Assignment 3: Malware Detection

Assigned: 4/11/24

Due: 14/11/24

Create a malware detection tool in Python with signature-based scanning. As described below, the assignment will involve file handling, dynamic hashing comparisons, and anomaly detection to create a malware detecting system. It is recommended to perform this on a virtual machine in case you don't create test malicious files.

Task A: Signature Database and detection

1. **Signature Database Creation.** Create a text file containing 50 random entries and label them as malware and non-malware texts or files, with their MD5 , SHA256 hashes. Create the file with a script or download the hashes from known resources (See part "Additional information part2"). The output should be in the template of the attached file: **malware_signatures.txt.**
2. **Malware detection.** Create test files with hashes of random text containing also malicious and non-malicious signatures (as shown in the previous task). Use Python's native os.urandom() for generating strings of random size and hashlib library to compute the hashes for the files. Compare them with the signature database created at taskA1. Identify if any of the test files have been categorized as malware by the signatures.
3. **Multi-Hash Validation.** Create a script to calculate the hashes (sha1, SHA512, SHA256) of the given pdf files (attached sample_pdfs). Perform pairwise comparisons between all hashes and report any interesting findings. How could this affect malware detection?

Task B: Search and Quarantine

1. **Recursive Directory Scanning:** Create a test directory with subdirectories containing test files (as created also in task A); include also test malware-infected files. Recursively scan the directory and its subdirectories using task A. Write detection logs, including all the details of: e.g., hashes, matched signatures, timestamps and file type (size). Provide a report .log
2. **Quarantine & Reporting Mechanism:** Build a quarantine system that isolate the suspicious files and print its threat level (Low, Medium, High) with a description.

Task C: Real-Time Monitoring and Anomaly Detection

1. **Real -Time Monitoring**. Create a monitoring tool that listens to a target directory for any file creation, modification or deletion with Python watchdog library. You can use the created directory from task B. Test with some file creation and modification and report the results.
2. **Real-Time Detection.** Include the detection script from task A to monitor and report if any malware-infected file is created or modified.
3. **Quarantine.** Quarantine any new generated or modified infected items from task C2 and keep updating the report consequently.

Tool Specification

The tool should accept the following command-line arguments:

- -d <directory>: The directory to scan.
- -s <signature file>: Path to the malware signature database.
- -o <output file>: File to save a report of infected files.
- -r (optional): Run in real-time mode to monitor the directory.

Submission

You will need to deliver in a compressed (.zip) folder containing the following:

1. All python source code with detailed comments.
2. A comprehensive malware signature database
3. A README file with usage instructions and additional notes on the implemented scripts. Include also the explanation of the task A3.

Notes

- Ensure robust error handling throughout the code, addressing issues such as invalid files and access permissions.
- Please google your question before asking ☺ .
- Submit the project in a single folder named **<login>_**assign2**.zip** where AM is your login name.

More details on the implementation

Use the os and shutil libraries for directory manipulation. The detection script can compare file contents using hash functions (MD5/SHA-256) or by checking file metadata.

Task A1

Generate for sample text string or test files the

```
md5_hash = hashlib.md5(data.encode()).hexdigest()
sha256_hash = hashlib.sha256(data.encode()).hexdigest()
```

and include the in your malware_file.txt.

Task A2

Create a script taskA_2_create_test_files.py that will create the files, including some test malicious files (including only malware signatures, not the payload ☺) .

Create a script taskA_2.py that will compute the hashes for the files, and compare to malware signature you created in malware_signatures.txt.

hash_md5 = hashlib.md5()

hash_sha1 = hashlib.sha1()

hash_sha256 = hashlib.sha256()

hash_sha512 = hashlib.sha512()

Additional information

Quarantine Risk — the script moves files to quarantine based on hash signature if deemed suspicious. Detects error/ false positives. In case of any system file is quarantined by mistake then your local computer may get a problem.

Settings by the tool for only scan, modify and file monitor round clock. If not properly configured, it might end up corrupting critical files accidentally causing a loss in data.

Security Risks: If you're working with real malware samples, running your host system through the cold would carry some risk if any unknown or malicious files manage to evade detection.

Part 1) Recommendations:

Virtual Machine (VM): Use a virtualized environment such as VirtualBox, VMware or Docker to run the tool for air-gapping and isolating your main OS from any threats.
Mock Files — similar to Stubar, you can test the script on benign pre-generated files with known hash signatures rather than actual malware samples.
Backups: Make sure you have a backup of your important files before running any malware-related software on your main system.

Part 2) Malware Signatures (task A1)

1. Resources for malware signatures
    a. Public Malware Databases:
        i. www.eicar.org, malshare.com, VirusTotal, Hybrid Analysis etc.
        ii. MalwareBazaar archive of malware instances available for seeing and making use of for the purpose of diagnosis.
        iii. OTX (Open Threat Exchange): An environment where individuals exchange information about threats, among other things, including malicious software.

b. Research Publications: research containing detailed references
c. Open-Source Projects: GitHub like the OWASP Web Malware Scanner
d. Malware Analysis Tools: Cuckoo Sandbox and Remnux, for instance, are tools that systematically analyze malware samples and catch their signatures.

2. MD5 Hashes of known Viruses. In order to create test files for known malware virus, you have to insert malware signatures.
    a) An example is MD5 Hash: d41d8cd98f00b204e9800998ecf8427e, where this hash is the MD5 hash of an empty file. It is commonly used in malware detection as a signature for certain types of malware that may create empty or placeholder files as part of their infection process. Some viruses and other malicious software might create empty files as a means to establish persistence or as a marker for infection.
    b) SHA256 Hash: 44d88612fea8a8f36de82e1278abb02f07e6a2c75dee2983b74b981f4d6db6c8 This is the SHA256 hash of the famous EICAR test file, which is commonly used to test the detection capabilities of antivirus software. The file itself is harmless and used in a controlled environment to check whether antivirus software is properly detecting threats. There is an example of how to create this in the attached generate_eicar.py using: https://www.eicar.org/download-anti-malware-testfile/

3) Basic Theory

1. What is Malware?

Malware or malicious software refers to software specifically crafted to damage or disrupt a system, automatically spread, steal information or often be delivering different forms of unwanted functionality. This can be a virus or worm, or even a trojan, ransomware, spyware, adware, etc. with the analyses on malware. Some common types can be:

Virus: attaches itself to a legitimate program to produce itself.

Worm: Self-replicates and move on from one computer to another through a network or an Internet connection.

Trojan: disguising itself as a legit piece of software to trick users to install them while its doing things you don't know about.

Ransomware: encrypts the files of the users and demands money to decrypt them, often against the user permission.

Spyware: tracks down the users and steal information about them without their endorsement.

Adware: once installed on a system, the adware software bluffs users to click on some prompted ads, often encouraged to install through sneaky ways for the ad server company to obtain revenue from the adverting company.

2. How can a Malware be detected?

Malware detection is often presented in the following three broader categories.

Signature-Based Detection: is the way malware is detected using previously identified signatures or patterns of the known malware Task A in the ACSC competition is the signature database in the malware detection problem where records of different malware are listed.

Heuristic Analysis: detects unknown/new threats or modified malware by means of newly created virsus traits before any of these traits is known/available.

Behavior Based Detection: tries to identify the status of the system of suspicious activities that might be happening. For example, it may notice a file is written then rewritten, besides other network and disk activities.

3. Hashing Algorithms

A hash is a small sized string that is produced from a data string using a hash function.

- MD5(Message Digest Algorithm 5): produces a 128-bit hash value, and could be commonly used given its performance advantage, however, it does not save from a collision and should be desired over the
- SHA function SHA256(Secure Hash Algorithm 256): Produces a 256-bit hash which is claimed to be secure than MD5 but with the cost of taking longer to complete. Might be desired for some uses.
- SHA1 and SHA512: Other hash algorithms for file validation check but should not be encouraged to be used.

.