Improvement of Binary Analysis Components in Automated Malware Analysis Framework

Keiji Takeda
KEIO UNIVERSITY

02/21/2017
Final Report

DISTRIBUTION A: Distribution approved for public release.
### Abstract

This research was conducted to develop components for an automated system to analyze malicious software (malware) with minimum human interaction. The system autonomously analyzes malware samples by examining malware binary programs and monitoring their behavior, generating data for malware detection signatures and developing their countermeasures.
1 Objective

This research was conducted to develop components for automated system to analyze malicious software (malware) with minimum human interaction. The system autonomously analyze malware samples by analyzing malware binary program and by monitoring their behavior, then generate data for malware detection signature and for developing their counter measure.

2 Research Outcome

Through this research project, components of a malware analysis framework which integrates both dynamic analysis and static analysis techniques have been developed. By using the developed components, a malware analyst is able to analyze malware functions while avoiding interference by the malware. The developed components analyzes malware executable by suppressing interference from malware, generates a list of C&C (Communication and Control) servers the target malware may connect, identifies code sections for encryption and decryption functions and identifies code sections for commands from the C&C servers.

The framework challenged two fundamental limitations of existing analysis platforms. The first challenge is to comprehensively extract the potential functions of malware and the second challenge is to complete the analysis in closed environment without requiring active C&C servers on the Internet. The framework conducts dynamic analysis by executing malware on a sandbox environment isolated actual computer network such as the Internet then conducts static analysis for the code sections where not executed due to self-protection mechanism of the malware. As a next step it forces to execute the portion of the code to analyze the malware automatically. The framework has process trace function by instructions, by API, and by system calls, taint analysis function and symbolic execution functions also implemented.

The framework utilizes QEMU open source machine emulator and virtualizer as base virtual machine, and utilizes modified DECAF (Dynamic Executable Code Analysis Framework) for dynamic analysis and uses Angr for static analysis and symbolic execution. The developed framework was tested with 412 malware obtained from...
multiple malware data sets. The test result shows that the framework has capability to suppress analysis evasion mechanism of malware (100%), to obtain potential address list of C&C servers (approximately 5% error rate), to detect libraries used to encrypt and decrypt (100% for known APIs) and to identify command handler without having actual C&C server (approximately 90%).

The developed components enables malware analysts to observe functions of malware quickly and helps implementation of countermeasures to prevent or mitigate damage caused by malware.

3 Publication


