Python恶意软件分析入门

Stefano / 2018-09-01 23:44:58 / 浏览数 3364 安全技术 技术讨论 顶(0) 踩(0)

原文：https://malwology.com/2018/08/24/python-for-malware-analysis-getting-started/

介绍

提升你的 Python 编程技巧可能正在你的待做清单上——就行清理衣橱、粉刷墙壁、拧紧螺丝一样（你知道我在说什么）。

通常来说，脚本是跨大多数安全学科的一个实用的工具，编写脚本可以帮助你自动完成琐碎的任务，规模化分析大量数据，分担你的工作量。

尽管有许多的编程语言可供选择，但 Python
是其中最受欢迎的那一个，抛开别的原因不谈，它是跨平台的，并且相对容易阅读和编写。许多现有的开源安全工具都是用 Python 写的，所以学习 Python
可以帮助你更好的理解这些工具。

这篇文章介绍了用于 PE 文件分析的 Python
程序，在下文中，一个脚本可以帮助你快速解析单个文件并提取关键特征，或在多个文件之间规模化测量以完成更高优先级的任务。

请注意，这篇文章假定读者已经对Python和编程的概念有了一些基本的了解。

已有的工具

现在已经有许多使用 Python 编写的恶意软件分析工具可供使用。以下是一些我觉得对静态文件分析比较有帮助的工具：

- pyew
- AnalyzePE
- pescanner
- peframe
- pecheck (在之前的文章有提到过)

这些工具可以产生有用的输出，是理解 Python
的绝佳选择。通过简单的浏览源代码和进行一些必要的搜索，你可以了解到作者是怎么编写这个程序的，和怎么修改程序以满足自己的目的。在你进行技术分析的过程中，你
Python 库来提取数据，把输出修改成你想要的形式。

pefile是一个流行的、存在已久的、用于分析PE文件的的库。这个模块提供了PE 文件组织的便捷入口。另一个最近更新的更通用的且跨平台的一个库叫做 Library
to Instrument Executable Formats (LIEF)，它包含了一个用于 PE 文件分析的Python 模块(文档在这)

这篇文章会用到Python2和 `pefile` 库来进行文件分析。请注意`pefile`是一个第三方模块，不是 Python
标准内置库。因此，你可能需要先安装它，尝试运行`pip install pefile`

探索 pefile

我们使用REMnux恶意软件分析 Linux 版（下载地址）。通过 Python 交互 shell 探索 pefile 模块，并编写一些简单的代码。相较于直接编写一个脚本，交互 shell
在用于学习可用模块和简单调试的时候是一个更好的选择。只需要在终端敲下`Python`，然后你就会看见如下的提示：

```
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

下一步，引入pefile 以使用它的函数：

```
>>> import pefile
>>>
```

可以通过帮助信息来了解这个模块。键入`help(pefile)`。下面是一部分的输出：

```
NAME
    pefile - pefile, Portable Executable reader module

FILE
    /usr/lib/python2.7/dist-packages/pefile.py

DESCRIPTION

    All the PE file basic structures are available with their default names
    as attributes of the instance returned.

    Processed elements such as the import table are made available with lowercase
    names, to differentiate them from the upper case basic structure names.

    pefile has been tested against the limits of valid PE headers, that is, malware.
    Lots of packed malware attempt to abuse the format way beyond its standard use.
    To the best of my knowledge most of the abuses are handled gracefully.

    Copyright (c) 2005-2013 Ero Carrera <ero.carrera@gmail.com>

    All rights reserved.

    For detailed copyright information see the file COPYING in
    the root of the distribution archive.

CLASSES
    __builtin__.object
        DataContainer
```

除了模块的概述以外，还有类的概述。往下活动能看到每个类的信息，我们关注到 PE 类：

```
class PE
 |  A Portable Executable representation.
 |
 |  This class provides access to most of the information in a PE file.
 |
 |  It expects to be supplied the name of the file to load or PE data
 |  to process and an optional argument 'fast_load' (False by default)
 |  which controls whether to load all the directories information,
 |  which can be quite time consuming.
 |
 |  pe = pefile.PE('module.dll')
 |  pe = pefile.PE(name='module.dll')
 |
 |  would load 'module.dll' and process it. If the data would be already
 |  available in a buffer the same could be achieved with:
 |
 |  pe = pefile.PE(data=module_dll_data)
```

描述说我们可以用这个类来访问 PE文件的结构，我们需要用它来分析 Windows 文件。输出也描述了怎么创建一个 PE 类的实例。让我们来动手试一下。在这篇文章中，我们使用一个[emotet样本]。

```
>>> import pefile
>>> file = pefile.PE("sample.exe")
>>>
```

可以回到帮助信息那看一下 PE 类有哪些方法和属性。或者输入`dir(pefile.PE)`来查看。部分输出如下：

```
['__IMAGE_BASE_RELOCATION_ENTRY_format__', '__IMAGE_BASE_RELOCATION_format__', '__IMAG
MAGE_BOUND_IMPORT_DESCRIPTOR_format__', '__IMAGE_DATA_DIRECTORY_format__', '__IMAGE_DE
AY_IMPORT_DESCRIPTOR_format__', '__IMAGE_DOS_HEADER_format__', '__IMAGE_EXPORT_DIRECTO
rmat__', '__IMAGE_IMPORT_DESCRIPTOR_format__', '__IMAGE_LOAD_CONFIG_DIRECTORY64_format
ormat__', '__IMAGE_NT_HEADERS_format__', '__IMAGE_OPTIONAL_HEADER64_format__', '__IMAG
_RESOURCE_DATA_ENTRY_format__', '__IMAGE_RESOURCE_DIRECTORY_ENTRY_format__', '__IMAGE_
E_SECTION_HEADER_format__', '__IMAGE_THUNK_DATA64_format__', '__IMAGE_THUNK_DATA_forma
__', '__IMAGE_TLS_DIRECTORY_format__', '__StringFileInfo_format__', '__StringTable_for
XEDFILEINFO_format__', '__VS_VERSIONINFO_format__', '__Var_format__', '__doc__', '__in
str__', '__unpack_data__', 'adjust_FileAlignment', 'adjust_SectionAlignment', 'close',
gn', 'full_load', 'generate_checksum', 'get_data', 'get_data_from_dword', 'get_data_fr
_dword_at_rva', 'get_dword_from_data', 'get_dword_from_offset', 'get_imphash', 'get_im
', 'get_offset_from_rva', 'get_overlay', 'get_overlay_data_start_offset', 'get_physica
word_from_data', 'get_qword_from_offset', 'get_resources_strings', 'get_rva_from_offse
tion_by_rva', 'get_string_at_rva', 'get_string_from_data', 'get_string_u_at_rva', 'get
ord_from_data', 'get_word_from_offset', 'is_dll', 'is_driver', 'is_exe', 'merge_modifi
```

可以看到有许多字，而且大多数不依赖于你之前对 PE
文件的分析。来看一些我们可能认识的专业术语。看到许多以"get_"开头的函数，它们可以帮助收集一些关于文件的静态信息。比如get_impash()返回一个IAT
的 MD5值。用我们的file实例试一下：

```
>>> file.get_imphash()
'41ae639fb296a73ebf2f7291a15a96f9'
>>>
```

get_imphash()如期的提供了 IAT 的 MD5值

另一个有用的"get_"函数是get_warnings。当pefile解析一个 Windows 可执行文件时，可能会发生错误。get_warnings返回一个 PE
文件在被处理时产生的警告。安全性分析就是为了调查异常的，所以这个输出可以揭示用于进一步分析的着手点。比如说，这个函数的输出可以指示出文件是被混淆过的，
或者 PEid）。在这种情况下，函数不会返回错误：

```
>>> file.get_warnings()
[]
>>>
```

继续往下，提取恶意软件分析时常用的其他静态信息。比如说，使用pefile查看可执行文件导入了哪些 DLL 文件和函数。为了解决这个问题，再次使用内建的
help()进行一些老式的尝试和报错。此方法适用于任何文档详细的 Python 模块。

首先，查阅 PE 类以检查我们的选项。键入help(pefile.PE)，然后往下看。感兴趣的是这一部分：

```
The sections will be available as a list in the 'sections' attribute.
Each entry will contain as attributes all the structure's members.

Directory entries will be available as attributes (if they exist):
(no other entries are processed at this point)

DIRECTORY_ENTRY_IMPORT (list of ImportDescData instances)
DIRECTORY_ENTRY_EXPORT (ExportDirData instance)
DIRECTORY_ENTRY_RESOURCE (ResourceDirData instance)
DIRECTORY_ENTRY_DEBUG (list of DebugData instances)
DIRECTORY_ENTRY_BASERELOC (list of BaseRelocationData instances)
DIRECTORY_ENTRY_TLS
DIRECTORY_ENTRY_BOUND_IMPORT (list of BoundImportData instances)
```

可以看到许多"DIRECTORY_ENTRY_"属性的引用，这些属性指向密钥文件组件的位置。因为我们对导入表感兴趣，所以关注DIRECTORY_ENTRY_IMPORT，它被描述为Imp
实例的一个列表。通过迭代这个列表看看他提供什么信息：

```
>>> for item in file.DIRECTORY_ENTRY_IMPORT:
...     print item
...
<pefile.ImportDescData object at 0x7f5ae64d88d0>
<pefile.ImportDescData object at 0x7f5ae64d8a90>
<pefile.ImportDescData object at 0x7f5ae64d8d90>
<pefile.ImportDescData object at 0x7f5ae64d8f10>
<pefile.ImportDescData object at 0x7f5ae64e80d0>
<pefile.ImportDescData object at 0x7f5ae64e8310>
<pefile.ImportDescData object at 0x7f5ae64e8490>
<pefile.ImportDescData object at 0x7f5ae64e8610>
<pefile.ImportDescData object at 0x7f5ae64e87d0>
<pefile.ImportDescData object at 0x7f5ae64e8950>
>>>
```

就像帮助信息里说明的一样，可以看到一个关于ImportDescData对象的列表。这些对象又代表什么呢？再次使用`help(pefile.ImportDescData)`：

```
class ImportDescData(DataContainer)
 |  Holds import descriptor information.
 |
 |  dll:        name of the imported DLL
 |  imports:    list of imported symbols (ImportData instances)
 |  struct:     IMAGE_IMPORT_DESCRIPTOR structure
```

如上所示，这个结构体包含了 DLL 的名字和一个已导入的符号表。这貌似就是我们想要的，再次迭代进行确认：

```
>>> for item in file.DIRECTORY_ENTRY_IMPORT:
...     print item.dll
...     for i in item.imports:
...             print i
...
KERNEL32.dll
<pefile.ImportData object at 0x7f5ae64d8510>
<pefile.ImportData object at 0x7f5ae64d87d0>
<pefile.ImportData object at 0x7f5ae64d8810>
<pefile.ImportData object at 0x7f5ae64d8850>
<pefile.ImportData object at 0x7f5ae64d8890>
VERSION.dll
<pefile.ImportData object at 0x7f5ae64d8a50>
WinSCard.dll
<pefile.ImportData object at 0x7f5ae64d8910>
```

我们取得了不错的进展，但现在有一个新的结构体需要研究一下。键入`help(pefile.ImportData)`

```
class ImportData(DataContainer)
 |  Holds imported symbol's information.
 |
 |  ordinal:    Ordinal of the symbol
 |  name:       Name of the symbol
 |  bound:      If the symbol is bound, this contains
 |              the address.
```

目前为止，我们只关注导入的名字，所以名字属性应该有我们想要的信息。将它放入到代码中，使输出更具有可读性。

```
>>> for item in file.DIRECTORY_ENTRY_IMPORT:
...     print item.dll
...     for i in item.imports:
...             print i.name
...     print("=====")
...
KERNEL32.dll
GlobalDeleteAtom
GetCommandLineW
SetFileApisToANSI
HeapFree
GetBinaryTypeW
=====
VERSION.dll
GetFileVersionInfoSizeW
=====
WinSCard.dll
SCardDisconnect
SCardConnectA
SCardTransmit
=====
```

成功了！代码为我们提供了导入的DLL的名称及其对应的导入函数名称。可以把输出变得更好看点，但是我们需要的信息已经全都有了。

规模化

像之前在介绍里说的一样,使用脚本自动完成工作可以使你规模化分析大量数据。单文件分析固然重要，但是日重工作中的恶意软件分析可能需要你分析成百上千个文件，再

另外再考虑文件的 imphash。在大量的文件中，通过
imphash可以更容易的鉴别出类似的函数或者用于产生二进制文件的公共打包器/打包工具。代码应该完成下列任务：

1. Create a list of all files in the directory (full path).
2. Open an XLSX file for writing (I often use Excel for easy viewing/sorting, but you can certainly output to CSV or, even better, write this information to a database).
3. Calculate and write each file's sha256 hash and imphash to the XLSX file.
4. Autofilter the data.

下面是完成这些任务的一种方法。

```
#~/usr/bin/env python
import sys,os
import pefile
import hashlib
import xlsxwriter

if __name__ == "__main__":

   #Identify specified folder with suspect files
   dir_path = sys.argv[1]

   #Create a list of files with full path
   file_list = []
   for folder, subfolder, files in os.walk(dir_path):
       for f in files:
```

```
            full_path = os.path.join(folder, f)
            file_list.append(full_path)

    #Open XLSX file for writing
    file_name = "pefull_output.xlsx"
    workbook = xlsxwriter.Workbook(file_name)
    bold = workbook.add_format({'bold':True})
    worksheet = workbook.add_worksheet()

    #Write column headings
    row = 0
    worksheet.write('A1', 'SHA256', bold)
    worksheet.write('B1', 'Imphash', bold)
    row += 1

    #Iterate through file_list to calculate imphash and sha256 file hash
    for item in file_list:

        #Get sha256
                fh = open(item, "rb")
                data = fh.read()
                fh.close()
                sha256 = hashlib.sha256(data).hexdigest()

        #Get import table hash
        pe = pefile.PE(item)
        ihash = pe.get_imphash()

        #Write hashes to doc
        worksheet.write(row, 0, sha256)
        worksheet.write(row, 1, ihash)
        row += 1

    #Autofilter the xlsx file for easy viewing/sorting
    worksheet.autofilter(0, 0, row, 2)
    workbook.close()
```

我把这个脚本命名为`pe_stats.py`，并在“suspect_files”目录中使用命令`python pe_stats.py suspectfiles`来运行它。为了填充目标目录，我从VT下载了100个高危文件（说明一下，我使用了基础 VTI 查询："type:peexe positives:50+"）。在 Excel 中打开输出数据，部分输出如下。

SHA256	Imphash
058cdcfee3e4c8b02cd22c43e2fc7f51527d26a7c9592b4330ffb894fa1caac8	09d0478591d4f788cb3e5ea416c25237
e70be2a962e0e34afc84c91d8590652b7fc42b017aeabbfed50a16341761c5a2	09d0478591d4f788cb3e5ea416c25237
78e51957bcf2f2b03454c8037b465b7e66ba60c2ccb4053d761afc163a35752f	09d0478591d4f788cb3e5ea416c25237
360dfa01504d0340366f9bad845cedfce3610c6f57ca6a9d5561fa14cc064042	09d0478591d4f788cb3e5ea416c25237
c8985e9b5f224a1889cc935df8a3bd895b5ade99875a6460daceae35abb80b7d	120aa51067095eec3023188f73fd5272
027ebae0e39d21f67f9f9d4078f801f2427d490be3c37e06667ff45dd6c12e62	1deedc28ba61d6a4bfbb6365a546a262
40dd5a3dac65852f26c34d557e5d9fe7dac9e2485cd59451e60d8d0516c09ae4	1deedc28ba61d6a4bfbb6365a546a262
ee4a9c73db16719240199a6784bbd1eb62f10df1374993d74084e57c541dd8e2	222e7b320f36011feb1642000d8fa826
2500b807f30d3b12877f546ce17c4816109e4218eb162323cd177e9bb26f0f34	222e7b320f36011feb1642000d8fa826
2d7fbeeaf6921802197c45297932f3fe1d69924d414573cc08b690f6da58b449	22cb62e7fd0bc3bd42f99f6676b40340
7728e2b78fde0c51427c67fdb85e51a654d7d51c2cfdd806cccae4c9400aef1b	28178deeb23ca335978bbb93418aba95
620a5293c6d6080b5a08040b817e6fa101a53c018dc73817b06db19811198fe7	28178deeb23ca335978bbb93418aba95
23f30167007583678 27de370c646cecfefac4552120de4b8bcbbaef3c7200eb4	28178deeb23ca335978bbb93418aba95
fdc01646c0f6fb664b0a01cdc9fe93d4d50fd014c59c98346b9a41a3d04c9c07	28178deeb23ca335978bbb93418aba95
d3228bb55000ccc2dde94734e2e8552ae536a5bdadd00d4e921ae4b3f77203a2	28178deeb23ca335978bbb93418aba95
bac118fc05c3228fe4927b40de4fd4700c775fe74c38092540fef64781cd6a06	28178deeb23ca335978bbb93418aba95

看一眼前几行就可以发现 imphash 有一个特定的模式。下一步，可以研究一下最大的导入表哈希集群，以了解为什么文件具有相同的imphash。你也可以再看一下`pefile`库的文档，研究一下表中其他的静态特性有什么用。有了这些的细节，会对样本分类和优先级分析有很大的帮助。我把这些作为任务留

总结

这篇文章提供了使用 python 分析 PE 文件的一个方法。更重要的是，它介绍了如果使用 Python 的内置函数 `help`以及一些PE文件的基础知识来系统化分析文件的特征，然后将其扩大到更大的文件集。

如果你想了解更多关于恶意软件分析的策略，你可以加入 即将开始的 SANS FOR610 课程以了解更多。

点击收藏 | 1 关注 | 1

上一篇：网鼎杯第四场shenyue2-wp 下一篇：ECShop全系列版本远程代码执行...

1. 0 条回复

- 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板