**Project Title:**

**Hash-Based Signature Detection for Malware and Ransomware**

**Introduction:**

The increasing prevalence of malware, especially ransomware, poses a significant threat to computer systems globally. Traditional antivirus solutions rely on signature-based detection to identify known threats. This project focuses on implementing a hash-based signature detection system to detect malware by comparing file hashes to a database of known malware signatures.

**Objective:**

The objective of this mini project is to develop a Python-based system that scans files, generates their cryptographic hash values, and checks them against a database of known malware hashes. If a match is found, the system flags the file as potentially malicious.

**Scope:**

This project covers the following:

* Understanding the fundamentals of hash-based detection.
* Implementing a system that computes SHA-256 hashes for files.
* Building a malware hash database for comparison.
* Scanning directories for malware-infected files by comparing their hashes.
* Reporting on detection results.

**Methodology:**

1. **Hashing Mechanism**:
   * Implement a file hashing system using the **SHA-256** algorithm, which generates a unique hash value for every file.
2. **Malware Signature Database**:
   * Create and maintain a **CSV file** of known malware hashes (e.g., hashes from public sources like VirusTotal) that will serve as the detection reference.
3. **File Scanning**:
   * Scan files in a specified directory, compute their hash values, and compare them against the malware hash database to detect infections.
4. **Result Evaluation**:
   * If a hash matches a known malware signature, the file will be flagged and reported as malware.

**Tools and Technologies:**

* **Programming Language**: Python
* **Libraries**:
  + hashlib (for generating file hashes),
  + os (for directory/file handling),
  + csv (for managing the malware hash database).

**Expected Outcome:**

* A Python-based system capable of scanning directories for malware by comparing file hashes with a pre-existing malware database.
* Detection of files infected with ransomware or other malware based on hash matches.
* A report highlighting test cases, detections, and system performance.

**Conclusion:**

This project will demonstrate the effectiveness of **hash-based signature detection** in identifying known malware and ransomware. While limited to detecting only known threats, this method remains an efficient way to combat certain types of malware. Future enhancements may include real-time detection and integration with online malware databases.

**References:**

* Use of public malware databases such as VirusTotal for gathering known malware hashes.
* Python’s official documentation for hashlib, os, and csv libraries.

Here's the Python implementation of your **Hash-Based Signature Detection** system, broken down step by step:

**Step 1: Import Required Libraries**

First, import the libraries you'll need to compute file hashes, handle directories, and read/write CSV files.

python

Copy code

import hashlib

import os

import csv

**Step 2: Function to Generate SHA-256 Hashes for Files**

This function computes the SHA-256 hash for any file.

python

Copy code

def get\_file\_hash(file\_path):

sha256\_hash = hashlib.sha256()

try:

with open(file\_path, "rb") as f:

# Read file in chunks to handle large files

for byte\_block in iter(lambda: f.read(4096), b""):

sha256\_hash.update(byte\_block)

return sha256\_hash.hexdigest()

except Exception as e:

print(f"Error reading {file\_path}: {e}")

return None

**Step 3: Function to Load Malware Hashes from a CSV File**

This function reads a CSV file containing malware hashes and stores them in a dictionary for quick lookup.

python

Copy code

def load\_malware\_hashes(csv\_file):

malware\_hashes = {}

try:

with open(csv\_file, mode='r') as file:

reader = csv.DictReader(file)

for row in reader:

malware\_hashes[row['hash']] = row['malware\_name']

except Exception as e:

print(f"Error reading CSV file {csv\_file}: {e}")

return malware\_hashes

**Step 4: Function to Compare File Hashes with Known Malware Hashes**

This function takes the file's hash and checks it against the malware database.

python

Copy code

def check\_file\_for\_malware(file\_path, malware\_hashes):

file\_hash = get\_file\_hash(file\_path)

if file\_hash:

if file\_hash in malware\_hashes:

print(f"ALERT: {file\_path} is infected with {malware\_hashes[file\_hash]}!")

else:

print(f"{file\_path} is clean.")

**Step 5: Function to Scan a Directory for Malware**

This function scans all files in a directory, generates their hashes, and compares them to the malware hashes.

python

Copy code

def scan\_directory(directory\_path, malware\_hashes):

for root, dirs, files in os.walk(directory\_path):

for file in files:

file\_path = os.path.join(root, file)

check\_file\_for\_malware(file\_path, malware\_hashes)

**Step 6: Main Execution of the Program**

In the main function, load the malware hash database, specify the directory to scan, and start the scan.

python

Copy code

if \_\_name\_\_ == "\_\_main\_\_":

# Load known malware hashes from CSV

malware\_hashes = load\_malware\_hashes("malware\_hashes.csv")

# Specify the directory to scan

directory\_to\_scan = "test\_directory" # Replace this with the path you want to scan

# Scan the directory for malware

scan\_directory(directory\_to\_scan, malware\_hashes)

**How the Program Works:**

1. **get\_file\_hash**: Generates the SHA-256 hash of each file in the specified directory.
2. **load\_malware\_hashes**: Loads a CSV file that contains known malware hashes and their names.
3. **check\_file\_for\_malware**: Compares each file’s hash against the known malware hashes and flags it if found.
4. **scan\_directory**: Recursively scans a directory for all files and calls check\_file\_for\_malware for each file.

**Sample CSV File (malware\_hashes.csv) Format:**

Here’s what the malware\_hashes.csv file should look like:

csv

Copy code

hash,malware\_name

5d41402abc4b2a76b9719d911017c592, ExampleMalware1

8c8dfe0fa43e417d0eb7a267d9b3bc9b, ExampleRansomware

d2d2d2f2f2f2f1b1b1b1b1b1a1a1a1a1, AnotherMalware

You can update this file with real malware hashes obtained from sources like VirusTotal or other malware databases.

**Testing the Program:**

1. **Create a Test Directory**:
   * Put some sample files in the directory you want to scan.
   * Add a file with a hash that matches one of the entries in malware\_hashes.csv to simulate malware detection.
2. **Run the Program**:
   * Execute the program, and it will scan the directory, hash each file, and flag any matches with the malware hash database.

**Optional Enhancements:**

1. **Real-Time Detection**: You could use a file-watching mechanism like watchdog to automatically scan new files added to the directory.
2. **User Interface**: Add a graphical interface using Tkinter for easier file selection and scanning.
3. **Hash Database Update**: Implement a feature to periodically update the malware hash database by fetching the latest hashes from an online source.