Project Specification - Group 1

Joakim Loxdal, Douglas Fischer,
Oussama Kaddami, Abdelmoujib Megzari

November 12th 2021

1 Introduction

Malware is harmful software, designed to cause some form of damage to a system without the consent of the system user. Some malware infects benign binaries to hide themselves from discovery.

To prevent this, detection software has been developed that tries to detect malware in binaries or other files by verifying signatures (hashes) [1] against a set of known signatures, or by signing the binaries cryptographically beforehand and verifying them on execution [2].

In our project we aim to implement a detection system into the OS level by storing signatures (hashes) of accepted binaries in a database and make sure that the signatures still hold before executing the binaries. We assume that the OS kernel is trusted.

2 Targeted Vulnerabilities

Some types of malware such as rootkits infect ELF files to assure continued access, or to hide themselves from the system administrator. Signing user binaries and modules in the system and doing signature verification at loading time will prevent execution of modified binaries. This can be extended to disk integrity check of the file system.

One strategy is to use public key signing through RSA, but this requires adding a crypto module into the kernel and to keep both the public and private keys securely hidden. Another strategy is to create hashes of the binaries and store these in a table/database, and verify that the binary has not been altered before execution. The challenge with this strategy is to keep an intact version of hashes that serve as references during loading time that an attacker cannot modify.

3 Minimal Requirements

1. Hash some user binaries for which integrity is sought.

2. Implement a storage for known trusted hashes
3. Check signature on demand at loading time of user binary code
4. Alert the user about the potential malware and stop the execution of the binary

4 Optional Requirements

1. Use our technique on both user level binaries and kernel modules
2. Optimize the overhead due of the signature check
3. Add an integrity check of the disk once in a while
4. Secure the storage of hashes

References

[1] L. Litty, H. A. Lagar-Cavilla, and D. Lie, "Hypervisor support for identifying covertly executing binaries.," in *USENIX Security Symposium*, vol. 22, p. 70, 2008.

[2] A. Apvrille, D. Gordon, S. E. Hallyn, M. Pourzandi, and V. Roy, "Digsig: Runtime authentication of binaries at kernel level.," in *LISA*, vol. 4, pp. 59–66, 2004.