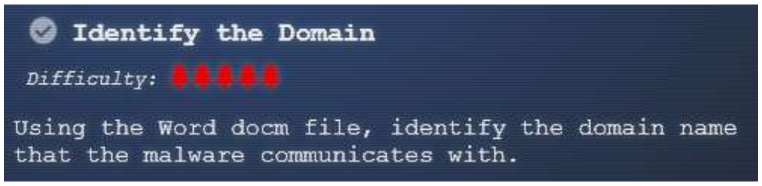
Objective--Identify the Domain (Part 1)

What you can learn from this

This objective is the first of three involved with reverse-engineering malware written in PowerShell. A true Linux person may disdain a language written for Windows, but there are good reasons to learn PowerShell. About 80% of the attacker's targets are Windows, and all recent versions of Windows come with PowerShell installed by default. If attackers want to "live off the land," what better way is there for them to do it but to write their malware in PowerShell? Chris Davis' talk on Analyzing PowerShell Malware is a must for this challenge. He will lead you through extracting PowerShell malware that is embedded in a Word document, "prettifying" the malware, and some basic troubleshooting using the PowerShell Integrated Scripting Environment (ISE).

Getting Started

The objective is to identify the domain that the malware connects with.

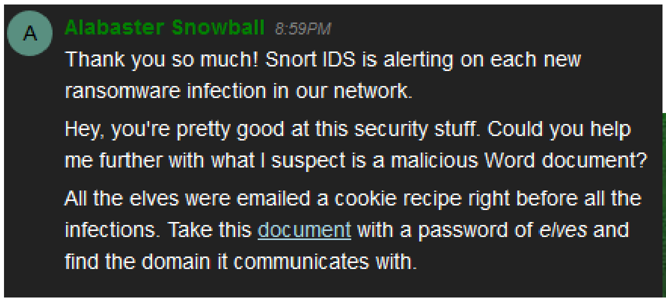


Identify the Domain

Difficulty:

Using the Word docm file, identify the domain name that the malware communicates with.

Alabaster had these hints to give after you solved his Snort problem. The link he gives is to a malicious Word document.

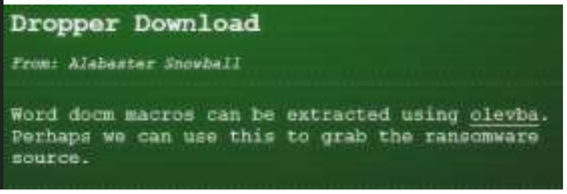


Alabaster Snowball 8:59PM

Thank you so much! Snort IDS is alerting on each new ransomware infection in our network.

Hey, you're pretty good at this security stuff. Could you help me further with what I suspect is a malicious Word document?

All the elves were emailed a cookie recipe right before all the infections. Take this document with a password of *elves* and find the domain it communicates with.



Dropper Download

From: Alabaster Snowball

Word docm macros can be extracted using olevba. Perhaps we can use this to grab the ransomware source.

A Word of caution

CounterHack Challenges and SANS have kindly given us simulated malware to play with that will not harm our computers. Never the less, this would be a good time to practice the Operations Security (OPSEC) that Chris discussed in his talk. It would be wise to do all the work on this malware in a Windows VM, not on your host computer. In fact, Windows Defender detects the Word document in chocolate_chip_recipe.zip as malware as soon as you unzip it. You will probably need to disable Windows Defender on your VM.

Steps

1) Follow Chris' instructions to extract the malware from the Word document using olevba.exe.
2) Use PowerShell to decode the dropper, again following Chris' instructions.
3) Copy the decoded dropper into PowerShell ISE or Visual Code and clean it up. (This is an extra step; Chris ran the dropper directly from PowerShell.)
4) Study the dropper to determine how it works.
5) Start a packet capture and execute the dropper.

Hand in

1) Turn in a screenshot of your cleaned version of the dropper script.
2) Roughly, how does the dropper work? H2A is a function that converts a hex string to ASCII; you don't need to discuss H2A.

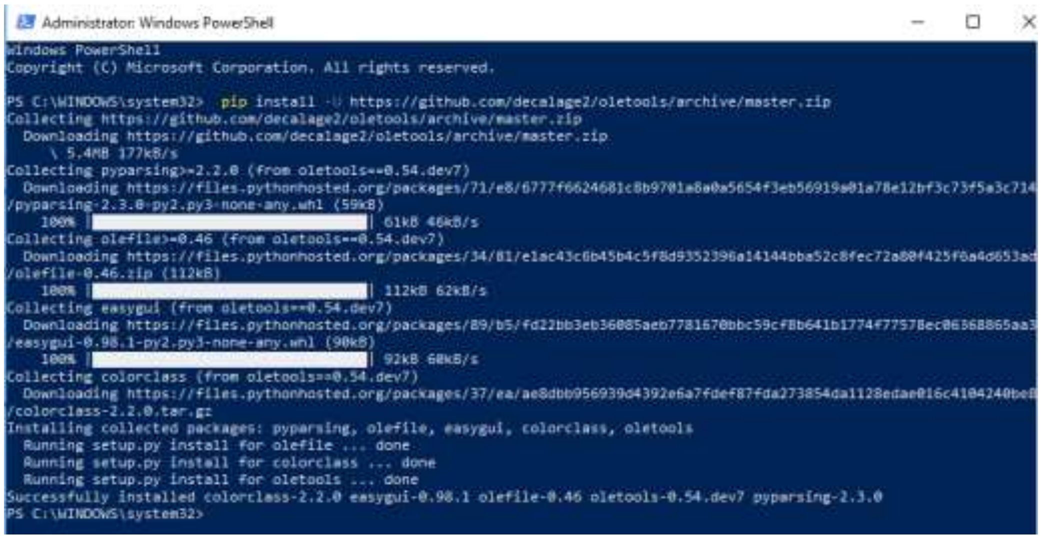
One note: If you elect to clean the malware (not just the dropper), and remove all semicolons the way Chris did, you will find that there are a few old-style FOR loops in the code that use semicolons. You will need to put those semicolons back.

Notes on Installing olevba.exe

Important note: If your machine is running Python 3, you need to use olevba3.exe.

The application Chris used to extract the malware from the Word document is Python based. Some versions of Windows 10 make Python available from the PowerShell prompt, but others do not. If you do not have Python in your version of Windows, you can download it here. Python's package manager, PIP, is now installed along with Python. You can use PIP to install oletools (olevba is one of the tools) using:

```
pip install -U
https://github.com/decalage2/oletools/archive/master.zip
```

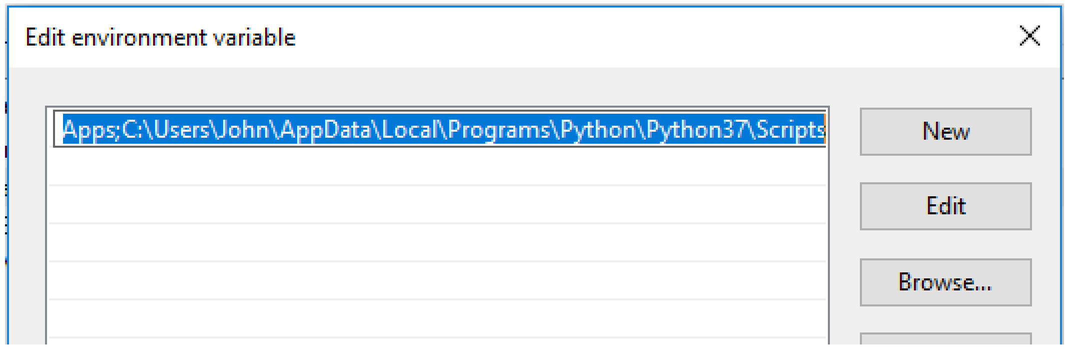


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\WINDOWS\system32>  pip install -U https://github.com/decalage2/oletools/archive/master.zip
Collecting https://github.com/decalage2/oletools/archive/master.zip
  Downloading https://github.com/decalage2/oletools/archive/master.zip
     \ 5.4MB 177kB/s
Collecting pyparsing>=2.2.0 (from oletools==0.54.dev7)
  Downloading https://files.pythonhosted.org/packages/71/e8/6777f6624681c8b9701a8a0a5654f3eb56919a01a78e12bf3c73f5a3c714
/pyparsing-2.3.0-py2.py3-none-any.whl (59kB)
    100% |████████████████████████████████| 61kB 46kB/s
Collecting olefile>=0.46 (from oletools==0.54.dev7)
  Downloading https://files.pythonhosted.org/packages/34/81/e1ac43c6b45b4c5f8d9352396a14144bba52c8fec72a80f425f6a4d653ad
/olefile-0.46.zip (112kB)
    100% |████████████████████████████████| 112kB 62kB/s
Collecting easygui (from oletools==0.54.dev7)
  Downloading https://files.pythonhosted.org/packages/89/b5/fd22bb3eb36085aeb7781670bbc59cf8b641b1774f77578ec06368865aa3
/easygui-0.98.1-py2.py3-none-any.whl (90kB)
    100% |████████████████████████████████| 92kB 68kB/s
Collecting colorclass (from oletools==0.54.dev7)
  Downloading https://files.pythonhosted.org/packages/37/ea/ae8dbb956939d4392e6a7fdef87fda273854da1128edae016c4104240be8
/colorclass-2.2.0.tar.gz
Installing collected packages: pyparsing, olefile, easygui, colorclass, oletools
  Running setup.py install for olefile ... done
  Running setup.py install for colorclass ... done
  Running setup.py install for oletools ... done
Successfully installed colorclass-2.2.0 easygui-0.98.1 olefile-0.46 oletools-0.54.dev7 pyparsing-2.3.0
PS C:\WINDOWS\system32>
```

The site for oletools is here, and here for the olevba tool.

To make life easier for myself, I added the paths for Python and PIP to my environment PATH variable. On my machine they were
`C:\Users\John\AppData\Local\Programs\Python\Python37` and
`C:\Users\John\AppData\Local\Programs\Python\Python37\Scripts`



Apps;C:\Users\John\AppData\Local\Programs\Python\Python37\Scripts

%USERPROFILE%\AppData\Local\Microsoft\WindowsApps;C:\Users\John\AppData\Local\Programs\Python\Python37;C:\Users\John\AppData\Local\Programs\Python\Python37\Scripts