Chapter 11

A BEHAVIOR-BASED APPROACH FOR MALWARE DETECTION

Rayan Mosli, Rui Li, Bo Yuan and Yin Pan

Abstract Malware is the fastest growing threat to information technology systems. Although a single absolute solution for defeating malware is improbable, a stacked arsenal against malicious software enhances the ability to maintain security and privacy. This research attempts to reinforce the anti-malware arsenal by studying a behavioral activity common to software – the use of handles. The characteristics of handle usage by benign and malicious software are extracted and exploited in an effort to distinguish between the two classes. An automated malware detection mechanism is presented that utilizes memory forensics, information retrieval and machine learning techniques. Experimentation with a malware dataset yields a malware detection rate of 91.4% with precision and recall of 89.8% and 91.1%, respectively.

Keywords: Malware, memory forensics, machine learning, handles

1. Introduction

The threat of malware is growing. The proliferation of electronic devices and the ever-increasing dependence on information technology have led to malware becoming an attractive tool for conducting criminal activities. Kaspersky Lab [10] reports that almost 250 million new and unique malware instances were detected during the second quarter of 2016 alone. Although substantial, the report was only able to present the amount of malware detected by anti-viral tools; it was not possible to estimate the total number of malware instances in the wild.

Current malware detection approaches focus on extracting unique signatures from captured malware samples and using the signatures in subsequent sightings of the same malware. This detection strategy is fast and has low false positive rates, but it is easily defeated by modifying

© IFIP International Federation for Information Processing 2017
Published by Springer International Publishing AG 2017. All Rights Reserved

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics XIII, IFIP AICT 511, pp. 187–201, 2017.
DOI: 10.1007/978-3-319-67208-3_11

the malware code via encryption or packing [15]. Another strategy is to use a machine learning model to detect malware based on static malware features [24]. Although this malware detection strategy is more robust than signature-based detection, it can still be defeated [15].

The path to improving the detection of unknown malware started with the shift from signature-based detection to behavior-based detection. Behavior-based detection, which focuses on the activities of malware when it infects a system, can be implemented in two ways. The first involves extracting behavioral traits from the malware code statically; these traits are called malware semantics [6]. The second approach involves running malware in a sandbox environment and dynamically monitoring its behavior. System calls constitute an example of malware behavior that can be monitored dynamically and subsequently leveraged in malware detection [17, 21, 26]. Other behavioral features include file activity [22], registry activity [1] and API calls [27]. Behavior-based detection certainly improves the detection of unknown malware, but it is often slow and resource intensive because it requires running the malware in a sandbox. Furthermore, false positives are often a concern due to the misclassification of benign software that exhibits behavior similar to malware.

Several researchers have applied memory forensics to capture artifacts of malicious behavior that reside in memory [9, 23, 35, 36]. Memory forensics involves the analysis of a memory dump to extract evidence of malicious activity. An investigation using memory forensics has two stages: (i) memory acquisition; and (ii) memory analysis. In the memory acquisition stage, a digital forensic professional obtains a memory image via an acquisition tool such as Memoryze or Winpmem. The memory analysis stage attempts to find evidence of malicious activity using a tool such as Volatility or Rekall. This research employs memory forensics to extract information from memory images that is subsequently analyzed to detect malware.

The malware detection approach discussed in this chapter focuses on handles, abstraction pointers that are used to identify and access system objects without knowing their exact locations in memory. A resource such as a file, registry key or mutant requires a handle to be opened before it can be accessed. The handle must be closed when the resource is no longer required. Failing to release a handle may cause a handle leak, which can result in reaching the upper limit on the number of handles permitted by an application [29]. More than 30 resource object types can be identified using handles. Whenever a process requires such a resource, it must open a handle to the resource. The proposed approach uses the number of handles opened by a process to determine if it is potentially