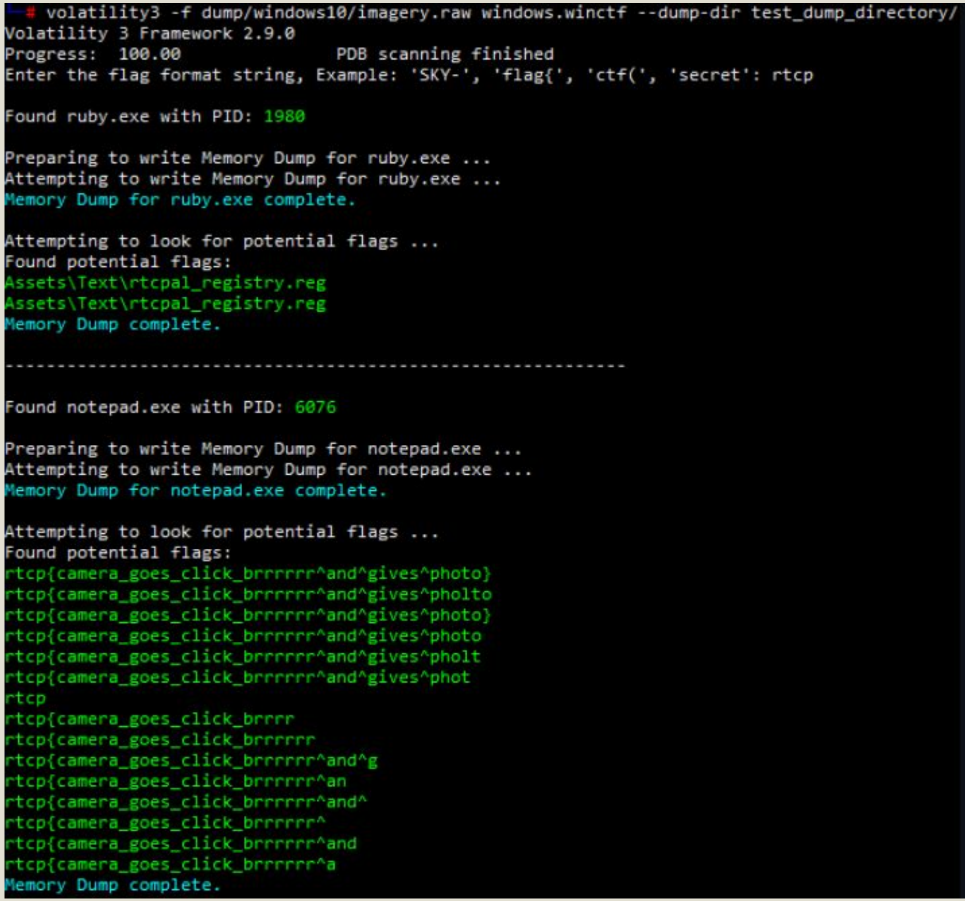
03.12.2024 https://github.com/Liabell/volatility-plugin.git

# WINCTF

## A VOLATILITY3 WINDOWS PLUGIN



```
└─# volatility3 -f dump/windows10/imagery.raw windows.winctf --dump-dir test_dump_directory/
Volatility 3 Framework 2.9.0
Progress:  100.00               PDB scanning finished
Enter the flag format string, Example: 'SKY-', 'flag{', 'ctf(', 'secret': rtcp

Found ruby.exe with PID: 1980

Preparing to write Memory Dump for ruby.exe ...
Attempting to write Memory Dump for ruby.exe ...
Memory Dump for ruby.exe complete.

Attempting to look for potential flags ...
Found potential flags:
Assets\Text\rtcpal_registry.reg
Assets\Text\rtcpal_registry.reg
Memory Dump complete.

------------------------------------------------------------

Found notepad.exe with PID: 6076

Preparing to write Memory Dump for notepad.exe ...
Attempting to write Memory Dump for notepad.exe ...
Memory Dump for notepad.exe complete.

Attempting to look for potential flags ...
Found potential flags:
rtcp{camera_goes_click_brrrrrr^and^gives^photo}
rtcp{camera_goes_click_brrrrrr^and^gives^pholto
rtcp{camera_goes_click_brrrrrr^and^gives^photo}
rtcp{camera_goes_click_brrrrrr^and^gives^photo
rtcp{camera_goes_click_brrrrrr^and^gives^pholt
rtcp{camera_goes_click_brrrrrr^and^gives^phot
rtcp
rtcp{camera_goes_click_brrrr
rtcp{camera_goes_click_brrrrrr
rtcp{camera_goes_click_brrrrrr^and^g
rtcp{camera_goes_click_brrrrrr^an
rtcp{camera_goes_click_brrrrrr^and^
rtcp{camera_goes_click_brrrrrr^
rtcp{camera_goes_click_brrrrrr^and
rtcp{camera_goes_click_brrrrrr^a
Memory Dump complete.
```

## Goal:

**Create a plugin that solves most basic forensics challenges present in CTF competitions**

## Features:

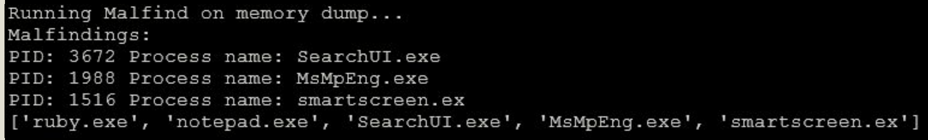
- Suspicious Process Detection
- Process Memory Dumping with Flag Format Analysis
- Hash Detection + Decryption
- Browser memory Dumping + Analysis

WINCTF

## Suspicious Process Detection

*Automatic Suspicious Process Detection using windows.malfind*

Solves:
“What is the suspicious process present in the memory dump?”



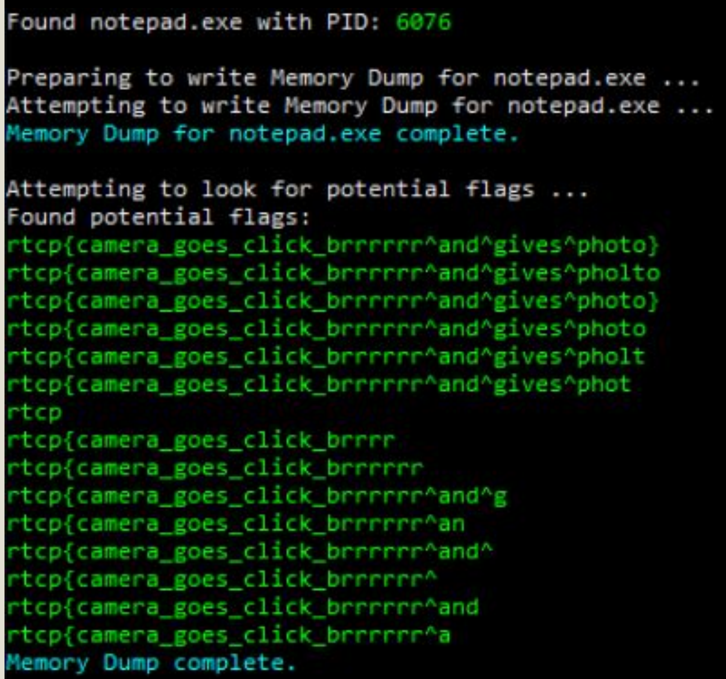
```
Running Malfind on memory dump...
Malfindings:
PID: 3672 Process name: SearchUI.exe
PID: 1988 Process name: MsMpEng.exe
PID: 1516 Process name: smartscreen.ex
['ruby.exe', 'notepad.exe', 'SearchUI.exe', 'MsMpEng.exe', 'smartscreen.ex']
```

WINCTF

## Process Memory Dumping Flag Format Analysis

*Automatic process detection and memory dumping with built in flag detection using custom flag formats. Works with both little endian and big endian memory formats.*

Solves: “What is the flag?”



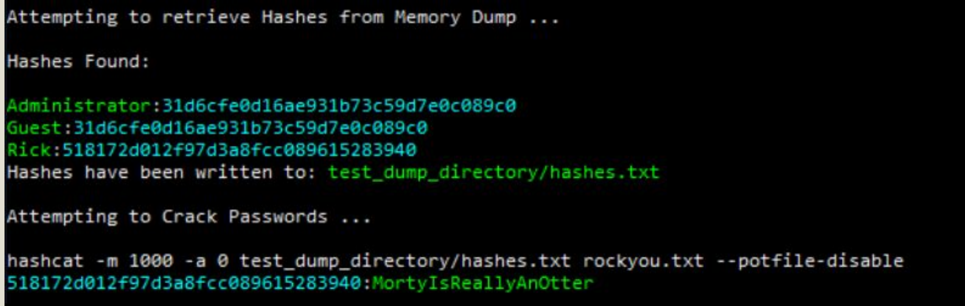
```
Found notepad.exe with PID: 6076

Preparing to write Memory Dump for notepad.exe ...
Attempting to write Memory Dump for notepad.exe ...
Memory Dump for notepad.exe complete.

Attempting to look for potential flags ...
Found potential flags:
rtcp{camera_goes_click_brrrrrr^and^gives^photo}
rtcp{camera_goes_click_brrrrrr^and^gives^pholto
rtcp{camera_goes_click_brrrrrr^and^gives^photo}
rtcp{camera_goes_click_brrrrrr^and^gives^photo
rtcp{camera_goes_click_brrrrrr^and^gives^pholt
rtcp{camera_goes_click_brrrrrr^and^gives^phot
rtcp
rtcp{camera_goes_click_brrrr
rtcp{camera_goes_click_brrrrrr
rtcp{camera_goes_click_brrrrrr^and^g
rtcp{camera_goes_click_brrrrrr^an
rtcp{camera_goes_click_brrrrrr^and^
rtcp{camera_goes_click_brrrrrr^
rtcp{camera_goes_click_brrrrrr^and
rtcp{camera_goes_click_brrrrrr^a
Memory Dump complete.
```

## Hash Detection + Decryption

*Automatic NTLM (windows) hash detection using windows.hashdump and decryption via Hashcat instance run as a subprocess*

Solves:
“What is the hash of <user>’s account?”
“What is the plaintext password of <user>’s account?”

(Administrator and Guest passwords are both empty strings in the example which is why they aren’t printed)



```
Attempting to retrieve Hashes from Memory Dump ...

Hashes Found:

Administrator:31d6cfe0d16ae931b73c59d7e0c089c0
Guest:31d6cfe0d16ae931b73c59d7e0c089c0
Rick:518172d012f97d3a8fcc089615283940
Hashes have been written to: test_dump_directory/hashes.txt

Attempting to Crack Passwords ...

hashcat -m 1000 -a 0 test_dump_directory/hashes.txt rockyou.txt --potfile-disable
518172d012f97d3a8fcc089615283940:MortyIsReallyAnOtter
```

WINCTF

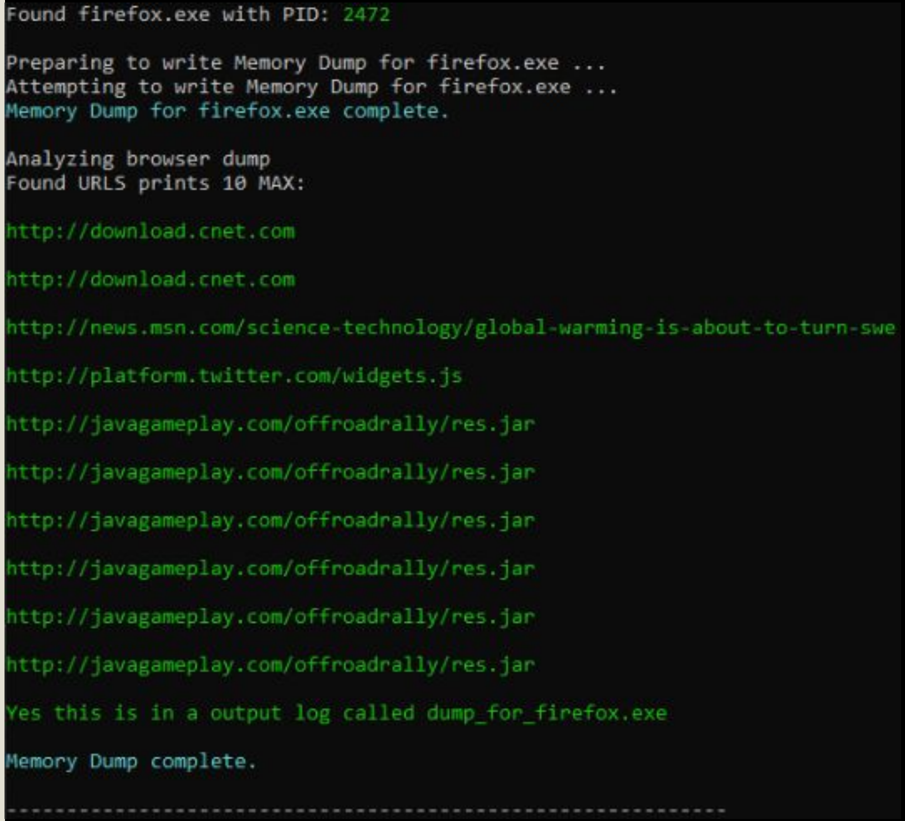
# Browser memory Dumping + Analysis

Automation of the detection and dumping of commonly used browsers:

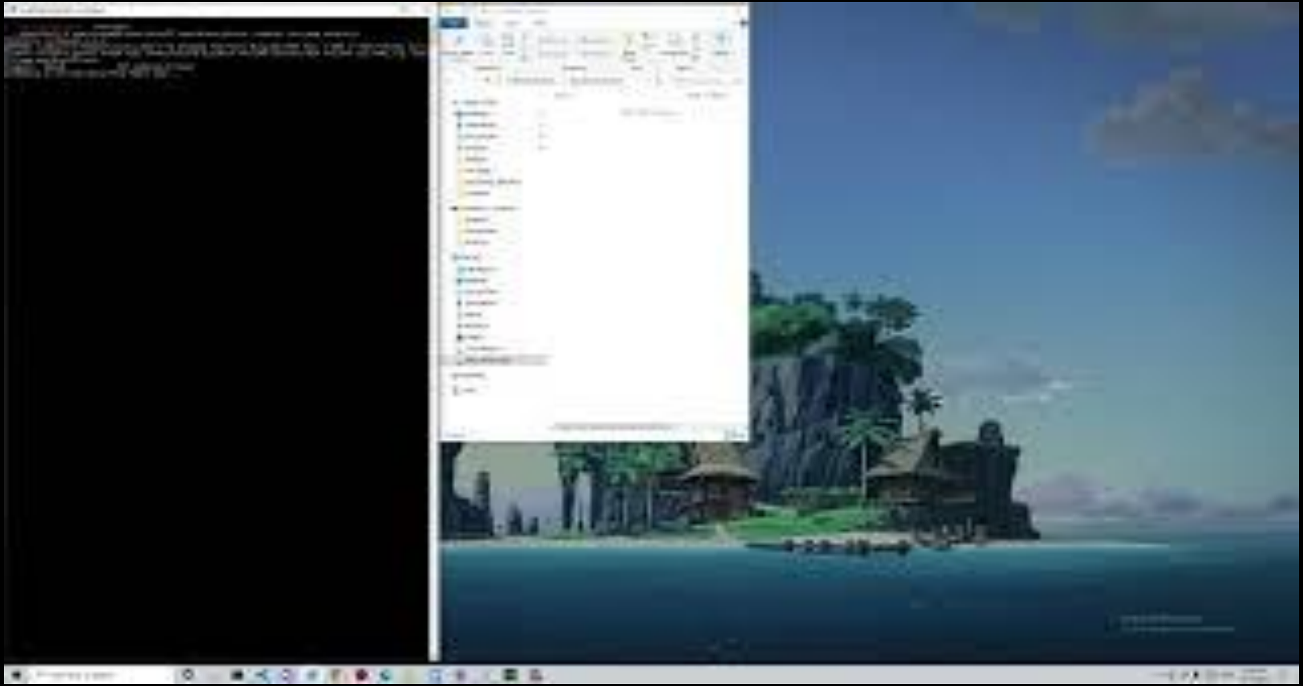
- firefox.exe
- MicrosoftEdge.exe
- chrome.exe

Memory dumps are run against strings to extract basic data that could be used to help guide a player an answer.

Solves:
“What website did <user> visit?”
“What file did <user> download”



```
Found firefox.exe with PID: 2472

Preparing to write Memory Dump for firefox.exe ...
Attempting to write Memory Dump for firefox.exe ...
Memory Dump for firefox.exe complete.

Analyzing browser dump
Found URLS prints 10 MAX:

http://download.cnet.com

http://download.cnet.com

http://news.msn.com/science-technology/global-warming-is-about-to-turn-swe

http://platform.twitter.com/widgets.js

http://javagameplay.com/offroadrally/res.jar

http://javagameplay.com/offroadrally/res.jar

http://javagameplay.com/offroadrally/res.jar

http://javagameplay.com/offroadrally/res.jar

http://javagameplay.com/offroadrally/res.jar

http://javagameplay.com/offroadrally/res.jar

Yes this is in a output log called dump_for_firefox.exe

Memory Dump complete.

------------------------------------------------------------
```



https://www.youtube.com/watch?v=T5b0ohbBObw

# DEMO

# THANK YOU

Try the plugin for yourself at:

https://github.com/Liabell/volatility-plugin.git

## Test Dumps Used:

Test Flag Finder functionality (try flag 'rtcp'): imagery.raw (Windows10): https://drive.google.com/file/d/1y4sfIaUrAOK0wXiDZXiOI-q2SYs6M--g/view

Test Password Cracking functionality (try flag 'hackflags'): OtterCTF.vmem (Windows 10): https://mega.nz/#!sh8wmCIL!b4tpech4wzc3QQ6YgQ2uZnOmctRZ2duQxDqxbkWYipQ

Test  browser memory analysis functionality: voltest.dmp (Windows 10): http://www.superponible.com/volatility/voltest.zip