Available online at www.sciencedirect.com

SciVerse ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Malware target recognition via static heuristics☆

T. Dube[a,*], R. Raines[a], G. Peterson[a], K. Bauer[a], M. Grimaila[a], S. Rogers[b]

[a] Air Force Institute of Technology, Wright-Patterson AFB, OH 45433-7765, USA

[b] Sensors and Information Directorates, Air Force Research Laboratory, Wright-Patterson AFB, OH 45433-7321, USA

ARTICLE INFO

Article history:
Received 12 April 2011
Received in revised form
1 September 2011
Accepted 9 September 2011

Keywords:
Malware detection
Intrusion detection
Malware
Situation awareness
Machine learning

ABSTRACT

Organizations increasingly rely on the confidentiality, integrity and availability of their information and communications technologies to conduct effective business operations while maintaining their competitive edge. Exploitation of these networks via the introduction of undetected malware ultimately degrades their competitive edge, while taking advantage of limited network visibility and the high cost of analyzing massive numbers of programs. This article introduces the novel Malware Target Recognition (MaTR) system which combines the decision tree machine learning algorithm with static heuristic features for malware detection. By focusing on contextually important static heuristic features, this research demonstrates superior detection results. Experimental results on large sample datasets demonstrate near ideal malware detection performance (99.9+% accuracy) with low false positive (8.73e-4) and false negative rates (8.03e-4) at the same point on the performance curve. Test results against a set of publicly unknown malware, including potential advanced competitor tools, show MaTR's superior detection rate (99%) versus the union of detections from three commercial antivirus products (60%). The resulting model is a fine granularity sensor with potential to dramatically augment cyberspace situation awareness.

Published by Elsevier Ltd.

1. Introduction

Malware heuristic analysis techniques generally fall into two distinct categories: static and dynamic (Szor, 2005). Static heuristics generally use non-runtime indicators (Szor, 2005), such as structural anomalies, program disassembly and *n*-grams (Schultz et al., 2001; Kolter and Maloof, 2004; Abou-Assaleh et al., 2004; Kolter and Maloof, 2006; Henchiri and Japkowicz, 2006). Alternatively, dynamic heuristics employ runtime indicators (Szor, 2005) normally obtained in virtual environments, such as commercial sandbox applications (Sunbelt Software, 2009; Norman ASA, 2009; ThreatExpert Ltd, 2009) or emulation capabilities of antivirus products (Szor, 2005; Symantec Corp., 1997).

Despite the success that static heuristics enjoyed during the 1990s (Szor, 2005), today's research and commercial antivirus products heavily favor dynamic heuristics (Szor, 2005; Symantec Corp., 1997; Lee and Mody, 2006; Bailey et al., 2007; Christodorescu et al., 2007; Dinaburg et al., 2008). Antivirus companies use a hybrid of static and dynamic heuristics in their commercial products (Szor, 2005; Symantec Corp., 1997). Most static heuristics require a pristine disassembly, which is difficult to achieve (Moser et al., 2007). Dynamic heuristics avoid this limitation (Moser et al., 2007), because they do not require disassembly, but rather observe program execution in a restricted environment for a specified observation period. Observing program behavior requires all program dependencies to be present (Szor, 2005), which is a stronger requirement

☆ Patent pending.
* *Corresponding author*. Tel.: +1 937 255 3636x4690; fax: +1 937 904 7979.
E-mail address: thomas.dube@afit.edu (T. Dube).

0167-4048/$ – see front matter Published by Elsevier Ltd.
doi:10.1016/j.cose.2011.09.002

for dynamic heuristics than static heuristics. The program may not successfully execute in the test environment when a required runtime library is absent.

Dynamic heuristic methods are generally slower than static methods (Symantec Corp., 1997), because they require an observation duration and emulation overhead. Their performance makes them operationally infeasible to test tens of thousands of unique programs on a single system in tactical situations. Dynamic heuristic analysis is also incomplete (Cavallaro et al., 2008), because no guarantee of observing malicious activity within the observation period exists. Many malware samples require a trigger condition (Cavallaro et al., 2008) to demonstrate their malicious behavior. For example, the Michelangelo virus (Software Engineering Institute, 1997) only executes its payload on March 6, the anniversary of his birth.

This research extends current malware detection research in three important ways. First, Malware Target Recognition (MaTR) demonstrates the utility of using only anomaly and structural static heuristics for robust malware detection, in contrast to previous research using the same sources of information (Schultz et al., 2001). Second, this work also achieves a significant performance improvement over other static heuristic malware detection research (Schultz et al., 2001; Kolter and Maloof, 2004, 2006; Tesauro et al., 1996). For fair comparison, MaTR competes against a retest of the Kolter and Maloof *n*-gram research (Kolter and Maloof, 2004), the best measures seen in similar work, on a larger dataset. Finally, a validation test against a publicly unknown malware set shows MaTR's superior performance over an *n*-gram model and three commercial antivirus products.

The following sections describe related research, MaTR and another static heuristic detection methodology, and hypotheses tested. The next topics covered are the experimental comparison of MaTR with a repeated experiment from other researchers and results illustrating MaTR's performance advantage against a suite of commercial antivirus products. Conclusions summarize this work and include brief synopses of limitations, potential impact, and future research.

2. Related work

While malware detection is a popular research area, nearly all current efforts focus on dynamic heuristic analysis. In static heuristic analysis, many efforts require the successful static disassembly of programs, which is commonly augmented by dynamic methods. Currently, the scope of MaTR is strictly static heuristic analysis, explicitly restricted to features readily available by cursory, non-runtime inspection of a program. This section briefly describes related research in static heuristic analysis of malware.

2.1. *Kephart, Tesauro and Arnold*

IBM researchers Kephart, Tesauro and Arnold provide the seminal research in *n*-gram analysis of malware. These *n*-grams are byte sequences of length *n* that occur in the target, which theoretically represent program structural components and fragments of instructions and data. They examine the use of *n*-grams in automatic signature extraction (Kephart and Arnold, 1994) for malware variants as well as for generic detection (Tesauro et al., 1996; Arnold and Tesauro, 2000).

While searching for methods to automate signature extraction for new variants of known malware, Kephart, et al. discover the utility of *n*-grams for generic malware detection (Kephart and Arnold, 1994). By determining the probability of finding specific *n*-grams in malicious and non-malicious programs, the authors fabricate a generic malware detection classifier.

Tesauro et al. successfully use neural networks to detect boot sector viruses (Tesauro et al., 1996). They manipulate the decision threshold boundary to increase the cost associated with false positives as they cite that a single false positive reading likely affects thousands of systems. Despite significant computational and space constraints as well as a small sample size for training and validation, they achieve a false positive rate of less than 1% while detecting over 80% of unknown boot sector viruses.

They train the network with trigrams (3-byte strings) that undergo a novel feature selection process. Initially, they canvas the entire sample corpus for trigrams and eliminate all that are common to both the malicious and non-malicious sets. Moreover, they reduce the list of trigram features to the set where each malicious training sample contains at least four trigrams. This selection process leads to a three order of magnitude feature reduction.

Expanding on their previous work, Arnold and Tesauro incorporate a voting system on multiple trained neural networks (Arnold and Tesauro, 2000). By training multiple networks with distinct features not used in others, they effectively avoid the major pitfall associated with heuristic scanners, high false positive rates. Their assumption is that these disparately trained networks rarely produce identical false positives. Szor cites that the Arnold and Tesauro network research has such a low false positive rate that Symantec incorporated it into its antivirus product default scanning (Szor, 2005).

2.2. *Schultz et al*

Schultz et al. make key contributions by testing three different sources of features to identify malware (Schultz et al., 2001). In their first approach, they examine information from the portable executable (PE) header as features, such as import libraries and the number of imported functions from those libraries, with Cohen's improved rule learning algorithm called RIPPER (Cohen, 1995). This method requires unpacking the samples before evaluation to reveal the true imports, but the authors do not refer to this difficult step. The second approach uses strings found in the binaries as features, which is again problematic without first unpacking.

The third method captures byte sequences expected to translate loosely to a representation of instructions, data, or both. Without first unpacking the binary, however, in the best case one would expect such byte sequences to most likely represent general program structure, unpacker stub instructions and offsets, unpacker data, or other global information. In the worst case, these byte strings may represent packed information, which is essentially indecipherable data.