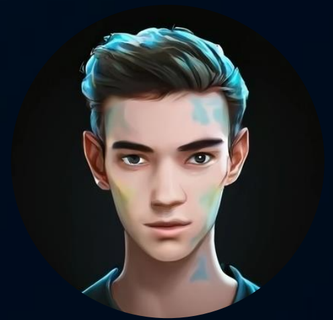
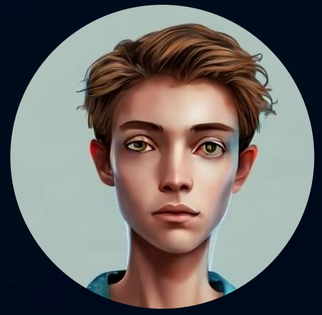
MALWARE DETECTION TOOL

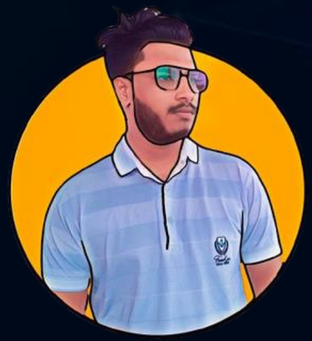
A Python-based tool for detecting malicious files



Tusher Alom Linkun
213002251



MD. ABDULLAH HASBIN SANET
212902008



MD. RAHUL ISLAM JOY
212902070

Contents

1. Introduction
    a. Malware
    b. Methodologies
    c. Hashing in cryptography
2. Project overview
3. Implementation
4. Challenges and future work
5. Conclusion

What is malware ?

MALWARE IS A TOOL/PAYLOAD WHICH IS DESIGNED TO HARM VICTIM'S DEVICE TO IMPLEMENT SYSTEM CRASHING, DATA FORWARDING, REMOTE ACCESS, ETC.

Types of malware

1. Trojan
2. Rat's
3. Ransomware
4. Dropper

Methodologies

1. Static analysis
2. Dynamic analysis
3. Reverse engineering
4. Encryption & decryption analysis
5. Behavioral analysis

Static Analysis

1. Source code analysis
2. Binary analysis
3. Hash value analysis

What is hash?

A HASH IS A FIXED-SIZE STRING OR NUMBER GENERATED FROM INPUT DATA OF ARBITRARY SIZE, USING A HASH FUNCTION. THE HASH VALUE (ALSO CALLED A HASH CODE OR DIGEST) UNIQUELY REPRESENTS THE DATA, TYPICALLY FOR THE PURPOSE OF QUICKLY IDENTIFYING OR VERIFYING IT.

Types of hashes

1. MD5
2. SHA1
3. SHA256
4. SSDEEP
5. TLSH
6. IMPHASH

Hash value analysis

1. MD5 – 128 bits
2. SHA1 – 160 bits
3. SHA256 – 256 bits

Project Overview

```
====== Main Menu ======
1. File Analysis
2. Hash Analysis
3. About Me
4. Exit
======================
Choose an option (1-4): 1
Enter the full file path to scan: C:\Users\RI Rahul\Documents\test_file2.txt
MD5: d41d8cd98f00b204e9800998ecf8427e - No match found
SHA1: da39a3ee5e6b4b0d3255bfef95601890afd80709 - No match found
WARNING: SHA256 hash matches known malware signature: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

```
====== Main Menu ======
1. File Analysis
2. Hash Analysis
3. About Me
4. Exit
======================
Choose an option (1-4): 2
Enter the hash type (md5, sha1, sha256): sha256
Enter the hash value to check: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
WARNING: SHA256 hash matches known malware signature: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Project Implementation

```
====== Main Menu ======
1. File Analysis
2. Hash Analysis
3. About Me
4. Exit
======================
Choose an option (1-4):
```

```
def main():
    while True:
        print("\n====== Main Menu ======")
        print("1. File Analysis")
        print("2. Hash Analysis")
        print("3. About Me")
        print("4. Exit")
        print("======================")

        choice = input("Choose an option (1-4): ").strip()

        if choice == '1':
            file_analysis()
        elif choice == '2':
            hash_analysis()
        elif choice == '3':
            about_me()
        elif choice == '4':
            print("Exiting the program. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

File Analysis

```
====== Main Menu ======
1. File Analysis
2. Hash Analysis
3. About Me
4. Exit
======================
Choose an option (1-4): 1
Enter the full file path to scan: C:\Users\RI Rahul\Documents\test_file2.txt
MD5: d41d8cd98f00b204e9800998ecf8427e - No match found
SHA1: da39a3ee5e6b4b0d3255bfef95601890afd80709 - No match found
WARNING: SHA256 hash matches known malware signature: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

User file path

```
def file_analysis():
    file_path = input("Enter the full file path to scan: ").strip()
```

Calculate and check hashes

```
for algo, hash_set in hash_algorithms.items():
        hash_value = calculate_hash(file_path, algo)
        if hash_value:
            if match_hash_with_file(hash_value, hash_set):
                print(colored(f"WARNING: {algo.upper()} hash matches known
malware signature: {hash_value}", 'red'))
            else:
                print(colored(f"{algo.upper()}: {hash_value} - No match found",
'green'))
        else:
            print(f"Error: Could not calculate {algo.upper()} hash for the file.")
```

Paths to the malware database files

```
database_dir =
"G:\\GUB\\CS\\Project\\malware_detection_tool\\malware_db"
    md5_db_path = f"{database_dir}\\md5_hash.txt"
    sha1_db_path = f"{database_dir}\\sha1_hash.txt"
    sha256_db_path = f"{database_dir}\\sha256_hash.txt"
```

Load hash values from the database files

```
    md5_hashes = read_hashes(md5_db_path)
    sha1_hashes = read_hashes(sha1_db_path)
    sha256_hashes = read_hashes(sha256_db_path)
```

Hash types and their respective database

```
    hash_algorithms = {
        'md5': md5_hashes,
        'sha1': sha1_hashes,
        'sha256': sha256_hashes
    }
```

Project Implementation – Hash Analysis

```
====== Main Menu ======
1. File Analysis
2. Hash Analysis
3. About Me
4. Exit
======================
Choose an option (1-4): 1
Enter the full file path to scan: C:\Users\RI Rahul\Documents\test_file2.txt
MD5: d41d8cd98f00b204e9800998ecf8427e - No match found
SHA1: da39a3ee5e6b4b0d3255bfef95601890afd80709 - No match found
WARNING: SHA256 hash matches known malware signature: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

```
# Hash analysis functionality
def hash_analysis():
    hash_type = input("Enter the hash type (md5, sha1, sha256): ").strip().lower()
    if hash_type not in ['md5', 'sha1', 'sha256']:
        print("Invalid hash type. Please choose from md5, sha1, or sha256.")
        return

    hash_value = input("Enter the hash value to check: ").strip()

    # Path to the malware database file
    database_dir = "G:\\GUB\\CS\\Project\\malware_detection_tool\\malware_db"
    db_paths = {
        'md5': f"{database_dir}\\md5_hash.txt",
        'sha1': f"{database_dir}\\sha1_hash.txt",
        'sha256': f"{database_dir}\\sha256_hash.txt"
    }
```

```
    # Load the respective hash database
    hash_set = read_hashes(db_paths[hash_type])
    if match_hash_with_file(hash_value, hash_set):
        print(colored(f"WARNING: {hash_type.upper()} hash matches known
malware signature: {hash_value}", 'red'))
    else:
        print(colored(f"{hash_type.upper()} hash: {hash_value} - No match
found", 'green'))
```

Challenges

1. Protecting the malware database from unauthorized access.
2. Efficient hash computation for large files.
3. Handling invalid inputs from users.
4. Large dataset handling for live server.

Future Work

1. Encrypt and securely host the malware database.
2. Extend to web and mobile platforms.
3. Add advanced detection features like:
    - Dynamic analysis
    - Behavior analysis.
    - Encryption & decryption analysis
4. Improve performance for large datasets.

Conclusion

The Malware Detection Tool Using Hash Validation successfully provides an efficient and accessible method for detecting malicious files. By utilizing cryptographic hash functions (MD5, SHA1, and SHA256), the tool accurately matches file and hash inputs against a database of known malware signatures. Testing results confirmed 100% accuracy with quick execution times, demonstrating the tool’s reliability and usability. Its modular design allows for future scalability and broader applications in cybersecurity.

THANK YOU