Conducting Malware Analysis with Python



Cristian Pascariu
Information Security Professional

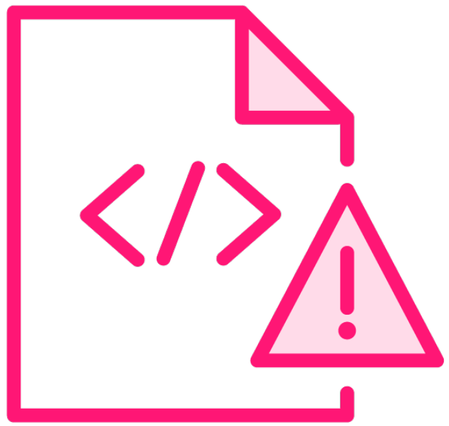
www.cybersomething.com

Overview

Analyze malicious documents

Detect packed executables

Analyzing Malicious Documents



Weaponized VBA macros

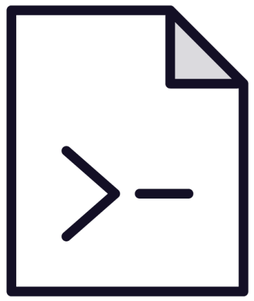
Many companies built entire solutions on macros

- Microsoft plans to block execution by default

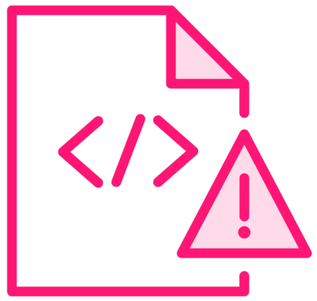
Analyzing Malicious Documents



Identified document with python-magic

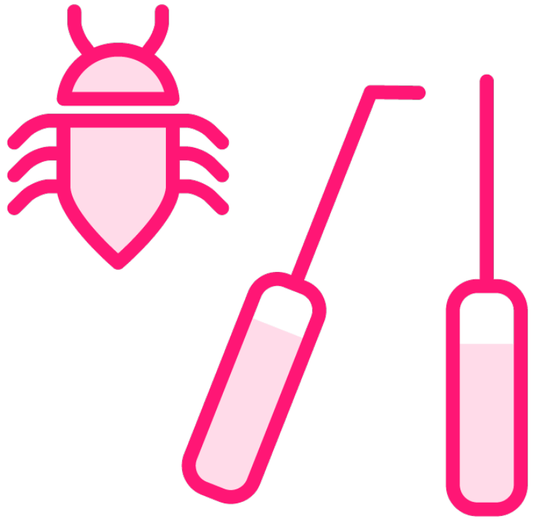


Identified self-executing macros using Yara rules



Analyze suspicious documents using Olevba

Analyzing Malicious Documents



Oletools is a set of tools for analyzing malicious documents

- https://github.com/decalage2/oletools

Part of detection solutions

Olevba can detect and extract VBA macros

- Can detect malicious indicators as well

Review docs for the oletools module

Analyze a malicious document

- Extract indicators

Analyzing Packed Executables

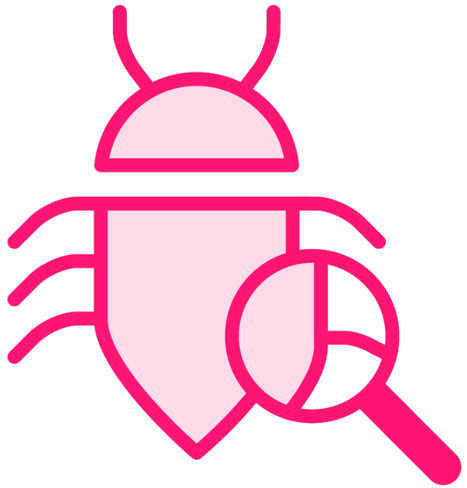


Malware authors attempt evade detection and hinder analysis

- Malware packers

Once executed it unpacks in memory

Automating the Detection of Packed Malware



Signatures for packers codified into Yara rules

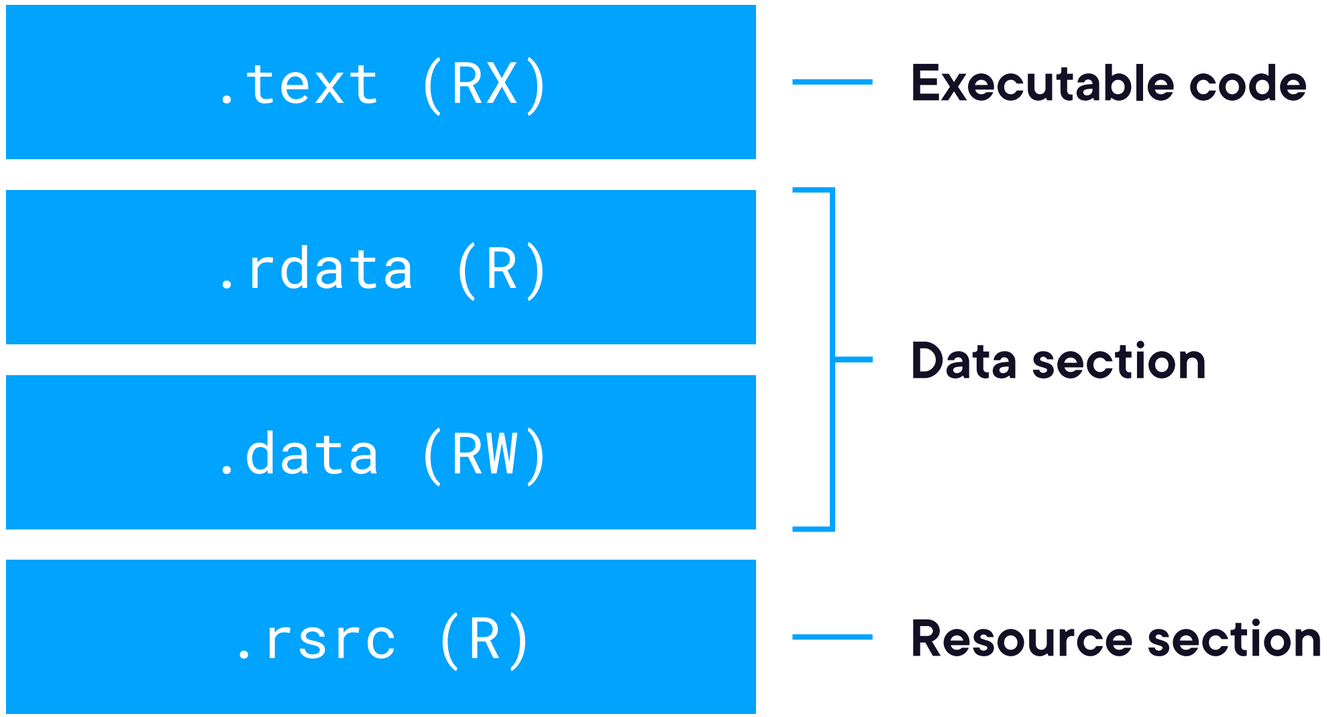
Low number of imports

- Only functions for unpacking are imported
- Detection is based on counting DLLs and functions

Sections marked RWX

- Unpacking involves reading, writing and executing the unpacked code
- Detection is based on section characteristics

Legitimate Sections



Suspicious Sections



Working with Flags

Section	Read	Write	Execute
.text	1	0	1
.rdata	1	0	0
.data	1	1	0

Each section contains a field for its characteristics

Working with Flags

Characteristics Mask for read

```
.text    101    &    100    !=    0
```

Bitwise AND

Working with Flags

	Characteristics		Mask for read		
.text	101	&	100	=	100

Bitwise AND

Section Flags

Flag	Value	Description
IMAGE_SCN_MEM_EXECUTE	0x20000000	The section can be executed as code.
IMAGE_SCN_MEM_READ	0x40000000	The section can be read.
IMAGE_SCN_MEM_WRITE	0x80000000	The section can be written to.

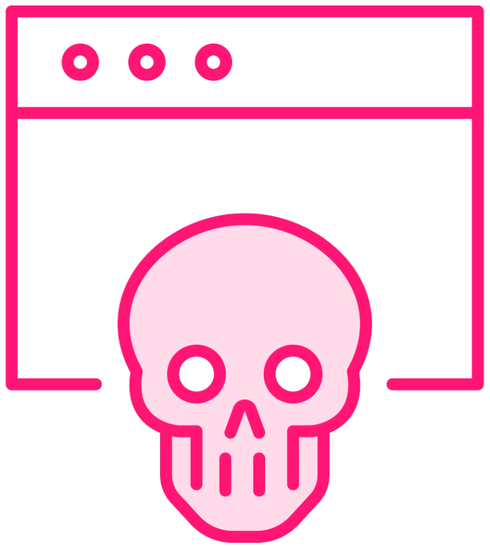
https://learn.microsoft.com/en-gb/windows/win32/debug/pe-format?redirectedfrom=MSDN#section-flags

Review pefile documentation

Identify packed executables

- Based on low number of imports
- Based on sections marked RWX

Dealing with Python Malware



Convert python scripts into standalone executables

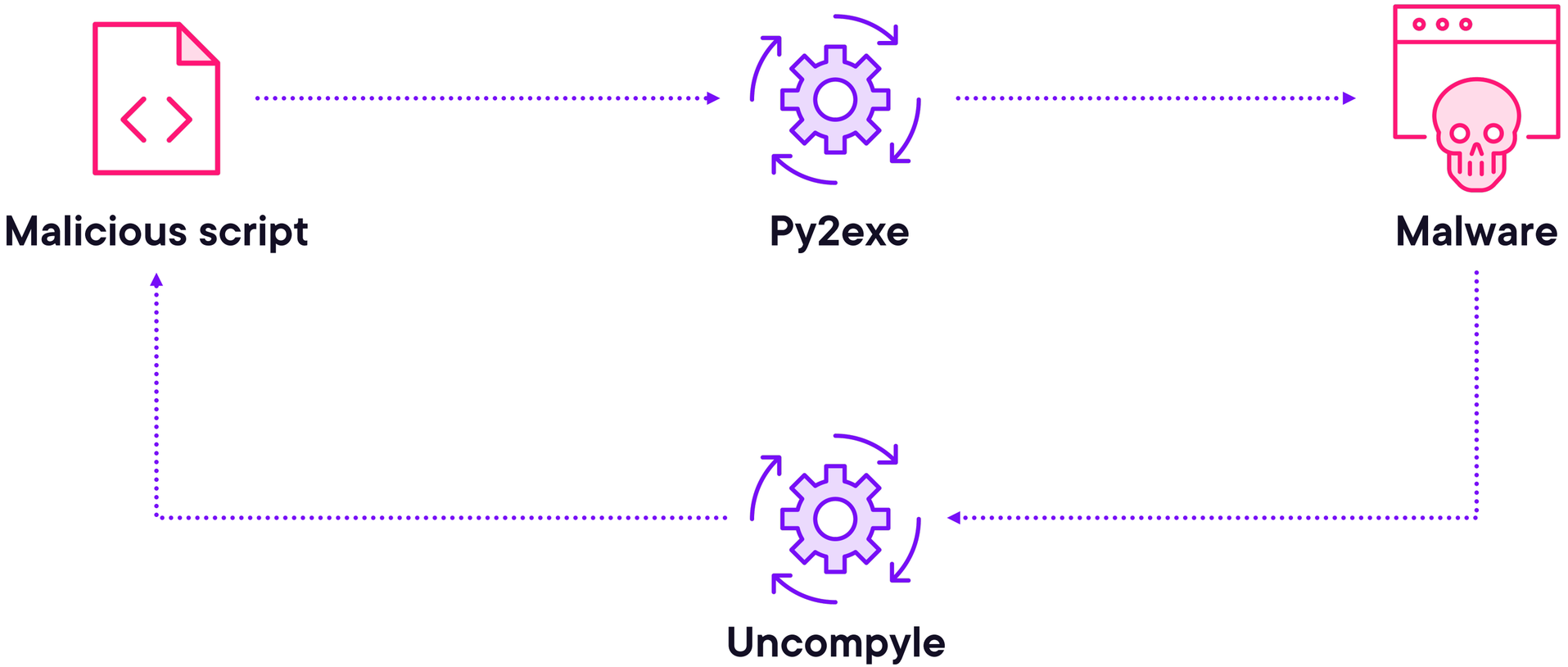
- Python not available on Windows by default
- Py2exe

Attackers leverage this as an attack vector

Analysis requires decompiling the executable

- Uncompyle

Dealing with Python Malware

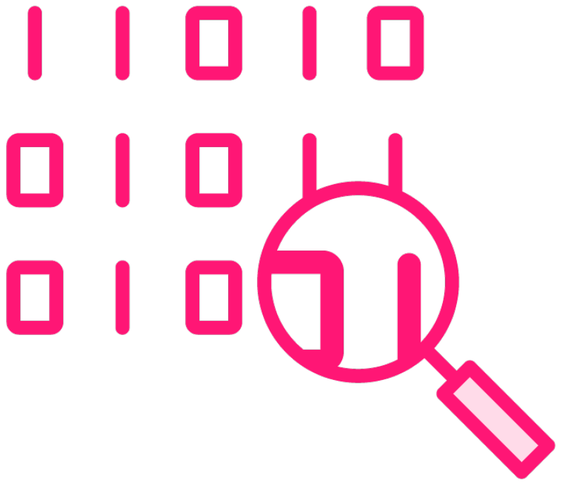


References

Py2exe
https://www.py2exe.org

Uncompyle
https://github.com/rocky/python-uncompyle6/

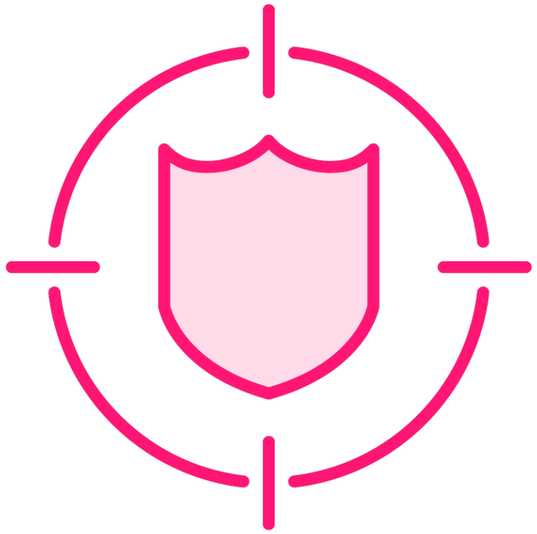
Managing Malware with Python



Viper

- Binary analysis and management framework for malware samples and scripts
- https://github.com/viper-framework/viper

Decoys and Honeypots



Leverage honeypots to capture and analyze malware samples

- Build, manage and monitor decoys with Python

References

Oletools
https://github.com/decalage2/oletools

Yara rules for detecting packers
https://github.com/Yara-Rules/rules/tree/master/packers

Pefile
https://github.com/erocarrera/pefile

Summary

Analyzed malicious documents

- Identified specific IoCs with olevba
- Extracted the obfuscated VBA script
- Integrated olevba into our script

Detected packed executables with pefile

- Based on low number of imported DLLs and functions
- Based on sections marked RWX

Python for Cyber Defense learning path

Thank you!