Malware Detection and Analysis with Python

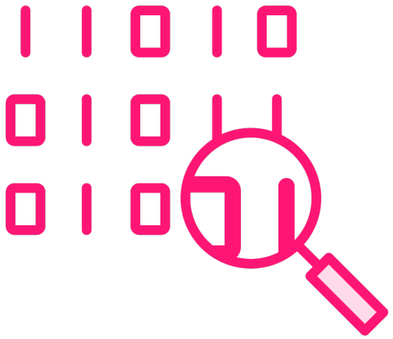
Detecting Malicious Files with Python



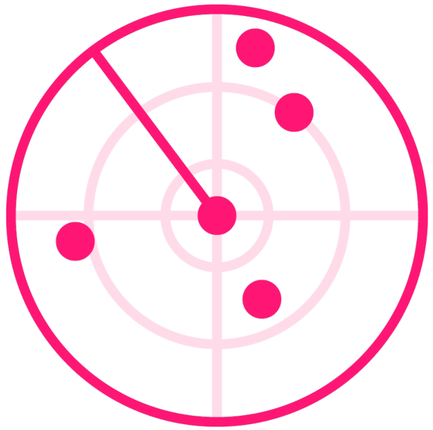
Cristian Pascariu
Information Security Professional

www.cybersomething.com

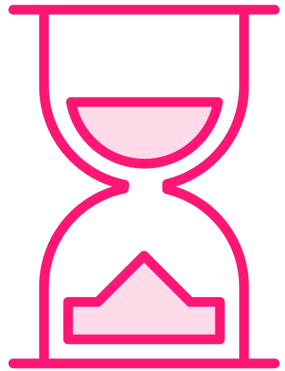
Objectives



Improve detection of malicious files

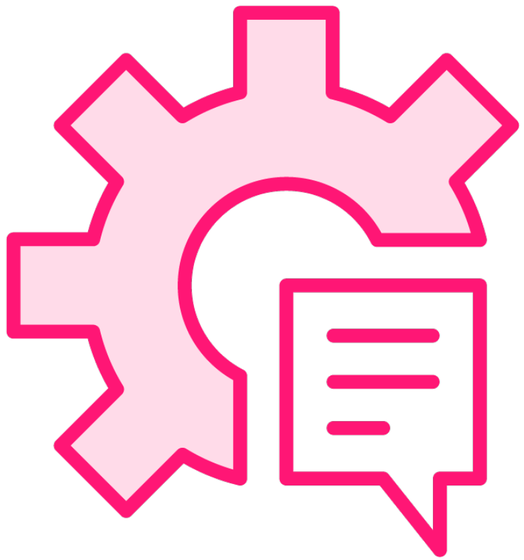


Eliminate blind spots



Save time

Course Logistics



Scripts will be developed for Python3

Can use any text editor or IDE

- Visual Studio Code on Linux is preferred
- WSL is also an option

Think of the scripts as building blocks

Samples available:

- https://ps.cybersomething.net/python-malware-analysis.html

Overview

File type identification

Generate file and import hashes

Identify malicious patterns with Yara

Leverage file reputation services

Classifying Files with Python



Cannot trust the file name, extension or icon

Identify file type based on file header

- The first bytes in the header of the file
- Also known as “magic numbers”

Detect if files are what they seem to be

Python package available: python-magic

- Available on Unix via the File command

File Headers

MZ
4D 5A





Sys

Scr

PK
50 4B





Jar

Apk

```
import os
```

The OS Module

Provides an interface to interact with the file system



```
import os


#check if a file
os.path.exists(<path>)
```

The OS Module

Provides an interface to interact with the file system



```
import os


#check if a file
os.path.exists(<path>)

#check if a file
os.path.isfile(<path>)

#check if a file
os.path.isdir(<path>)
```

The OS Module

Provides an interface to interact with the file system



```
import argparse
```

The Argparse Module

Module to handle arguments for command line tools



```
import argparse

#instantiate an argument parser
parser = argparse.ArgumentParser()
```

The Argparse Module

Module to handle arguments for command line tools



```
import argparse

#instantiate an argument parser
parser = argparse.ArgumentParser()

#add an argument
parser.add_argument(
    "-s",
    "--file_summary",
    help = "Get the file type and hashes",
    action = "store_true")
```

The Argparse Module

Module to handle arguments for command line tools



```
import argparse

#instantiate an argument parser
parser = argparse.ArgumentParser()

#add an argument
parser.add_argument(
    "-s",
    "--file_summary",
    help = "Get the file type and hashes",
    action = "store_true")

#retrieve arguments
args = parser.parse_args()
```

The Argparse Module

Module to handle arguments for command line tools

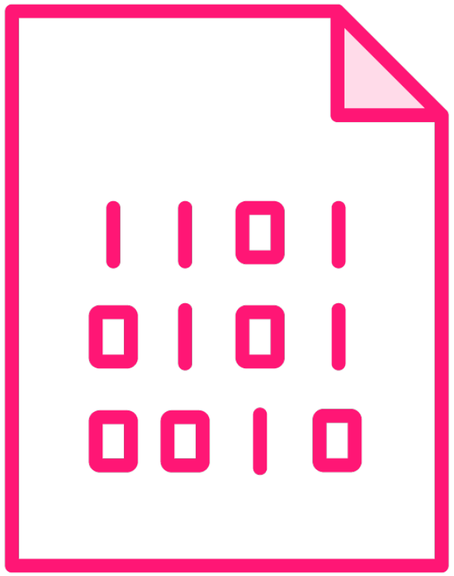


Set up a virtual environment

Install python-magic package

Classify unknown files

Working with File Hashes



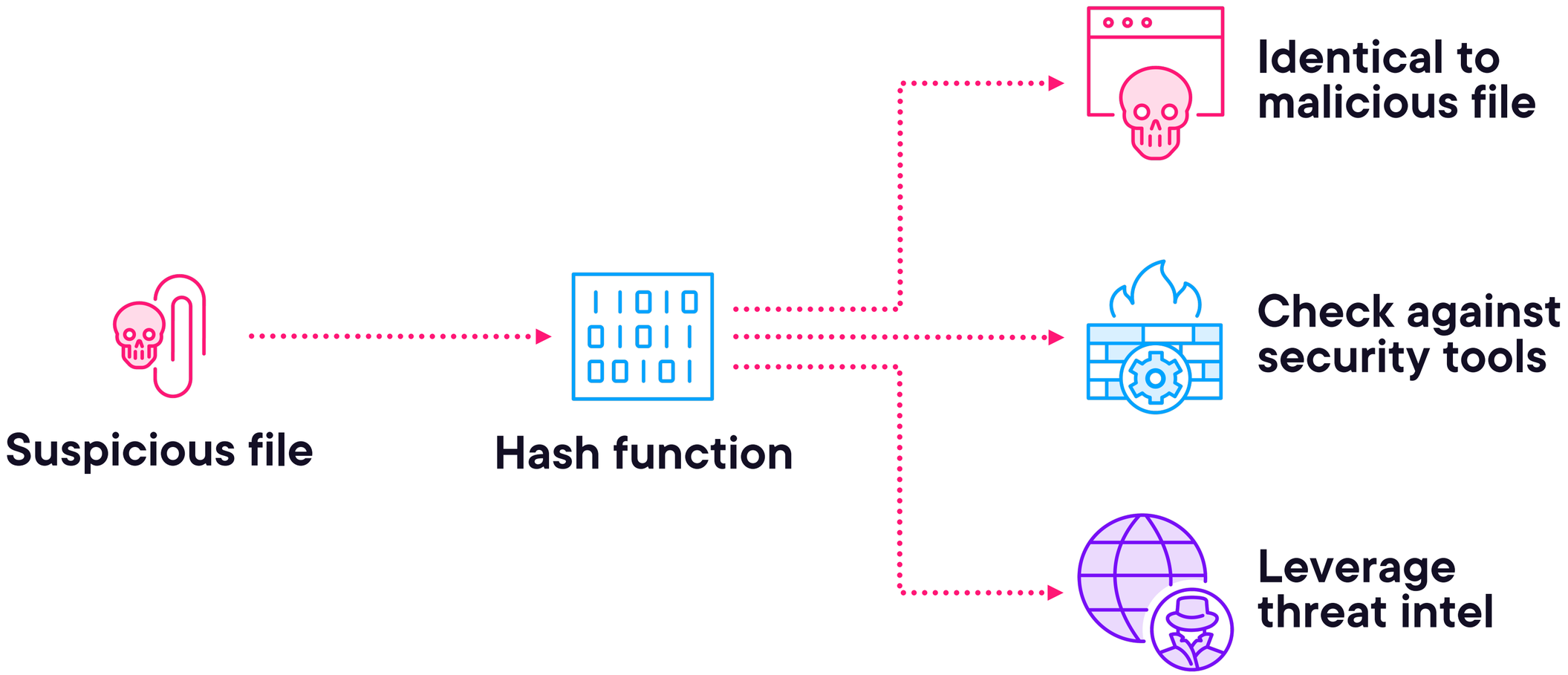
Represent the file content using a hash value

- Fixed length string based on content of variable length
- Hash value does not change regardless of file name or extension

Establish if two files are identical

File hash signatures have strong support within security tools

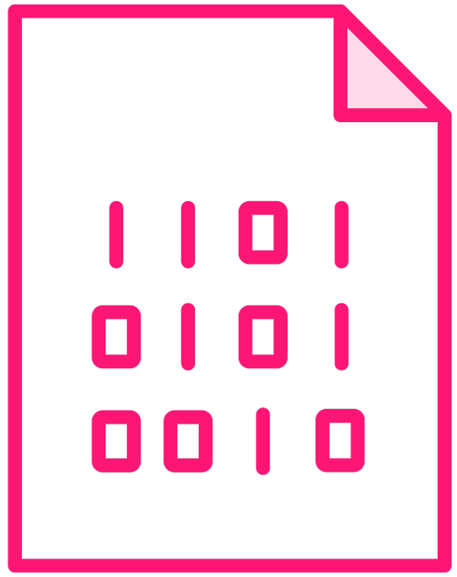
Working with File Hashes



Working with File Hashes

Challenges	VS	Opportunities
Modern malware can and will subvert this technique Any change will generate another hash value		Malware authors can reuse components between campaigns Malware can abuse legitimate files

Fuzzy Hashing and Import Hashing



Fuzzy hashing

- Hash blocks of content
- Tolerance to file changes
- Enables similarity comparison

Import hashing

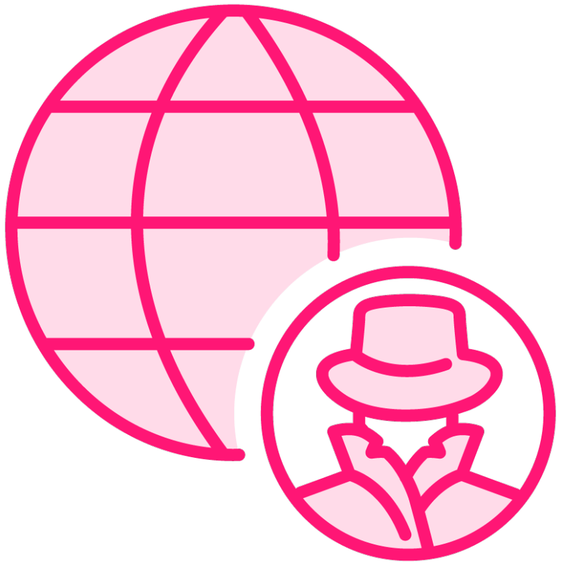
- Hashes the contents of the Import Address Table (IAT)

Introduce hashlib

Generate file hashes

Generate import hashes

Pattern Matching with Yara



Overcomes hash-based signatures limitations

- Identify byte sequences and patterns

Rule mechanism based on conditions

Yara is adopted by many security tools

```
import yara
```

The Yara-Python module

Leveraging Yara rules from Python



```
import yara


#instantiate an argument parser
rule = yara.compile(<rule_path>)

#add an argument
matches = rule.match(<file_path>)
```

The Yara-Python module

Leveraging Yara rules from Python



Review yara-python docs

Install yara-python using pip

Leverage Yara rules for detection

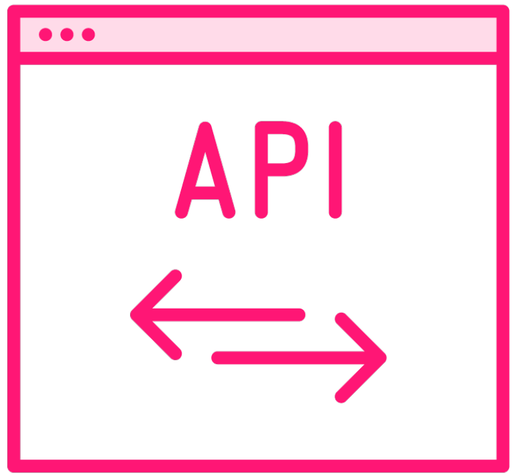


Learn more about Yara

Threat Hunting with Yara

Cristian Pascariu

Leverage Threat Intelligence



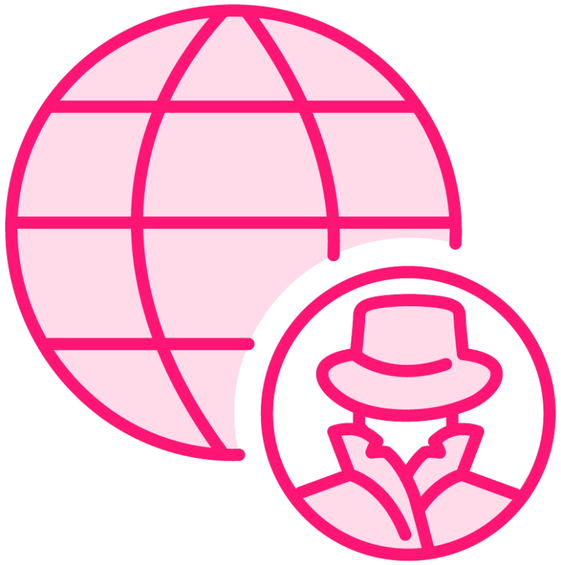
Leverage file reputation services

- Including threat intelligence feeds

Integrations vary based on the solution

- Native module available
- Supported by REST API

VirusTotal API



VirusTotal is a very popular solution

- Integration available via python module

Getting an API key requires registering

- Free tier available

Features

- Lookup file based on hash value
- Scan websites and domains
- Submit files for scanning

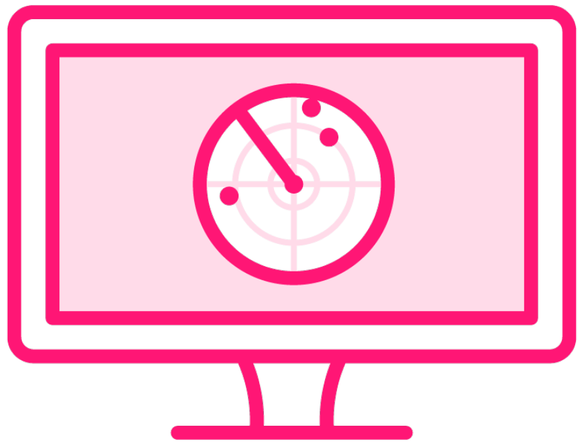
Review VirusTotal API docs

- Securely store the API key

Install vt-py using pip

Retrieve information for suspicious files

Getting Familiar with Existing Solutions



Loki

- IoC scanner
- https://github.com/Neo23x0/LOKI

Binary alert

- Cloud scanning solution for files
- Built for AWS with serverless functions
- https://www.binaryalert.io/



Running scripts in Docker

Developing Python3 Apps with Docker

Steven Haines

References

Python-magic
https://pypi.org/project/python-magic/

Yara-python
https://yara.readthedocs.io/en/stable/yarapython.html

VirusTotal python client
https://virustotal.github.io/vt-py/

Summary

Interacted with files on disk and command line arguments

Generate file and import hashes

Leverage Yara rules for pattern matching

Interact with the VirusTotal API to extract file reputation information

Conducting Malware Analysis with Python