A Software Integrity Checker

HANN, Evan

Supervised by Prof. DING, Cunsheng

Overview

- There are many malicious applications targeting Android breaching the system integrity
- Some such as rootkits can attack by modifying system files breaching the data integrity
- Mechanisms to defend against them are necessary

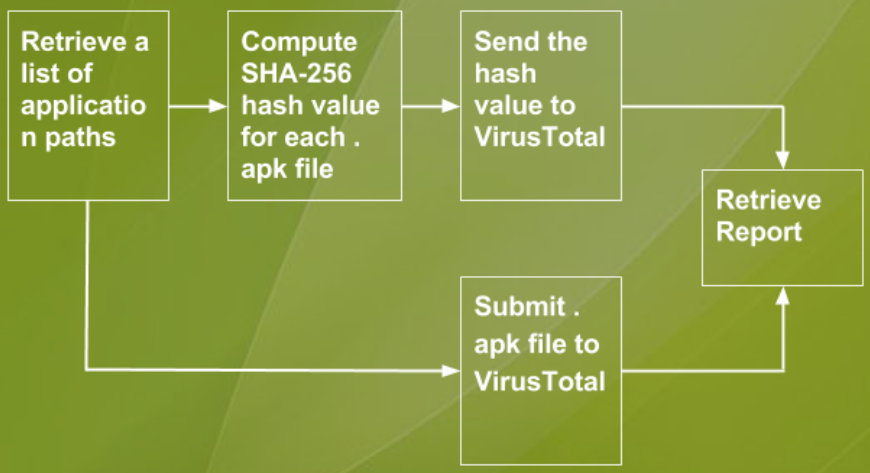
Objectives

- Develop a malware scanner for detecting malicious applications to verify system integrity
- Develop a file monitor for detecting file changes to verify data integrity

Online Malware Analysis

- Malware analyses are offloaded to VirusTotal servers
- Saving computational and storage resources
- More than 47 antivirus engines available for better detection
- Can simply retrieve results by matching hash values if the apps have been submitted by others

Flow Diagram for Malware Scanner



SHA-2

- SHA-2 is one of the hash algorithms chosen for the project
- A hash algorithm is a function H that transforms a message M into a hash value H(M)
- And it is computationally infeasible to create another message N with H(N) = H(M), where N ≠ M
- Therefore, hash algorithms such as SHA-2 can be used to detect modifications in file

New standard SHA-3

- The other hash algorithm chosen is SHA-3, aka Keccak
- SHA-3 was implemented in the project as it is not supported by Android Java Library
- Structurally different from SHA-1 and SHA-2
- Attacks against the previous SHA families would not work against SHA-3

AES

- AES is an encryption algorithm, which provides data confidentiality as well as data integrity services
- The file storing hash values is protected with AES
- Encryption key is stored in KeyStore directory, which is accessible by the KeyStore daemon only
- Difficult for hackers to remove or replace any particular hash values

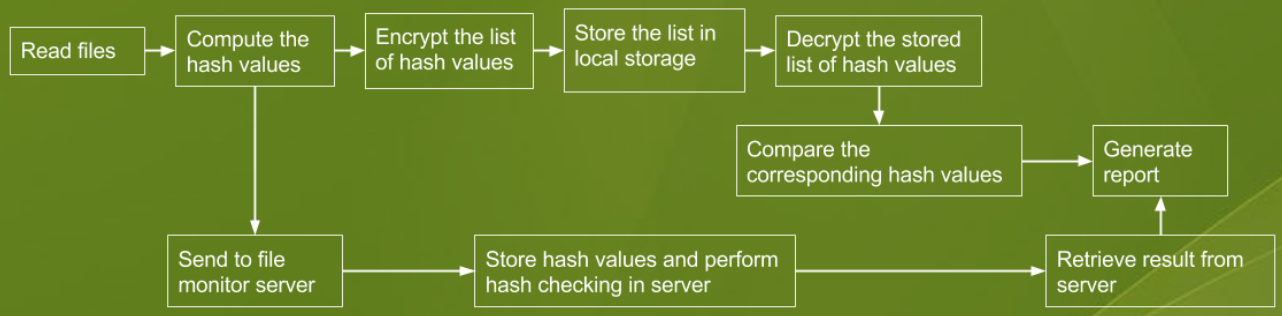
SHA-3 Pseudo-code

```
KECCAK[r,c](M) {
  Initialization and padding
  S[x,y] = 0,
  P = M || 0x01 || 0x00 || … || 0x00
  P = P xor (0x00 || … || 0x00 || 0x80)

  Absorbing phase
  forall block Pi in P
    S[x,y] = S[x,y] xor Pi[x+5*y],
    S = KECCAK-f[r+c](S)

  Squeezing phase
  Z = empty string
  while output is requested
    Z = Z || S[x,y],
    S = KECCAK-f[r+c](S)

  return Z
}
```

File Monitor Server

- Storing hash values in a server instead of storing in the local storage
- Additional barrier for hackers to tamper hash value records
- Hash value comparison is also carried out in the server
- Saving computational and storage resources

Flow Diagram for File Monitor



Real Time Monitoring

- Real time monitoring by hash checking is not practical
- FileObserver, supported by Android API, can detect file system change in real time
- By monitoring the inodes of files
- Based on the Linux counterpart inotify
- No cryptographic computation is necessary
- Lightweight, low overhead

Results

- Successfully implemented malware scanner and file monitor
- Able detect malware samples across various malware families
- Able to submit file to VirusTotal for malware analysis
- Able to monitor file changes by hash checking
- Successfully coupled with file monitor server
- Able to alert user in real time when the file is changed

Screenshots