Milware: Identification and Implications of State Authored Malicious Software

Trey Herr, Eric Armbrust
The George Washington University

Abstract

The difference between state and non-state authored code is typically described in vague terms of sophistication, contributing to the inaccurate confirmation bias of many in the policy community that states simply 'do it better'. Leveraging the results of reverse engineering several malware samples, including `Sandworm` and `Tinba`, this paper is an interdisciplinary effort to distinguish between state authored code, milware, and that produced by non-state actors, malware. Working through this initial set of samples, the paper describes a new analytic framework for differentiating state authored code from other samples. This MAlicious Software Sophistication or MASS index relies on a set of characteristics which describe the behavior and construction of malicious software: propagation to and within a target network, exploit severity, and payload customization. Highlighting these distinctions then serves to support a larger analysis of the policy implications these separate categories of malicious code have. By identifying a systematic difference between non-state authored code and that created by states, this pilot project is an effort to generate a new analytic asset for the technical community and highlight attendant policy implications.

1 Introduction

Pervasive development and use of milware constitutes not only a direct technical challenge of decomposing and analyzing well obfuscated code but also threatens a set of key assumptions underpinning the current information security research and defense paradigm. States operate in a different legal regime than criminal groups and individuals, inverting the power relationship between attacker and defender and altering what is possible in the defense against and prosecution of sources of information assurance threats. This paper develops the MASS index as a rudimentary tool for analysts to distinguish between state and non-state authored code but its primary contribution is to highlight five major implications of milware:

Public disclosure is not as effective. States have little to fear from public disclosure of their activities and so the traditional paradigm of revealing tactics and techniques to dissuade attackers and aid defenders is less effective.

States may be doing R&D for all malicious actors. States have far more resources to develop new techniques and exploits than non-state actors. The eventual proliferation of this code by individuals and criminals means the state of the art will continue to advance, funded by governments.

Even where they do not build the capabilities, states may be distorting the market. State's financial resources may price defenders out of the market for exploits and even bring new sellers into play.

Existing legal tools presume the targets of prosecution are non-state actors. Law enforcement targets individuals and non-state groups but states are operating under this same legal regime, allowing them to act with relative impunity.

Milware privileges access over effects. States have taken advantage of the current emphasis on defensive and information assurance standards over software developer liability.

Before understanding the implications of this distinct category of code, our first task is to recognize its existence. Previous work presented has attempted to move beyond the simple sophisticated/unsophisticated dichotomy and succeeded in developing a metric that measured social engineering tactics. [1] We advance this scholarship by focusing on the functional characteristics malicious code and comparing the work of state and non-state actors to better understand what is common to malicious software and what depends on the unique operational demands of state versus non-state actors.

Starting with a description of the samples analyzed and our selection process, this paper explains the characteristics we developed to delineate between milware

and malware. Walking through an example reversing process, the paper describes the key features in the sample, highlighting where they tend to differ between the categories. Milware has a higher frequency of 0day use and generally employs vulnerabilities with a more severe CVE classification. While the targets of malware, and milware overlap, their movement within target networks differs; milware propagates to particular high value nodes in a deliberate, often human directed, fashion while malware reflects an automated, scattershot approach.

2 Background

Malicious software analysis has long been focused on individually functional components in code, using information gleaned from particular modules to describe the function of an entire sample in a sound but somewhat limiting bottom up approach. Our approach attempts to work top down, establishing three broad components of all malicious samples; propagation methods, exploits, and payloads. [2]

Propagation is the means of transporting malicious code from origin to target; this could be as simple as a mass email for spear phishing attacks or as complex as a crafted dropper module. Exploits act to enable the propagation method and payloads operation. The payload is code written to achieve some desired malicious end.

The three components work in concert but each have substantially different roles; the propagation method moves the entire software package from origin to target while the payload is written to manipulate system resources and create some effect on a computer system. Exploits are not themselves malicious, but rather act to support the propagation method and payload. Achieving root access on a machine does not create an effect by itself; rather it makes such an operation possible, credentials theft for example. This terminology is at odds with some usage, which combines the payload's function into the exploit and uses the combination as a verb, to 'exploit' a system. [3]

This approach however limits our ability to categorize and understand malicious code; the two are logically distinct both in sequence and form as exploits are written to a particular vulnerability present in some piece of software while payloads are written to achieve a particular effect. This distinction, between writing to the target and writing to the effect, make for divergent practices in developing, selling, and integrating exploits and payloads. Without one or several exploits, a payload would almost never be able to execute on a target computer - these exploits serve to manipulate the target system into giving malicious code access and user privileges necessary to function. Each of the three components of this framework [2] has a distinct purpose and works in combination with the others to constitute a malicious package.

Differentiating between sophisticated and unsophisticated samples is not a new area of work in the information security community; static and dynamic analysis techniques have been in use and evolving for several decades. [4, 5, 6, 7, 8] The origin of samples has come to the foreground more in the past decade as efforts have shifted to understand the nature of threats and their activities beyond the network perimeter. As part of this change, the motivations of attackers and their political status has become a factor of interest. [1, 9] Analytically, the next step is understanding the relationship between the complexity and capability of a piece of malicious code and the nature of its authors. We do this by focusing on the discrete components outlined above, the propagation method, exploits, and payload, an approach which has seen some use. [10]

3 Data Collection

We selected a set of malware and milware samples for both accessibility and their chronological proximity. Our ability to work directly with the milware samples was limited by agreement and so relies in part on open source documentation. Malware selection was driven by three key factors: scope of impact, availability of samples, and knowledge of lineage. Using malware samples with a long history of use allowed us to consider crossover between samples and select more complete instances of code.

To achieve the clearest possible distinction between milware and malware, we selected two samples from each category. Malware samples included Game Over Zeus (`GOZ`) and Tinybanker (`Tinba`) while `Sandworm` and code used in the Sony Attacks were selected as our milware samples. Each code has had wordlwide impact of one type or another and propagates using a different method.

4 Methods

To gain a fuller understanding of our samples, we undertook to reverse engineer each in a controlled lab environment using both static and dynamic analysis. This process leveraged existing and widely used tools, including IDA Pro [11], BOCHS [12], WinDBG [13], Immunity [14], and SysAnalyzer [15], in a live running environment. These tools helped document the changes our samples made to the testing system and parsed each binaries, enabling us to work with readable machine code. This analysis also considered how samples interacted with their surrounding network so we included TCP-

DUMP [16] and WireShark [17] to gather and analyze outbound traffic.

4.1 Milware

The Sandworm group, a Russian linked state sponsored or supported hacking organization, has been running a multinational intelligence gathering campaign on Western states from as early as 2009. [18] The propagation method typically observed with Sandworm attacks involves spear-phishing email to the target with an attached powerpoint document that, when opened, will trigger CVE-2014-4114 to compromise the target system. [19]

The Sandworm attacks targeted largely NATO states and affiliated countries including the United States, United Kingdom, and the Ukraine; the breadth of this spread allows us to to analyze the propagation patterns behind a single case of milware exploiting a single CVE. The payload employed similar functionality to several malware samples, including the GOZ code examined in this paper, through its selection of communication tactics with a command & control server. This gave us a unique opportunity to compare and contrast the specific functionality between the two pieces of code.

The second milware sample under analysis is from the recent Sony attacks. While not a coherent, multi-target attack like Sandworm attacks, this particular chain of events provides insight into the functional process of an attack by state affiliated actors. The world was able to observe every step of the compromise, from infiltration through social engineering to the compromise of Sonys infrastructure to the subsequent exfiltration and termination of corporate data. These milware samples exhibit particular characteristics in their propagation methods, exploit selection, and payload behavior that help delineate them from more conventional malware binaries.

4.2 Malware

GOZ is piece a of malware that, when installed on a target system, will establish a P2P connection based on the Kademlia protocol. [20] The binary propagates using phishing emails containing links that redirect to compromised web hosts that serve malware to the target, using the Blackhole Exploit Kit to leverage vulnerabilities in the victim's web browser. A dropper module then deposits a piece of code on the target system to execute the GOZ payload which attempts to contact the CC server. [21]

The selection of GOZ was based off the observed similarities to Sandworm in botnet functionality and provided a basis to contrast the the two with regard to their communication protocol. [21, 18] GOZ's use of the Blackhole Exploit Kit demonstrates a common feature of malware; reliance on pre-existing third-party exploit packages. This is directly at odds with Sandworm's use of a more particularly selected and applied exploit to gain access to target systems.

This use of an exploit kit is nothing new an indeed is present in the second malware sample analyzed in this paper. Tinba is a banking trojan that has become notorious for its incredibly small size, clocking in at only 20kb. Tinba's propagation method uses a set of compromised websites to distribute code to victim systems, either via linking to a web host in a phishing email or by targeting certain users browsing habits; the group behind Tinba is notorious for compromising pornographic web sites and indeed this now serves as the main mode of Tinba propagation. [22] Once the victim visits the compromised web host, much akin to GOZ, Tinba uses the Blackhole Exploit Kit enable installation of a dropper on the victim's system. Lastly, this dropper downloads and installs the Tinba payload from a predetermined list of URLs. Tinba, unlike GOZ, does not emphasize on botnet functionality in its payload, instead it focuses on skimming victim's banking information through the use of web-injects and function hooks. [23]

Both samples of malware differ in the type of data targeted and the manner of exfiltration but each makes use of the Blackhole Exploit Kit to successfully propagate and execute their respective payloads. This fact demonstrates some of the limitations of developing code for malicious purposes - designing several payloads may well be less costly in terms of information and human capital than keeping abreast of the latest vulnerabilities present on a particular target systems and continually developing new exploits to match. This encourages malware's relatively broader propagation - the proportional rate of success for all attacks is lower than for more targeted efforts but the victim pool is made substantially broader to compensate. Each group distributing malware, while they might find it cost-effective to develop an original payload, appear willing to use the same vulnerabilities found in many other samples of malware to gain code execution on the target system.

This breadth of propagation is further reinforced by the selection of a kit like Blackhole, which targets browsers with a variety of potential exploits in any given iteration. Blackhole targets victim interactions with the web browser; consequently, by loading an iframe containing malicious JavaScript into the victim's browser, Blackhole will detect which exploits the victim is vulnerable to and craft an environment wherein the victim will be exposed to a condition that triggers the exploit and executes the associated payload. [24]

While not a core sample, the Blackhole Exploit Kit is taken into consideration in the analysis due to its signifi-

cance in GOZ and Tinbaand prevalence in other malware as a means to support propagation.

4.3 Sample Analysis

Our methods of analysis focused on differences and similarities amongst the gathered samples. In addition to traditional reverse engineering techniques, we also chose to study how the samples acted in a network environment similar to those found in corporate and personal targets. In so doing, we were able to observe how malware propagated both to, and within, a target differently from milware. Comparing Sandworms use of CVE-2014-4114 with Tinba and GOZ's use of the Blackhole Exploit Kit, we were able to identify key differences between their respective propagation methods.

Starting with GOZ and Tinba was, we determined the life cycle of each sample by using IDA Pro to examine what the code from each malware sample looked like, what it was packed with, if it was encoded, and what techniques were used to compromise and gain root access to the system. At higher level, to measure differences such as impact on the system and internal network propagation, we simulated a network of computers and watched how our samples interacted with each node on the virtual network. The paper's goal is to create a more robust series of identifiers for which to classify malicious samples. To do this we targeted core similarities shared between each sample of malware and milware, then classified the strongest found similarities within each category.

To follow the life cycle of an attack, we started with the propagation of code to the target. While GOZ alone used the Cutwail botnet to spread, both GOZ and Tinba used emails containing links to malicious web servers and compromised web hosts, inadvertently serving malware. [21, 22] The emails contained lures to bait the victim to click through to these sites; these lures were crafted email templates emulating unpaid invoices, negative account balances and social media advertisements containing logos and identifiers that belonged to legitimate companies. To verify this, we set up a honey-pot to try and attract emails from these spammers and what we received was poorly customized and did, in fact, lead to known malicious web hosts serving Blackhole Exploits. [25] The found URL patterns are as follows:

```
[redacted].php?pBzmU=ePRGAAKDWk
&CMSgsDyzkuFvs=JnhjMIPLmQY
200 OK (application/java-archive)
```

```
[redacted].php?DgdAXYmfoKifsN=sNZsfdfdRLslud
&xpcuSaClYajZ=bsczZZysmLE
200 OK (application/java-archive)
```

GOZ and Tinba both make heavy use of the Blackhole Exploit Kit as a means of facilitating code execution on the victims computer. This fact wassignificant in our analysis; unlike milware, malware is willing to make use of third-party exploit kits. This use of a commercial exploit kit helps support the analysis that malwares objective is to infect as many victims as possible. Some of Blackhole's releases make use of 15 to 24 existing CVEs with CVSS scores ranging between 6 and 9.5; however, most of the CVEs used in Blackhole are half-day exploits which are known vulnerabilities that have been patched by the manufacturer. Subsequently, this indicates that those who are using Blackhole do not necessarily need to guarantee code execution on a high percentage of their targets, but instead propagate to as many victims as possible.

Moving from propagation to payload, we found that while the purpose of each sample was different, Tinba being a banking trojan and GOZ a botnet, there were key similarities between the two. Most predominant of these was the ability to glean account information from, and dynamically inject web content into, targeted browsers. This functionality is seen on a number of malware samples in the wild and requires very little specific information about the victim prior to attack. [26, 27, 28]. In a limited extension of this pilot study,digging deeper into the payload's code base, we found that even when looking at a new strain of these samples, there were a striking number of similarities. [28]

Analyzing each of the two milware samples, we took considered the findings from our malware binaries and used the PrEP framework to draw parallels between the two sets in order to identify key differences in propagation, exploitation and payload functionality. While our work with the milware samples was limited by relatively greater reliance open source documentation than with the malware samples, we were able to utilize an arms-length relationship with an information security vendor which included some analytical support with the source of our samples. Although we did not posses binaries, we were able to obtain the .text, .code, .data and .bss sections, where applicable, as well as the imports and exports of the binaries to statically analyze. To confirm or refute our propagation hypotheses, we relied on first hand accounts, gathered emails and official documentation.

Working with Sandworm, we immediately looked to the propagation method attackers chose to distribute their code. Examining multiple email samples sent to victims, we found a generally high level of customization in the prose which specifically targeted their victims. [29] The Sandworm attackers also used attached Powerpoint slides to initiate CVE-2014-4114 on the victim's com-

Sample	Classification	Propagation	Exploitation Method	Payload	Features
Sandworm	Milware	Tailored Spear Phishing	CVE-2014-4114	BlackEnergy	The payload, fontcache.dat, is capable of executing the following commands: delete, ldplg, unlplg, update, dexec, exec, and updcfg.
Sony Attacks	Milware	Tailored Spear Phishing & Physical Access	SMB Worm Tool	Listening Implant Lightweight Backdoor Proxy Tool Destructive Hard Drive Tool Network Propagation Wiper	The payload allows attackers to download and execute further payloads, migrate through the network, pivot to other machines, exfiltrate data and wipe hard drives.
Game Over Zeus	Malware	Email Spam & Compromised Web Hosts	Blackhole Exploit Kit	GOZ Server	The generic payload allows attackers to skim banking information from the victim, use the victims internet connection to initiate a DDoS attack and pivot to other machines on the Network
Tiny Banker	Malware	Email Spam & Generic Spear Phishing	Blackhole Exploit Kit	Tinba Server	The self-contained payload allows the attacker to skim banking information, pivot to other machines and persist on the machine through a backdoor.

Table 1: Overview of Examined Samples

puter. [18] While popular security practice tells users to never download attachments from untrusted emails, the work that the Sandworm Group put into crafting legitimate looking emails provided a convincing, tailored, basis for the recipients to trust the messages and their content.

Looking at the Sony attack, we had to take a different approach to analyzing the propagation of the attack due to the fact that the initial compromise was most likely due to an insider and not through an email campaign as seen in our previous samples [30]; however, we still wanted to focus on the degree of customization present in the code. To do this we looked at how the milware acted within the scope of Sonys network. We suspect, due to the tool set used [31], that the attackers attempted to breach the higher priority targets in order to further cement their presence in the network. This process would involve targeting network administrators' computers and using the stolen elevated credentials to breach the Active Directory and Domain Controller servers to gain access to credentials used network-wide.

Finally, we considered the design and functionality of each sample's payload and suporting exploits. In both cases, the actual functionality of the payloads were tailored to a particular purpose. In the case of Sandworm, the attackers used a version of the BlackEnergy remote administration tool that supported the use of delete, ldplg, unlplg, update, dexec, exec, updcfg. These commands allowed the attacker to gain and maintain persistent access to the victims machine through its backdoor functionality as well as allowed the attacker to run any new payload sent to the system. [32] The payload deployed on victim machines in the Sony attacks contained narrowly defined functionality, designed for the purpose of exfiltrating corporate data, further propagation of the payload through the network, dynamic access to compromised nodes, and the destruction of victim hard drives. [31]

The exploits supporting each of these payloads were tailored to their function. In the case of Sandworm, CVE-2014-4114 was used via an email attachment to give the attacker time to create a backdoor in the system to enable further code execution. [19] Within Sony, the SMB Worm Tool was used to propagate itself through the network after the initial victim was compromised. [31] This allowed the attackers to traverse network undetected due to its zero day classification. [31]

5 MASS Index

The analytic tools available to delineate between malicious samples are varied but still rudimentary. Developing a means to consistently differentiate between types of code, state-authored samples for example, would be useful to the information security research and policy community. Leveraging a set of characteristics derived from the pilot analysis of milware and malware samples outlined above, this paper proposes the MAlicious Software Sophistication or MASS Indexas a basic tool to identify the authorship of malicious code samples. The MASS Indexconsists of four main categories by which to classify malicious software: propagation to the victim, propagation within a network, severity of the vulnerability used, and tailoring of the payload.

5.1 Propagation

Propagation methods can be classified according to their Scale and Specificity. Scale determines the total possible target pool i.e. how many computers and devices from

the global population are accessible. Code which propagates over the internet is likely to be found much farther afield than that spread over compromised storage media. The Scale of a compromised web site would be tremendous if the site in question is Google or tiny if a GeoCities page. Conventional botnets may have tens of millions of slave machines and present an excellent method of propagating to targets indiscriminately [33]. A Propagation method which targets all internet connected computers (large scale) is thus different from one which targets only users connected to a particular local area network (small scale). GOZ's server contains tools such as DGAs, USB, NFS, Samba spreading tools, designed to propagate as widely as possible.

Specificity measures the targeting constraints placed on malicious code, determining how much of the possible target pool is of interest or active. These could be technical limitations, focusing on a particular operating system or software version, based on personal information like account credentials or details about co-workers, or the presence of certain filenames on the victim's machine. Specificity can help to contain the spread of malware infections, lowering the likelihood of detection and limiting defensive response.

In propagation to the victim, milware tends towards medium to small scale and highly specific propagation methods while we find malware employs large scale methods with little to no specificity. Propagation scale can be relatively easily established by looking at the format code is spread in and the degree of customization of the delivery vector. This can range from the use of the Targeted Threat Index [1] with regard to email propagation, to examining how much prior access or knowledge the attacker had about the target. In the case of GOZ infections, we see a tendency for GOZ to spread to any other computer where code execution can be achieved.

In propagation within a network, after the attacker has compromised the target, there is a great deal of variation between mil- and malware samples. Milware demonstrated a trend towards attacking higher value targets before further propagating to lower value targets, cementing its position in the network. Malware by contrast, propagated without obvious human input or regard for the value of individual nodes.

5.2 Exploit Severity

The severity of the exploits used to compromise the targets serves as another major indicator in the MASS Index. Reuse of exploits or the use of exploits with a low CVSS generally indicated malware use of higher scored and chained exploits was associated with milware. GOZ uses complex 0days to penetrate the target system; however, as demonstrated in section 4.2, these 0days are compiled together in a Blackhole, a third party collection of exploits. Because GOZ relies so heavily on third party exploit kits, the CVSS score, while critical, is predictable across the board. Consequently, the wider exposure of these exploits via Blackhole will substantially increase their likelihood of discovery and mitigation, leading to their use as signatures in many IDS/IPS solutions. Conversely, in milware we see a higher CVSS scores across the board; Sandwormhad an over all rating of 9.3 with a propagation subscore of 10 and exploitability subscore of 8.6 [34].

5.3 Payload Customization

Examining the tool set and functionality of the payload in question, we found the level of capability and customization in milware payloads to be higher on average. This involves the payload in question not containing a broad set of tools generically used for post-exploitation capability on machines from a desktop PC to a server, but a set of tools tailored to each specific target, i.e. a payload for a web server, a payload for a desktop, and a payload for a Domain Controller. When looking at Sandworm, we found a very small set of tools intended to guarantee persistence and data exfiltration. With regard to the Sony attacks, Table 1 shows that while the payload did contain some generic tools, they were all tweaked to the constraints of Sony's network and limited their functionality to said network. By contrast, GOZ's payload was identical regardless of the target in question.

5.4 Limitations of Metric

As this is a pilot program, access to both samples and manpower present major limiting factors. To mitigate these problems, we chose samples which were wide spread and/or well documented. In doing so, we could increase the validity of our findings as well as expedite the process of reverse engineering. We further supplemented our limited resources with technical and incident response documentation of the milware samples and the assistance of an anonymous information security organization for part of our analysis. This is a pilot project and so the technical indicators we've identified in the MASS index, could be made more robust with access to a more substantial data set of both mal- and milware samples. We have developed a proposal for future work and would be eager discuss these next steps with parties beyond those we have already solicited.

6 Findings

Using the MASS index, we can begin to draw a conclusion as to consistent means by which to distinguish be-

Malware	Milware
Less attention to detail with regard to exploitation phase	Exploitation phase tailored towards targets
More robust payload/backdoor	More compact payload with limited functionality
Propagation more indiscriminate	Propagation has a much more limited scope
Less robust persistence modules	Persistence modules tailored target to allow for maximum epoch
Not tailored to the target beyond the exploit kit's capabilities	Each phase of milware life cycle is tailored to the target

Table 2: Core Set of Differentiating Features between Malware and Milware

tween state authored code, milware, and malware. From our research, we were able to ascertain a core set of features present in both sample sets that correlated to the average life cycle of an attack. Within these features, we also identified ways in which the malware and milware samples differed in their operation. Table 1 provides an overview of our findings with regards to the core set of features present on each sample while Table 2 identifies the divergence identified between malware and milware.

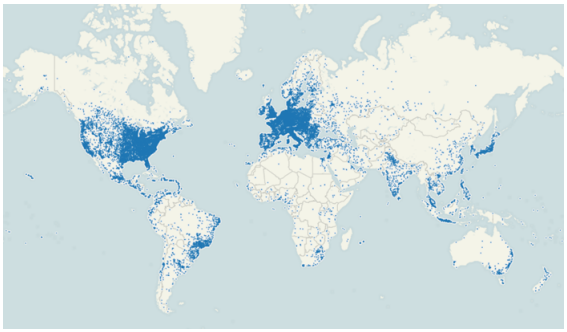

Figure 1: Geographic distribution of nodes in the ZeuS P2P network by Bot ID. (Source: Dell SecureWorks)

The first major indicator that we found across samples was the method by which code propagated to the victims. Referring to the Targeted Thread Index, we found that the emails containing malware had a Targeting Score from 1 to 3 while the emails containing milware had a Targeting Score from 4 to 5. [1, 35] We also noted that the method used to expose victims to malicious code varied. As can be seen in Figure 1, the propagation of `GOZ` was very large scale and employed something akin to the 'shotgun technique', prioritizing spread of malicious emails to as many victims as possible. Milware incidents such as `Sandworm` used similar propagation methods with similar scale but far more restrictive specificity, pushing code to a highly constrained group of targets. [29] Some work on the Sony attacks suggests they may not even have involved an email based, and thus medium to large scale, propagation method. [30] [36]

Examining how samples acted within the target network, we found that milware demonstrated a more human touch approach as opposed to malware focused on spreading to as many hosts as possible and typically moved to topographically proximate peers instead of high value nodes within the network. This aligns well with our first observation of malware being focused on wide-spread initial propagation vice milware's tendency to prefer precision. As demonstrated by the Sony attacks, careful propagation within Sonys network infrastructure was key to the attackers success. The presence of the Network Propagation Wiper in the attackers toolkit indicates and interest and investment in covert navigation of Sonys network. [31]

Both malware samples relied on the Blackhole Exploit Kit to support propagation and gain access to target systems. While these different exploit kit versions did contain exploits with a relatively high CVSS, many of these had already been patched or had mitigation techniques well known to many professionals. However, when looking at our milware, both samples used exploits with critically high CVSS ratings that were also of the zero day category. This shows us that milware will prefer to use a single exploit that guarantees code execution on the target as opposed to malware which prioritizes wider propagation and so leverages it based collections of exploits to successfully compromise a broader array of targets.

Lastly, the way samples interacted with the target systems post-exploitation was of great interest to us. The milware samples demonstrated a much higher level of customization towards their targets, in that the capabilities of the payload did not exceed what was needed of them. Our malware samples by contrast were focused on getting maximum utility out of the code, targeting widely used information types and applications. This difference allows us to parse a characterizing distinction between mal- and milware. It also reaffirms our general conjecture that malware targets the widest range of victims while sacrificing probability of success while milware has a much more narrow victim scope and optimizes for a higher success rate.

6.1 Implications

The existence of state-authored malicious software is not a new concept but there has, so far, been little scholarly investigation into the implications the pervasive development and deployment of such a separate category of code might have. Developing the MASS index, this pa-

per demonstrates that these categories can be systematically differentiated. The implications of these two categories of code are substantial:

Governments experience little direct cost from disclosure of their activities to the public. The commercial and academic information security community has become generally oriented around a sunlight is the best disinfectant model when it comes to sophisticated actors and their code. While not an industry standard behavior with all attackers for competitive reasons, information about particular techniques, tactics, and exploits has become both an information sharing device and marketing tool for information security researchers. One underlying expectation is that such disclosure will dissuade attackers and aid defenders. While the publication and circulation of information about milware is useful as an academic research tool, several years of slick APT reports, including a trove of information on Chinese [37] and Russian [38] activities and revelations by Symantec [39] and Kaspersky about several alleged US samples, including one espionage framework in place for almost 14 years [40], seems to have done little to dissuade milware development or deployment.

This lack of impact is in large part due to the fact that milware reverses the traditional hierarchy of information security, where defenders have the onus of legitimacy and hackers are operating outside of the law - their existence a product of the confluence of fundamental insecurity in most commercial software and opportunities for financial gain. States, to a very rough extent, are the law and have little fear of material harm from the pubic effects of disclosure about their activities.

Second, states have substantially greater resources to develop innovative attack methods and new exploits. While this resource disparity has long been focused on the threat of states developing destructive payloads, an acknowledged resource and expertise intensive endeavor [41], a more immediate threat is that these techniques and exploits developed for milware applications will trickle down to malware authors. [42] The chief threat of malware then is not the prospect of readily available destructive payloads, but that states might inadvertently fund a massive research and development apparatus for non-state groups, further intensifying the disparity in capabilities between attacker and defender.

It is not a novel idea to suggest that there are a small set of sophisticated threat actors in the information security space whose code and tactics may leak into the actions of others but recognizing the source of these innovations as states identifies a key problem. States financial and human resources are substantially greater than any non-state organizations meaning the output of innovation, in terms of both variety and volume, is sufficiently great so as to constitute a fundamentally different phenomenon.

Even where state resources are not used directly to develop new code, the presence of a market like mechanism for groups to buy, sell, and trade components of malicious software has been established (though not yet well studied). [43, 44] While estimates of the prices and popularity of different tools is subject to some debate, the resources of state actors will impact these markets. The rise of milware may be pricing software vendors and other defensive organizations, operating through bug bounty programs, out of the market. More insidiously, the presence of states with financial resources to burn and an appetite for the latest and greatest vulnerabilities in widely used commercial software may well encourage substantial growth in the number and talent of individuals who participate in this market as sellers. As the prospect for financial gain increases, more and more people join in to sell their malicious wares to the highest bidder. Milware then offers the prospect of becoming a driving force in the sophistication and variety of malicious software components, especially vulnerabilities, available on the web. For states, this might already be encouraging an arms race to compete for the most effective espionage tools and weapons. For non-state actors, it may make milware-like capabilities available to terrorist groups or criminal organizations.

Fourth, states are largely immune from the existing array of legal tools used to locate and prosecute malware authors and distributors as these resources presume targets that can be subject to a states jurisdiction. A cooperative, hierarchical model exists in the infrequent collaboration between national law enforcement agencies tasked with cybercrime. Limiting the use of milware is an interstate monitoring and enforcement task more akin to conventional heavy arms sales or export control restrictions. Non-state actors can be pursued and prosecuted but states and their milware will largely be subject to the states own willingness to self-restrain or the ability for other states to compel the same. This constitutes a parallel enforcement and mitigation problem for all parties malware tends to be large scale and much is indiscriminate. Milware by contrast is focused on small target sets and is distributed by actors who are substantively different in terms of motivation, resources, and dependence on other entities.

Finally, the regulatory apparatus in place in many states, especially the U.S., privileges standards for the defense of networks and information systems rather than holding liable the manufacturers of software and hardware in place on these networks. Milware takes advantage of this status quo by prioritizing the acquisition and maintenance of access to targeted systems for long periods of time over deploying particular effects at frequent intervals. The continued vulnerability of most major software families presents a less cost intensive and more

obscured operational pattern to enable less frequent but more substantial intrusions rather than engaging in small but regular attacks.

States are daring software vendors to build better software with the expectation that they can continue to beat information security vendors and existing security practices at the network and system level. Malware, distributed by individuals and non-state actors, prioritizes effects over access the particular machine compromised by an infection is less consequential than the successful execution of code to bring about the manipulation or data exfiltration desired. This is because most malware targets certain resource types within vast networks, banking credentials or PII, rather than the data tied to an individual. Milware, by contrast, is concerned with access to more narrowly tailored targets and particular pieces of information.

7 Future Work

Using a limited set of samples, this paper develops the MASS index, a metric to systematically differentiate state and non-state authored malicious software based on the pattern of propagation to and within targets, the CVSS score and 0day frequency, and the degree of payload tailoring to a particular target. This is a pilot project however and so is limited in the scope and scale of data collection and the sophistication of methodologies in use. In order to develop a more robust version of the MASS Indexand further extend and support the conclusions of this paper, we identify two possible (non-exclusive) directions for future work.

The first involved taking a large-N approach, analyzing an additional 30-40 milware and 100-150 malware samples. This would allow for more rigorous empirical scholarship, integrating a wider array of target types and styles of code authorship. An alternative involves mapping code lineage, seeking to understanding what distinguishes the evolution of state and non-state authored malware over time. By collecting as many directly related samples as possible of 3-5 variants, both state and non-state authored we can use these lineage maps to mark changes in exploit and feature selection and consider how these changes interact with the previously established MASS Indexcriteria.

In addition, we acknowledge that the term milware obfuscates other distinctions between state developed code, especially differences in the organizational structure and culture responsible for deploying code for national strategic effect, tactical battlefield use, and espionage. While these three sub-categories of milware each posses a distinct function, there is analytic utility obtained from mapping their broad similarities and differences with malicious code developed by non-state actors. Exploring the individual differences between each of these sub-categories and malware would present an interesting extension of this work.

8 Conclusion

Malicious software has long been used to describe a range of threats. From the trope of basement bound teen-aged hackers clutching Mountain Dew to the modern, much marketed, Advanced Persistent Threat, broad overuse of the term malware has impacted researchers ability to specify the range and variety of threats in the information security space. The idea of state authored code, milware, as a separate category highlights a set of challenges to the existing legal architecture and security research paradigm which are fundamentally oriented to combat individuals and organizations.

Through the MASS Index, this paper has described a rudimentary new means to differentiate between state and non-state authored code. Highlighting the implications of milware as a new category of code, we find several conventional assumptions in place for information security which should be subject to careful review and potential revision. Milware constitutes a new class of malicious software whose priorities, as a tool of state influence, and sophistication are different from code employed by non-state groups. By failing to make a distinction between milware and malicious code written by non-state actors, the information security community conflates the capabilities and intentions of criminal groups with those of states, degrading the ability to successfully adapt and respond to either.

References

[1] Seth Hardy, Masashi Crete-Nishihata, Katharine Kleemola, and Adam Senft. Targeted threat index: Characterizing and quantifying politically-motivated targeted malware. In *This paper is included in the Proceedings of the 23rd USENIX Security Symposium.*, pages 527–541, August 2014.

[2] Trey Herr. Prep: A framework for malware & cyber weapons, February 2014.

[3] P Bright. Massive sql injection attack making the rounds694k urls so far. `http://arstechnica.com/security/2011/03/massive-sql-injection-attack-making-the-rounds694k-urls-so-far/`, March 2010.

[4] Ekta Gandotra, Divya Bansal, and Sanjeev Sofat. Malware analysis and classification: A survey. `http://www.scirp.org/journal/PaperDownload.aspx?paperID=44440`, May 2014.

[5] U Bayer, A Moser, C Kruegel, and E Kirda. Dynamic analysis of malicious code. `http://dx.doi.org/10.1007/s11416-006-0012-2`, August 2006.

[6] I You and K Yim. Malware obfuscation techniques: A brief survey. `http://dx.doi.org/10.1109/BWCCA.2010.85`, November 2010.

[7] A Moser, C Kruegel, and E Kirda. Limits of static analysis for malware detection, 2007.

[8] M Schultz, E Eskin, F Zadok, and S Stolfo. Data mining methods for detection of new malicious executables, May 2001.

[9] D Ddl F Li, A Lai. Evidence of advanced persistent threat: A case study of malware for political espionage. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6112333&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6112333, October 2011.

[10] Nikolaj Goranin and Cenys Antanas. Analysis of malware propagation modeling methods, April 2008.

[11] https://www.hex-rays.com/products/ida/.

[12] http://bochs.sourceforge.net/.

[13] http://www.windbg.org/.

[14] http://debugger.immunityinc.com/.

[15] http://www.woodmann.com/collaborative/tools/index.php/SysAnalyzer.

[16] http://www.tcpdump.org/.

[17] https://www.wireshark.org/.

[18] Stephen Ward. isight discovers zero-day vulnerability cve-2014-4114 used in russian cyber-espionage campaign. http://www.isightpartners.com/2014/10/cve-2014-4114/, October 2014.

[19] William Sanchez. Timeline of sandworm attacks. http://blog.trendmicro.com/trendlabs-security-intelligence/timeline-of-sandworm-attacks/, November 2014.

[20] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. 2002.

[21] Brett Stone-Gross. The lifecycle of peer-to-peer (gameover) zeus, July.

[22] Peter Kruse. Threat report: W32.tinba (tinybanker) the turkish incident. 2012.

[23] Assaf Regev. Tinba malware reloaded and attacking banks around the world, September 2014.

[24] Fraser Howard. Exploring the blackhole exploit kit. March 2012.

[25] Kafiene. Blackhole exploit kit goes 2.1.0, shows new url patterns, June 2013.

[26] Aurelian Neagu. The top 10 most dangerous malware that can empty your bank account. https://heimdalsecurity.com/blog/top-financial-malware/, August 2014.

[27] Kaspersky Labs. Kaspersky lab statistics: attacks involving financial malware rise to 28 million in 2013. http://www.kaspersky.com/about/news/virus/2014/Kaspersky-Lab-statistics-attacks-involving-financial-malware-rise-to-28-million-in-2013, April 2014.

[28] Dell SecureWorks Counter Threat Unit(TM) Threat Intelligence. Top banking botnets of 2013. http://www.secureworks.com/cyber-threat-intelligence/threats/top-banking-botnets-of-2013/, March 2014.

[29] Critical Intelligence. Sans icsthreat briefing. http://www.critical-intelligence.com/resources/papers/CI-Sandworm-BE2.pdf, October 2014.

[30] Bruce Schneier. More data on attributing the sony attack. https://www.schneier.com/blog/archives/2014/12/more_data_on_at.html, December 2014.

[31] US-CERT. Alert (ta14-353a) targeted destructive malware. https://www.us-cert.gov/ncas/alerts/TA14-353A, December 2014.

[32] Kyle Wilhoit and Jim gogolinski. Sandworm to blacken: The scada connection. http://blog.trendmicro.com/trendlabs-security-intelligence/sandworm-to-blacken-the-scada-connection/, October 2014.

[33] Brian Krebs. Researchers clobber khelios spam botnet. http://krebsonsecurity.com/2012/03/researchers-clobber-khelios-spam-botnet/, August 2013.

[34] MITRE. Vulnerability summary for cve-2014-4114. https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-4114, October 2014.

[35] Tom Fox-Brewster. Russian malware used by 'privateer' hackers against ukrainian government. http://www.theguardian.com/technology/2014/sep/25/russian-malware-privateer-hackers-ukraine, September 2014.

[36] Anthony Freed. Norse investigation focusing on a small group, including sony ex-employees. http://blog.norsecorp.com/2014/12/29/ex-employee-five-others-fingered-in-sony-hack/, December 2014.

[37] Mandiant. Apt1: Exposing one of chinas cyber espionage units, 2013.

[38] Kaspersky. Red october.

[39] Symantec Security Response. Regin: Top-tier espionage tool enables stealthy surveillance. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/regin-analysis.pdf, November 2014.

[40] Kaspersky Labs Research Team. Equation: The death star of malware galaxy. https://securelist.com/blog/research/68750/equation-the-death-star-of-malware-galaxy/, February 2014.

[41] Ralph Langner. To kill a centrifuge. http://www.langner.com/en/wp-content/uploads/2013/11/To-kill-a-centrifuge.pdf, 2013.

[42] Udi Shamir. The case of gyges, the invisible malware government-grade now in the hands of cybercriminals. http://www.sentinel-labs.com/wp-content/uploads/2014/07/Sentinel-Labs-Intelligence-Report_0714.pdf, July 2014.

[43] Andy Greenberg. hopping for zero-days: A price list for hackers' secret software exploits. http://www.forbes.com/sites/andygreenberg/2012/03/23/shopping-for-zero-days-an-price-list-for-hackers-secret-software-exploits/, March 2013.

[44] Chris Borgen. Regulating the global market for zero-day exploits. http://opiniojuris.org/2013/07/15/regulating-the-global-market-of-zero-day-exploits/, July 2013.