X32 = r"C:\tools\DynamoRIO-Windows-7.0.0-RC1\bin32\drrun.exe
CFG_DYNRIO_CLIENTDLL_X32 = r"C:\malware\tools\DDR_Talos\IDAplugin\ddr32.dll"
CFG_DYNRIO_DRRUN_X64 = r"C:\tools\DynamoRIO-Windows-7.0.0-RC1\bin64\drrun.exe"
CFG_DYNRIO_CLIENTDLL_X64 = r"C:\malware\tools\DDR_Talos\IDAplugin\ddr64.dll
```

警告信息

确保我们正在配置的目录是存在的。 如果它们不存在，则alpha版本将不会创建目录。 该程序将只显示一条错误消息。

此外，您必须首先将计划在IDA中分析的恶意软件样本复制到DDR_plugin.py脚本中SERVER_LOCAL_SAMPLE_DIR变量中配置的目录。
这将在下一版本中自动完成。

附录

Python Requirements

- Python27-x64

ddr_plugin.py/IDA machine (Analyst PC):

- 访问
  (http://docs.python-requests.org)

```
C:\python27-x64\Scripts>pip install -U requests
```

如果使用多个Python版本，请确保为IDA相同版本安装这些软件包。

ddr_server.py的系统需求 (Malware host):

- Flask
  (http://flask.pocoo.org/)
- PyOpenSSL
  (https://pyopenssl.org/en/stable/)

```
e.g.
pip install -U Flask
pip install -U pyOpenSSL
```

其他需求

ddr_server.py machine (Malware host):

- DynamoRIO Framework (https://www.dynamorio.org/)

只需使用DynamoRIO主页上的二进制安装程序即可。

测试环境：

ddr_plugin.py/IDA (Analyst PC - Windows 10 64bit):

IDA Version 7.2.181105 Windows x64

C:\Python27-x64\Scripts\pip.exe freeze

```
certifi==2017.7.27.1
chardet==3.0.4
first-plugin-ida==0.1.1
idna==2.6
requests==2.18.4
requests-kerberos==0.11.0
urllib3==1.22
winkerberos==0.7.0
yara==1.7.7
```

ddr_server.py machine(Malware host - Windows 7 64 bit):

C:\Python27-x64\Scripts\pip.exe freeze

```
asn1crypto==0.24.0
certifi==2018.11.29
cffi==1.11.5
chardet==3.0.4
Click==7.0
cryptography==2.4.2
enum34==1.1.6
Flask==1.0.2
idna==2.7
ipaddress==1.0.22
itsdangerous==1.1.0
Jinja2==2.10
MarkupSafe==1.1.0
pycparser==2.19
pyOpenSSL==18.0.0
requests==2.20.1
six==1.11.0
urllib3==1.24.1
Werkzeug==0.14.1
yara-python==3.6.3
```

DynamoRIO安装：

DynamoRIO Version: 7.0.0-RC1

安装目录: C:\tools\DynamoRIO-Windows-7.0.0-RC1

■■■■■■■■■■■■https://blog.talosintelligence.com/2019/01/ddr.html

点击收藏 | 0 关注 | 1

上一篇：某wind 9.0.2 任意文件删... 下一篇：Hitcon-Training l...

1. 0 条回复
   - 动动手指，沙发就是你的了！