Exploiting Ransomware Paranoia For Execution Prevention

Ali AlSabeh, Jorge Crichigno, Elias Bou-Harb
Integrated Information Technology Department, University of South Carolina, Columbia, South Carolina
The Cyber Center For Security and Analytics, University of Texas at San Antonio, USA

Abstract

• Several ransomware detection approaches have been proposed in the literature that interchange between static and dynamic analysis.
• Recently, ransomware attacks were shown to fingerprint the execution environment before they attack the system to counter dynamic analysis.
• In this project, we exploit the behavior of contemporary ransomware to prevent its attack on real systems and thus avoid the loss of any data.
• We explore a set of ransomware-generated artifacts that are launched to sniff the surrounding and develop an approach that monitors the behavior of a program by intercepting the called APIs.
• The approach determines in real-time if the program is trying to inspect its surrounding before the attack and abort it immediately prior to the initiation of any malicious encryption or locking.
• Through empirical evaluations, we study how ransomware and benign programs inspect the environment and demonstrate how to prevent ransomware with a low false positive rate.

Contributions

• Exploring the behavior of contemporary ransomware by collecting relevant artifacts related to fingerprinting the execution environment, such as inspecting running processes, system files, registries, and CPU performance.
• Designing and developing a host-based approach which can detect contemporary ransomware through monitoring their "paranoia" (i.e., generated behavior targeting the execution environment) to prevent it from encrypting/locking the host machine through investigation techniques rooted in API interception methods.
• Executing empirical evaluations using real ransomware datasets, and achieving an accuracy of 99% on training data, and 84% on testing data.

Ransomware detection techniques gaps

• Focusing on detection rather than prevention.
• Detection is mainly based on ransomware's high level behavior (encryption/locking).
• Lack of considering contemporary ransomware behavior.

Methodology

• Collect diverse set of malware samples, and filter the ones related to ransomware.
• Among the collected ransomware samples, perform ransomware family labeling to assure that the samples are representative and diverse.
• Collect a set of APIs that are related to environment inspection (executed by different ransomware samples) prior to enumerating and encrypting the target's files.
• Tune the ratio of false positives by assigning a rank/priority that shows how likely this API is to be launched by evasive ransomware samples for sniffing the environment.
• Integrate the collected APIs inside a DLL and monitor programs' execution by injecting this DLL into the address spaces of the executing processes.
• The proposed monitoring mechanism will begin the moment a program is executed.
• If the monitored program attempts to fingerprint the environment through reaching a certain threshold where it is considered evasive ransomware, then a kill signal is sent to abort its execution.
• Else, it is deemed as a benign operation and its execution is uninterrupted.

Empirical evaluation

Scores reported by our implemented approach when measuring the evasiveness of ransomware training samples.

Concluding remarks and future direction

• The proposed approach addresses evasive ransomware attacks that perform fingerprinting to check if they are being executed in a real or monitored environment, and prevents them from executing their intended encryption/locking behavior.
• The gathered prioritized artefacts were able to identify evasive ransomware samples from benign ones with a low false-positive rate.
• The approach can be enhanced by exploring deferring techniques to delay the execution of contemporary ransomware, and make it generic to operate on various operating systems.