Know Your Enemy: Surveying the Problem of Malware Classification

Patrick Austin
Department of Computer Science and Engineering
University of Nevada, Reno
paustin@nevada.unr.edu

***Abstract*—The proliferation of malicious software, or malware, as a vehicle for compromise of computer and network security costs individuals, businesses, and governments worldwide billions of dollars every year. Systems designed to detect malware in order to remove it and systems designed to prevent malware installation entirely are necessary to combat this threat. Yet malware design and behavior change and evolve continuously, and anti-malware systems must utilize minimal resources yet achieve maximal accuracy. A robust understanding of how malware is designed, how malware interfaces with operating systems, and how the appearance and behavior of malware can be detected will be key to any effective anti-malware strategy. In this paper we survey the topic of classifying software as legitimate or belonging to a particular family of malware based on appearance and behavior.**

I. Introduction

The use of malicious software, or malware, as a means to compromise computer and network security and perpetrate cybercrime is far from a new phenomenon. The earliest known malware for PCs dates back to the mid-1980s; malware targeting the Windows operating system began to spread widely via internet in the mid-1990s; large-scale security breaches based on malware attacks were facts of life by the early 2000s and have been endemic in networked computer systems ever since [1].

The manner in which malware generally operates is no real mystery. Like any piece of software, malware interfaces with the operating system in order to access computer resources. Malware is different only insofar as that access is done with deliberate and malicious intent to harm, aiming to compromise or act against the well-being of the user or the system. Under the larger umbrella of malware, researchers and security experts have defined a number of prominent sub-types widely used by attackers in the modern day such as adware, ransomware, and botnet recruitment malware [2]. Researchers also classify malware into categories based on the fashion in which the malware spreads. This is the origin of the classic distinction between a virus, which rides on and spreads itself to other programs, executing when they execute, and a worm, which is standalone piece of malicious software that typically spreads via access to a computer network [3].

Despite the long history of the problem and a significant body of prior research, there is growing reason to be concerned about malware proliferation and growing demand for effective, economical solutions. By the accounting of the analytics firm Accenture, the incidence of cybersecurity breaches in business environments increased by 27.4% between 2016 and 2017 alone; likewise, the average cost of achieving cybersecurity for firms increased by 22.7% [4]. Malware related breaches, and anti-malware cybersecurity efforts, constitute no small part of these costs. In the government sector, recent months have seen major cyber-systems of entire cities and nations held hostage by ransomware [5][6]. The apparent ease with which vital infrastructure can be targeted by and infected with malware is even more alarming when the possibility of cyberwarfare perpetrated by state actors is taken into account.

In the face of the continuing escalation of cybercrime activity, and in light of the further threat of cyberwarfare, we believe now is an opportune moment to review the structure and operation of different types of malware beginning from a real-world, low-level, operating-system oriented perspective. Furthermore, we aim to provide a picture of anti-malware strategies through the lens of the problem of malware classification.

How can we categorize malware in terms of behavior? How can the behavior and structure of malware be observed and differentiated from the behavior and structure of normal software? How can this decision be made accurately, while minimizing both computation and the need for costly human analysis and intervention? How can we move beyond strategies which merely identify known malware based on signatures of observed behavior, and work towards next-generation, future-proof approaches which are capable of identifying new types of malware quickly, efficiently, and accurately? In this paper we survey the available body of research on these important questions.

II. Challenges

There are a number of challenges which must be addressed in order to realize a comprehensive, robust, flexible, and economical anti-malware strategy. The high degree of difficulty of these problems is evidenced by the reality that malware detection and prevention is still a prominent and unsolved problem in computer security to this day, and the field remains reliant on a mélange of heuristic and ad-hoc approaches. In this section we discuss what makes several of these challenges so serious and offer a brief sketch of the problem from a threat model perspective.

An important initial challenge is that malware detection schemes have extremely high costs for false negatives, and fairly high costs for false positives. In other words, accuracy in terms of a high true positive/true negative rate is paramount. False negatives reflect malware that is being allowed to freely execute on the target system and an effectively compromised device. False positives reflect legitimate software being mislabeled as malware and likely being prevented from functioning as a consequence, which can be highly onerous to the user and can be extremely costly to an organization [7]. Yet at the same time, high quality malware detection software must also have low costs in terms of computational overhead when operating on a real-world system lest the impact of the detection scheme itself on performance prove burdensome to users.

A good malware detection scheme will also minimize the amount of costly and failure-prone human expertise and supervision necessary in the security process. In fact, as we will show in the next section, there is a vibrant area of research concerning machine learning as applied to malware classification. Such approaches provide powerful tools but are not yet capable of providing comprehensive security without some level of supervision [8]. Cybersecurity professionals are still vital to the process of analysis of emerging malware threats, the creation of malware signatures for those threats, and the general coordination of anti-malware strategy within organizations. While humans cannot presently be removed from anti-malware systems, and arguably should not be removed completely considering the complexity and dynamism of the problem, minimization of human effort would be likely to reduce cybersecurity costs and increase accuracy in anti-malware systems.

But perhaps the core challenge of malware detection and classification is the polymorphic and metamorphic nature of malware behavior over time. A comprehensive scheme that detects all previously observed malware can still fail to recognize new malware targeting new vulnerabilities, or even old vulnerabilities being exploited in new ways. An attacker can morph and mask malware’s true behavior in subtle ways, making malicious behavior difficult to detect. As the operating systems on which computers function change, evolve, come into being, and fall out of favor, the suite of vulnerable system calls available for malware to exploit likewise morphs continuously [9]. An ideal anti-malware scheme, then, is one that is capable of morphing and learning in a fashion that is capable of responding to new and concealed malware threats, not just handling threats that have already been observed and broken down in terms of signature behaviors.

Given these serious and complex challenges, identifying threats and vulnerabilities from an attacker’s point of view according to threat modeling is a vast task whose specifics will depend on the attacker’s platform of choice and specific objective. Therefore, we can only broadly and briefly sketch a threat model in the scope of this paper. The core advantages of an attacker using malware and the core vectors for attack are inherent in modern computer systems: (1) computers are widely networked within organizations and via internet, allowing for malware to spread quickly, unpredictably, and on a vast scale; (2) modern operating systems change and evolve continuously even as malware changes and evolves continuously, constantly giving the attacker new avenues to exploit; (3) private individuals and users within organizations are often given a wide degree of latitude in terms of network access and command of system resources, which malware can then exploit; (4) given the realities of the previous points, anti-malware software and systems are often limited in resources and reactive in nature, utilizing best-estimate heuristic approaches to detection and responding poorly, if at all, to emergent threats.

These fairly intractable challenges, and this wide-open threat model, should serve to give a sense of how serious the problem of malware detection is from a security perspective.

III. Existing Research

In this section we discuss the existing body of research in the anti-malware area, focusing on malware detection. We will survey interesting, innovative, and foundational papers and offer some brief perspective on the merits and demerits of each.

A. Surveying the Surveys

First, we examine some papers that survey malware classification generally. Jacob, Debar, and Filiol’s taxonomy of approaches to behavioral detection of malware is perhaps the foundational study of our own survey, as it explicates a key distinction between behavior-based malware detection approaches and appearance-based malware detection approaches [10]. Behavior-based detection analyzes what malware does or how it functions in execution, such as the types of system calls it makes. Appearance-based malware detection merely looks at the syntactical structure of the malware, usually with humans studying and deriving signatures via code analysis. The authors argue that appearance-based detection is a doomed strategy in the long term, as malware exhibits increasingly complex and rapid evolutionary behavior, including random, automatic mutation. Ultimately, they argue that more flexible, less human-intensive behavior-based approaches will come to dominate the malware detection field, and they survey a variety of such approaches.

This distinction between behavior and appearance-based approaches can be used to contextualize and understand many of the individual strategies we will go on to analyze. The utility of the framework the authors provide is hugely useful in giving context to strategies in the field and constitutes a considerable merit. On the demerit side, one might wonder if the authors have declared the death of appearance-based approaches prematurely; their paper is nearly ten years old, and as we will see appearance-based approaches continue to see wide use and considerable success. Yet it is also true that there are robust up-and-coming behavioral approaches available in the literature.

Jacob et al. go on to discuss a further categorical divide between static and dynamic approaches. But we find Gandotra, Bansal, and Sofat's survey on the topic to provide a more up-to-date and succinct explanation of this distinction [11]. Static approaches involve analysis of code without execution, and dynamic approaches rely on executing the code and observing its behavior. Thus, this is a roughly analogous distinction to the one between appearance and behavior.

The significant merit in this paper is that it draws on more recent research, with an emphasis on machine learning as applied to malware detection in both the static and dynamic areas. Gandotra et al. ultimately advocate for the creation of hybrid approaches that combine the efficiency of static strategies with the flexibility and resilience of dynamic ones.

B. Static Approaches

One testament to the possibility that Jacob et al. declared the death of appearance-based approaches prematurely is the continuing prominence and success of static analysis of malware demonstrated in the literature. Here we analyze a variety of interesting and influential papers operating in the static domain of malware detection that innovate the area beyond a basic signature-based approach. The two apparent issues in static analysis are the features that should be extracted from a binary that can be used to determine whether the underlying code is malicious, and the classification strategy which uses those features to make a final decision. As we will see, researchers have explored creative new directions in both respects in the last several years.

To begin with a relatively straightforward approach, Tian, Batten, and Versteeg attempt to classify malware into families using the sequence of function lengths in the code as the deciding property [12]. These vectors of function lengths are then used in clustering to generate malware families. The advantage of using such a simple property of the code is scalability and speed, though even the authors admit the accuracy achieved leaves much to be desired. While speed and simplicity are merits, this approach is obviously limited, and intended to serve as a single tool with a particular niche in a larger toolkit.

Santos et al. propose a method that extracts and classifies based on the frequency of appearance of opcode sequences in the assembly code for potential malware [13]. Although this is a static approach where no code is executed, the authors tout a significant and compelling advantage: their model performs well at detecting unknown malware that has not previously been exposed to the detection scheme, which is a typical weakness of signature-driven static approaches. This is a compelling finding that suggests the potential for endurance of static schemes that other authors in the area have, perhaps, underestimated.

Neural networks can provide one strategy for the classification side of the malware-detection problem and are in vogue in machine learning in general, so it should come as no surprise to see authors experimenting with their use. In one very recent study, Le et al. experiment with neural networks, simply using the raw binary of the software as the input [14]. They achieve very high accuracy with low computational overhead beyond the initial creation of the neural net and boast the ease of using the system for non-domain experts who may have need to analyze software to see if it is malicious, such as those in law enforcement fields. However, the authors train and test their system on known malware; this approach is unlikely to be robust against the types of rapid self-encryption and random mutation that highly advanced modern malware can employ, as these factors can massively distort the raw binary.

Ly, Kha, and Hwang offer another neural net-oriented approach [15]. Where Le et al. used a convolutional neural net structure, Ly et al. use a network with both convolutional and recurrent neural layers. The code is transformed into a sequence of blocks of assembly code to use as input to the network. This more complex neural net and more fine-grained input will likely stand up better to unknown malware than the one generated by Le et al., which is a noteworthy merit along with the high rate of accuracy, which the authors profess to be on par with or exceed state-of-the-art signature approaches. However, a demerit is that this paper also neglects to actually test or explore the approach against unknown, emergent malware, as was done in the Santos et al. study [13].

Arp et al. offer a tool called DREBIN that differs in both classification strategy and code parameters examined; notably, their study also focuses on malware targeting the Android mobile operating system where previous approaches were PC-centric [16]. The authors acquire a number of features about the app in question based on code analysis, such as number of permissions requested by the app and the number of suspicious or problematic API calls. This data is then transposed into vector space and classified via a linear SVM scheme. The authors emphasize two additional aspects that the previous studies did not: human interpretability of results and the performance limitations of mobile platforms. DREBIN returns a rationale report explaining the reasoning behind the classification made in human terms, which is difficult if not impossible in a neural net approach. We will have more to say about the challenges posed by malware detection for mobile platforms in the open research challenges section.

Finally, we consider a creative study by Nataraj et al., who generate graphical visualizations of malware binaries and use them for classification using algorithms from the domains of image processing and computer vision [17]. Merits include novelty, speed, lack of necessity for disassembly of code, and the potential for unanticipated synergy arising from the crossing of two fairly disparate sub-disciplines in computer science. The major demerit is that like most of the static approaches analyzed here, this strategy is likely to fare poorly against zero-day malware attacks.

In summary, the static malware-detection area is a vibrant one, using a variety of approaches which generally secure both high degrees of speed and accuracy. However, most (though not all) of these strategies share in the weakness of signature-based approaches to zero-day attacks and obfuscated, mutating code. The continued prominence of research in the static area suggests that similar strategies will be with us for some time to come.

C. Dynamic Approaches

While the foregoing section treated just a handful of the many recent studies of static malware-detection approaches, the dynamic area is far sparser. While we have much more to say about this observation in the next section, here we note a few significant studies that do discuss and experiment with dynamic detection tools.

Egele et al. offer a survey of the field of automated dynamic malware analysis tools [18]. Perhaps the most obvious and significant observation we take away is the level of complexity and difficulty that dynamic approaches bring into play. The authors thoroughly elaborate the range of options available in dynamic detection to execute code that may be malicious in a secure fashion, such as various types of emulation or virtualization, and none of them are computationally cheap or simple from the point of view of an ordinary user, where static approaches were often both. Even when handled correctly, these approaches have pitfalls: the authors describe the troubling case of malware exploiting a CPU bug which exists on the real hardware but does not exist on the virtualized representation of the hardware. Likewise, the tools they describe that are used for automated dynamic analysis are all complex and computationally pricey pieces of diagnostic software firmly intended for use by experts in the malware security field, requiring deep knowledge of operating systems to utilize and/or interpret effectively. In other words, Egele et al. illustrate how far many dynamic approaches appear to be from readiness for the mass market.

Brumley, Hartwig, and Liang offer a tool they call Minesweeper that uses dynamic execution to try to find hidden trigger conditions that might cause stealthy malware to become active, such as the occurrence of a certain date or certain network commands aimed at starting denial of service botnet attacks [19]. The authors frame this as a step towards automating a stage of malware testing that was often done in a tedious, manual fashion previously. This is in the wheelhouse of many of the automated approaches presented by Egele et al. insofar as it is a tool that assists security professionals in testing malware in a laboratory environment. Tools like this are certainly important and useful but are unabashedly not for mass consumer use.

Likewise, Lanzi, Sharif, and Lee offer a tool called K-Tracer, which analyzes code running at the level of the Windows kernel to detect the behavior of active rootkits [20]. Rootkits are a type of malware which uses administrative authority on the system to manipulate system behavior in ways that make malware behavior harder to track and malware itself harder to eliminate. K-Tracer detects hooks placed in the operating system where a rootkit tries to divert execution from legitimate operating system behavior to malicious malware code. This is another valuable tool for security experts.

To provide a general characterization, dynamic approaches usually offer tools oriented towards security experts studying malware in laboratory environments. This is challenging and important work to be certain, but it falls short of the efficient, automated, consumer-ready approaches that we observed when considering static approaches. In the next section we comment on some of the ramifications of this observation.

IV. Open Research Challenges

In this section we emphasize a few takeaways from our survey of the literature and assess where the malware classification field is headed in the context of larger computing trends. We believe these findings suggest important open challenges meriting further research.

Perhaps the first and most prominent conclusion to draw from a survey of the literature is that dynamic approaches have, in general, not yet lived up to the hype. In terms of volume alone, our literature search has revealed that the pace of research in the static area significantly outstrips that of research in the dynamic area. There are many quite logical reasons for this. Static approaches are generally faster, simpler, and easier to understand. They are often highly accurate as well, especially against known malware where a signature can be created or a supervised learning model can be trained. As Santos et al. showed, some up-and-coming static strategies may even prove resilient to the historical weaknesses of the approach, namely detecting obfuscated and zero-day malware attacks [13].

However, on the other hand, many of the static approaches surveyed do fare little better than a signature-based approach would in the face of zero-day malware. This ambivalence to the prospect of emergent, camouflaged, or metamorphic malware is dangerous. Stealth malware techniques have become significantly more robust and complex over time, with no signs of slowing down; as mutating malware becomes more and more the norm many static approaches will run the risk of failing to keep up [21].

Yes, existing dynamic approaches are surely powerful and useful as analysis tools in the hands of security experts. But dynamic approaches were supposed to be the ones flexible and robust enough to handle the challenges of stealth malware in general and mitigate the need for costly security experts in the first place. That goal clearly remains to be achieved.

The creation of dynamic approaches that can prove accurate, secure, robust, and efficient enough to run on consumer hardware in widely-used anti-malware systems remains the most significant open research challenge in this area in our assessment. Failing that, hybrid approaches should be explored that incorporate dynamic and static aspects to the extent that is tolerable from a performance perspective. Failing that, we suggest work towards advancements in static approaches against their historical weaknesses, which could more or less obviate the need for dynamic strategies in the first place. Santos et al. have suggested some reason for hope on this front [13].

It also seems wise to consider the malware-detection problem in the context of larger trends in computing in order to consider open challenges that are particularly pressing. The rise of mass mobile computing, the proliferation of IoT (Internet of Things) and smart devices, and the emergence of cryptocurrency mining malware are three such phenomena we will briefly discuss.

Mobile platforms have experienced explosive growth in userbase the last decade [22]. Likewise, malware aimed at mobile platforms has become increasingly common and costly to users and organizations, constituting an increasingly pressing concern [23]. The relatively lower power of battery-limited mobile systems and the challenges posed by trojans tailored to mobile platforms make this an area with unique challenges from a malware-detection perspective.

We have already discussed the DREBIN platform for malware detection on Android; the authors particularly emphasize the unique challenges posed by performance requirements on the mobile platform [16]. Consider dynamic approaches in this context, for example. If executing code to analyze its behavior is infeasibly challenging and computationally expensive on much more powerful desktop platforms and operating systems, it is especially difficult to see the approach taking off in the mobile environment. In light of this and similar concerns, one pressing open research challenge is the adaptation of PC-oriented anti-malware strategies to mobile platforms, which will likely emphasize further optimization of existing algorithms, as well as innovation of new mobile-focused strategies, all in order to minimize the anti-malware system's performance footprint.

The security issues at stake with the massive spread of IoT devices, particularly their role in empowering botnets and denial of service attacks, have been explored in detail elsewhere [24]. The unchecked spread of malware to these devices is a root cause of IoT-based network attacks and crafting a defense strategy is a pressing and open challenge.

IoT devices are highly heterogenous and non-standardized in terms of interface and hardware, with highly variable computing power, operating system design, and use of proprietary software. This makes IoT systems difficult to patch and devise anti-malware software for. All of these issues make the creation of a standardized, robust anti-malware approach for IoT a real challenge in search of a research breakthrough [25].

Finally, to explore one interesting and recent issue at the forefront of the anti-malware battle, we consider the rising occurrence of malware that is designed to use the host's system resources, especially the graphics card if it is present, to mine cryptocurrency for the attacker [26]. Mittal et al. offer a compelling study of security vulnerabilities of the GPU, where defense against attackers has historically not been emphasized in terms of design as it has been in the CPU and memory fields [27]. They propose several short-term techniques to improve GPU security, but comprehensive changes in design which take these new threats into account without significantly sacrificing GPU performance are a significant research challenge situated on the forefront of current anti-malware trends.

V. Conclusions

In this paper we have explored the impetus for and challenges posed by malware classification. We have offered a broad survey of economic and practical issues at stake with current anti-malware approaches and the threat model they address. We have offered a sketch of the ways in which malware is and has been detected in consumer anti-malware software, as well as tools and approaches used by security experts in the lab. We have explored several cutting-edge approaches to the classification of malware that could form the core of next generation malware detection and prevention systems. We have also discussed a variety of open challenges, most centrally that of moving beyond static history-driven and signature-based approaches into more dynamic ones that are more capable of addressing the constantly changing nature of malware in real-world cyber-systems, even with minimal human intervention.

Before a problem can be addressed, it must be understood. Our hope is that a broad understanding of malware, and an assessment of the successes and failures of the current milieu of anti-malware strategies, can constitute a first step toward more effective anti-malware solutions.

References

[1] Milošević, Nikola. "History of malware." *arXiv preprint arXiv:1302.5392* (2013).

[2] "Malwarebytes 2017 state of malware report." Malwarebytes. January 2018, https://www.malwarebytes.com/pdf/white-papers/CTNT_2017 stateofmalware/

[3] Aycock, John. Computer viruses and malware. Vol. 22. Springer Science & Business Media, 2006.

[4] "Accenture 2017 cost of cyber crime study." Accenture. September 2017, https://www.accenture.com/us-en/insight-cost-of-cybercrime-2017

[5] "Massive cyberattack hits Europe with widespread ransom demands." Washington Post. June 2017, http://wapo.st/2sN9t3y

[6] "A cyberattack hobbles Atlanta, and security experts shudder." New York Times. March 2018, https://nyti.ms/2GgRAyl

[7] Parsons, Thomas. "A false positive prevention framework for non-heuristic anti-virus signatures." (2009).

[8] Gandotra, Ekta, Divya Bansal, and Sanjeev Sofat. "Malware analysis and classification: A survey." Journal of Information Security 5.02 (2014): 56.

[9] Thorsten Holz and Herbert Bos. Detection of intrusions and malware, and vulnerability assessment. Springer Science & Business Media, 2011.

[10] Jacob, Grégoire, Hervé Debar, and Eric Filiol. "Behavioral detection of malware: from a survey towards an established taxonomy." Journal in computer Virology 4.3 (2008): 251-266.

[11] Gandotra, Ekta, Divya Bansal, and Sanjeev Sofat. "Malware analysis and classification: A survey." Journal of Information Security 5.02 (2014).

[12] Tian, Ronghua, Lynn Margaret Batten, and S. C. Versteeg. "Function length as a tool for malware classification." Malicious and Unwanted International Conference on Malicious and Unwanted Software, IEEE (2008).

[13] Santos, Igor, et al. "Opcode sequences as representation of executables for data-mining-based unknown malware detection." Information Sciences 231 (2013): 64-82.

[14] Le, Quan, et al. "Deep Learning at the Shallow End: Malware Classification for Non-Domain Experts." Digital Forensics Research Workshop preprint (2018).

[15] Ly, Vu Duc, Nguyen Trong Kha, and Seong Oun Hwang. "DeepMal: Deep Convolutional and Recurrent Neural Networks for Malware Classification." IET Research Journals (2015): 1-9.

[16] Arp, Daniel, et al. "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket." Ndss. Vol. 14 (2014).

[17] Nataraj, Lakshmanan, et al. "Malware images: visualization and automatic classification." Proceedings of the 8th international symposium on visualization for cyber security. ACM (2011).

[18] Egele, Manuel, et al. "A survey on automated dynamic malware-analysis techniques and tools." ACM computing surveys (CSUR) 44.2 (2012): 6.

[19] Brumley, David, et al. "Automatically identifying trigger-based behavior in malware." Botnet Detection. Springer, Boston, MA, 2008. 65-88.

[20] Lanzi, Andrea, Monirul I. Sharif, and Wenke Lee. "K-Tracer: A System for Extracting Kernel Malware Behavior." NDSS (2009).

[21] Rad, Babak Bashari, Maslin Masrom, and Suhaimi Ibrahim. "Camouflage in malware: from encryption to metamorphism." International Journal of Computer Science and Network Security 12.8 (2012): 74-83.

[22] “Global mobile OS market share in sales to end users from 1st quarter 2009 to 2nd quarter 2017.” Statista. August 2017, https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

[23] Felt, Adrienne Porter, et al. "A survey of mobile malware in the wild." Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. ACM (2011).

[24] Zhang, Zhi-Kai, et al. "IoT security: ongoing challenges and research opportunities." Service-Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on. IEEE (2014).

[25] BalaGanesh, D., Amlan Chakrabarti, and Divya Midhunchakkaravarthy. "Smart Devices Threats, Vulnerabilities and Malware Detection Approaches: A Survey." European Journal of Engineering Research and Science 3.2 (2018): 7-12.

[26] “Cryptocurrency-mining malware: why it is such a menace and where it’s going next.” ZDNet. April 2018, https://www.zdnet.com/article/cryptocurrency-mining-malware-why-it-is-such-a-menace-and-where-its-going-next/

[27] Mittal, Sparsh, et al. "A Survey of Techniques for Improving Security of GPUs." arXiv preprint arXiv:1804.00114 (2018).