NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

TIME SENSITIVITY IN CYBERWEAPON REUSABILITY

by

Carissa G. Hall

December 2017

Thesis Advisor: Neil Rowe
Second Reader: Wade Huntley

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2017	**3. REPORT TYPE AND DATES COVERED** Master's thesis	
4. TITLE AND SUBTITLE TIME SENSITIVITY IN CYBERWEAPON REUSABILITY			**5. FUNDING NUMBERS**
6. AUTHOR(S) Carissa G. Hall			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

A cyberweapon is weaponized software code that exploits flaws in software. It is only effective if the flaw still exists at the time of weapon deployment. Because of this, there is only a small window of time in which a particular cyberweapon can be used. Many argue that cyberweapons can only be effectively used once, and that after first use, the vulnerability will be patched. However, the target must first detect the attack, find the vulnerability that was exploited, reverse-engineer the cyberweapon to identify signatures, then create and implement a patch. This window of opportunity between attack detection and patch implementation allows an attacker to reuse the cyberweapon against different or even the same targets as long as the window of opportunity remains open. An attacker can increase the length of time the window remains open by obfuscating the cyberweapon's signatures to make it harder to detect the attack or by making it harder to locate and remove the weapon. This can be accomplished by incorporating survivability into the weapon's design requirement. This thesis explores the strategic implications of reusable cyberweapons by specifically looking at stealth as the critical attribute that allows a cyberweapon to go undetected and survive long enough to be effectively used more than once.

14. SUBJECT TERMS cyberweapons, cyberweapon reusability			**15. NUMBER OF PAGES** 65
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

TIME SENSITIVITY IN CYBERWEAPON REUSABILITY

Carissa G. Hall
Lieutenant, United States Navy
B.S., United States Naval Academy, 2011

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN CYBER SYSTEMS AND OPERATIONS

from the

NAVAL POSTGRADUATE SCHOOL
December 2017

Approved by: Neil Rowe, Ph.D.
Thesis Advisor

Wade Huntley, Ph.D.
Second Reader

Dan Boger, Ph.D.
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

A cyberweapon is weaponized software code that exploits flaws in software. It is only effective if the flaw still exists at the time of weapon deployment. Because of this, there is only a small window of time in which a particular cyberweapon can be used. Many argue that cyberweapons can only be effectively used once, and that after first use, the vulnerability will be patched. However, the target must first detect the attack, find the vulnerability that was exploited, reverse-engineer the cyberweapon to identify signatures, then create and implement a patch. This window of opportunity between attack detection and patch implementation allows an attacker to reuse the cyberweapon against different or even the same targets as long as the window of opportunity remains open. An attacker can increase the length of time the window remains open by obfuscating the cyberweapon's signatures to make it harder to detect the attack or by making it harder to locate and remove the weapon. This can be accomplished by incorporating survivability into the weapon's design requirement. This thesis explores the strategic implications of reusable cyberweapons by specifically looking at stealth as the critical attribute that allows a cyberweapon to go undetected and survive long enough to be effectively used more than once.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

CVE	common vulnerabilities and exposures
CPU	central processing unit
DOD	Department of Defense
DLL	dynamic link libraries
IT	information technology
OLE	object linking and embedding
OS	operating systems
RAM	random-access memory
ROM	read-only memory
SMB	server message block
US-CERT	United States Computer Emergency Readiness Team
USB	universal serial bus
WINE	Worldwide Intelligence Network Environment

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would first like to thank my advisor, Dr. Neil Rowe, for his contributions and guidance throughout the writing process. I would also like to thank my second reader, Dr. Wade Huntley, for his helpful feedback and assistance in scoping my topic. Lastly, I would like to thank my husband and my parents for all of their contributions throughout this process. This would not have been possible without all of you. Thank you.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. OVERVIEW

Exploitable flaws in software code that are unknown to the software manufacturer are referred to as zero-day vulnerabilities. Many individuals and organizations are interested in exploiting these vulnerabilities by selling information about them or by creating a weaponized zero-day exploit for that software vulnerability (Stockton & Golabek-Goldman, 2013). A cyberweapon is software code that exploits a specific vulnerability on a specific system with the intent to cause destructive physical or digital effects (Herr & Rosenzweig, 2014). Cyberweapons are usually based on flaws in software and will usually only be effective if the flaw still exists at the time of weapon deployment. Because of this, there is an inherent time sensitivity surrounding the use of cyberweapons where there is only a small window of time when they can be effectively used or reused.

One factor driving the time-sensitive decision of when to use a cyberweapon is the potential for obsolescence. Many researchers argue that the longer the attacker holds onto the cyberweapon, the vulnerability becomes more likely to be discovered and patched, thus rendering the cyberweapon obsolete (Huntley, 2016). Undeniably, there are several ways in which a vulnerability could be discovered prior to its use. For example, an insider could expose the vulnerability, it could be discovered during a penetration test, or it may be obtained from a third party. Vulnerabilities can also be inadvertently removed through regularly-scheduled software updates or system upgrades. Any of these allow the target to patch the vulnerability and remove any opportunity for the attacker to use or reuse the cyberweapon.

It has been argued that cyberweapons cannot be reused because they are perishable. The perishability of a cyberweapon refers to its ineffectiveness after first use because it exposes the vulnerability that was exploited. Once the vulnerability is exposed, it can be patched and the cyberweapon will be ineffective (Huntley, 2016). However, this requires that the target recognize that they have been attacked. This could take a varying

amount of time depending on the overtness of the cyberweapon's effects and the caliber of the target's cyber defenses. Once the target realizes they have been attacked, they (or the software vendor or some agency working on their behalf such as an incident response team) must now locate the vulnerability that was exploited. This process includes reverse-engineering the cyberweapon to identify signatures and the cyberweapon's targets within software. However, the weapon may have been designed to erase its tracks, leaving minimal artifacts behind to classify signatures or reverse engineer. Once the analyst determines the exploited vulnerability, someone (often the software vendor) must write a patch to fix the vulnerability. This may be time-consuming depending on the complexity of the system. Patching the vulnerability may also adversely affect other systems. In this instance, it may take days or weeks to write a fully functioning patch. Writing a patch could also introduce new vulnerabilities into the system. To counter this, many vendors heavily test their patches to ensure there are no new potential zero-day vulnerabilities. Once the patch is written, it must be distributed and implemented to complete the process. If the target has end-users who will also be impacted by the vulnerability, the target must also disseminate the patch to those users who must also implement the patch on their systems. Thus, there is a significant gap in time between when a cyberweapon is launched by an attacker and when a system is fully patched, to include all end-users.

B. SIGNIFICANCE OF A REUSABLE CYBERWEAPON

In this gap in time, there is a window of opportunity for the attacker to reuse the cyberweapon. To appreciate the significance of this gap, first it is important to distinguish between low-level malware and highly tailored cyberweapons by comparing the intent of the payload. Highly tailored cyberweapons are designed to cause major effects in a single use, and are not designed to be reused. Conversely, low-level malware used in ransomware or denial of service attacks is meant to be reused, relying on the fact that not all users will patch their systems quickly or use antivirus software. However, this is not to say that a highly tailored cyberweapon cannot be effectively used more than once. If a cyberweapon is intended to be reused against the same or different targets, for example,

the cyberweapon could target the same vulnerability across multiple platforms that are geographically dispersed. The attacker could launch these attacks simultaneously or at different times, relying on the fact that information from attacks is often kept secret from all but a few responding agencies who may not communicate well. The cyberweapon could also be designed using obfuscation or encryption techniques that change the signature of the malware with the intention of getting it past antivirus software by making it look different.

A cyberweapon that is designed to be used effectively more than once provides a significant strategic advantage to the attacker and it is extremely hard to defend against. Once a target is attacked, the victim must begin the post-attack process and their systems will not be secure until it has been fully patched. This gap in time allows for reuse of cyberweapons. This paper will show that cyberweapons can be effectively used more than once, and that there is a benefit to the time-sensitive component inherent in cyberweapons.

The remaining chapters are organized as follows. Chapter II is a literature review of related research. It introduces the concept of cyberweapon reusability resulting from the window of opportunity that is created between when a vulnerability is exploited and when it is fully patched and implemented. Chapter III expands on the previous chapter and introduces techniques an attacker can use to influence the window of opportunity directly, such as manipulating malware signatures to extend the window. Chapter IV discusses the importance of stealth when creating a cyberweapon and how that directly influences its reusability. Lastly, Chapter V provides major conclusions and opportunities for future research.

THIS PAGE INTENTIONALLY LEFT BLANK.

II. RELATED RESEARCH

A. OVERVIEW

Cyberspace exists in both physical and logical ways. The Joint Publication on Cyberspace Operations (U.S. Joint Chiefs of Staff, 2013) states that cyberspace can be defined in terms of a physical, logical, or cyber-persona layer. The physical layer is the geographic location of the physical hardware and infrastructure. The logical layer involves cyber components that are closely related but may not reside in one geographic location, and their interconnections. The cyber-persona layer refers to the actors in cyberspace. Together, the layers are used to define cyberspace as an "interdependent network of IT infrastructures, including the Internet, telecommunications networks, computer systems, and embedded processors and controllers, and the content that flows across and through these components" (U.S. Joint Chiefs of Staff, 2013, p. I-2).

B. DEFINING CYBERWEAPONS

To describe something in the cyber domain, parallels are often made between the conventional and digital world. Rid and McBurney (2012) define conventional weapons as the tools used to threaten or cause physical harm to a target. The physical effects of conventional weapons range on a spectrum based on their resulting level of damage. The same also applies to cyberweapons. Generic low-level malware is malicious software that can influence a system, but does not cause direct harm. Most malware used by criminal attackers is low-level malware that causes minimum damage. On the opposite end of the spectrum, there are cyberweapons—highly capable pieces of malware which can penetrate a system and possibly autonomously cause physical damage. Cyberweapons are "computer code that is used, or designed to be used, with the aim of threatening or causing physical, functional, or mental harm to structures, systems, or living beings" (Rid & McBurney, 2012, p. 7). They are used to conduct cyberattacks designed to deny, degrade, disrupt, destroy, and manipulate adversary's networks. The effects can be both physical and logical depending on the desired goal (U.S. Joint Chiefs of Staff, 2013).

The method of delivering a weapon to the target also differs in the cyber domain. Cyberweapons usually take advantage of flaws in software code. In fact, there are an estimated 20 flaws per every thousand lines of source code (Smeets, 2017). Some of these flaws are vulnerabilities that can ultimately be exploited if discovered. Vulnerabilities unknown to the software vendor, zero-day vulnerabilities, provide a special advantage to the attacker. Ablon and Bogart (2017) define a zero-day vulnerability as a vulnerability without a publicly released patch. A vulnerability becomes an exploit when there is code that can take advantage of the vulnerability and grant the attacker access to the system, something not always true for a vulnerability. Zero-day vulnerabilities give attackers the advantage because they often grant the attacker a more unobstructed access to a system.

However, there are other ways for an attacker to gain access to a system that do not require zero-day vulnerabilities. Users who fail to conduct timely software security updates and antivirus updates will have unpatched operating systems. Security updates remove previous vulnerabilities in software, so an unpatched system gives attackers easy access into a system. Social engineering can also enable attackers to access a system. For instance, user information can be used to conduct email phishing in which an attacker sends a link with malware attached to it. The malware could contain a virus, a Trojan Horse, a worm, a rootkit, or could send the user to a malicious site. The user is especially vulnerable if there is no antivirus software installed or if it is not up to date. A malicious attacker could also conduct a brute-force password attack to gain system access using personal information they gathered. There are also several kinds of insider threats that could enable attacks from within. An insider threat could install some form of removable media into the system which could also initiate a backdoor on hardware or software. The attacker ultimately just needs to gain access onto the network somewhere, such as through an unprotected printer, and then they can begin privilege escalation and persistence.

1. Classifying Types of Cyberweapons

Herr (2014) introduces a model for describing cyberweapons which they refer to as the PrEP framework. The model identifies three components: a propagation method, an exploit, and a payload. The propagation method is defined as the way in which the weapon is delivered to the target. This can be done through removable media, compromised certificate authorities, compromised email attachments, or compromised websites, and could have multiple stages. The exploit is the code that takes advantage of a specific vulnerability in the target that allows access to the system. Vulnerabilities can be in the operating system, the browser, firmware, or even in hardware. Herr breaks exploits into three categories: access, privilege escalation, and code execution. The payload is the weapon itself, the code that accomplishes a goal such as espionage, persistence, or denial of service. This framework can apply to many kinds of cyberweapons such as malware, worms, and viruses. It also allows distinguishing between different variations of the same cyberweapon.

Herr and Rosenzweig (2014) acknowledge that all cyberweapons are malware, but not all malware are cyberweapons. A cyberweapon must contain all three components of the PrEP framework, but generally a destructive payload is what differentiates a cyberweapon from malware. The authors use Metasploit as a case study. Metasploit is an open-source software framework that consists of a set of "hacking tools including exploit configuration modules, payload selection and application features, obfuscation modules to evade intrusion detection systems, and graphical collaboration tools" (Herr & Rosenzweig, 2014, p. 316). The authors explain that the exploits on Metasploit cannot be considered malware unless they are used in conjunction with a propagation method. Further, to be considered a cyberweapon, the exploit and propagation method must be used in conjunction with a payload that causes destructive effects.

Another way to classify cyberweapons is introduced by Herr and Armbrust (2015) which uses the characteristics of the code. They analyzed samples of open-source malware that have been previously attributed to different actors. They defined a "Malicious Software Sophistication or MASS index," which "relies on a set of

characteristics which describe the behavior and construction of the malware including the severity of exploits and customization of the payload" (Herr & Armbrust, 2015, p. 1). These characteristics could be used to identify whether the code came from a state-sponsored actor and make current attribution techniques stronger. This work also references the PrEP framework, to suggest that differences in characteristics will occur in each of the three components of the PrEP framework.

C. VULNERABILITY LIFE CYCLE

Cyberweapons exist because of the vulnerabilities in software code. To explain the life cycle a vulnerability undergoes, Arbaugh, Fithen, and McHugh (2000) introduced a life cycle model that consists of vulnerability birth, discovery, disclosure, correction, publicity, scripting, and death. Birth refers to the date the flaw was written into the code. Discovery refers to the date that an actor finds the vulnerability. Disclosure is the date that the actor reveals information about the vulnerability. Correction is the date the vulnerability was patched by the software vendor. Publicity refers to the date the public learns of the vulnerability. Scripting refers to the date an attacker exploits the vulnerability. Death refers to the day that the vulnerability is no longer exploitable. The authors note that a vulnerability may not progress though every one of these states. The critical states of the life cycle model are the discovery, disclosure and correction of vulnerabilities.

1. Discovering Vulnerabilities

Wilson, Schulman, Bankston, and Herr (2016) introduce the concept of a vulnerability ecosystem that consists of several different kinds of actors who are interested in finding software vulnerabilities for a variety of different reasons. The actors are characterized into four groups: independent agents, small teams, larger teams, and governments. The independent agents could be security researchers or amateur hackers. The small teams are academic labs or security firms. The larger teams are the large technology companies. Governments refer to State-sponsored agencies. The actors who search for vulnerabilities may be white-hat hackers who are seeking to find and patch

vulnerabilities, or they may be black-hat hackers who are finding vulnerabilities for destructive purposes. Some actors may be driven by financial incentives where they can report the vulnerability to a bug bounty program or sell it to a third party, regardless of whether or not the vulnerability will later be exploited. In fact, there is an entire market for buying, selling, and renting vulnerabilities. Government actors have more capital than the other actors and will use it to purchase zero-days and other advanced tools. However, they are also highly capable of discovering and developing their own vulnerabilities. Criminal actors are focused on purchasing tools with the largest net return, which means the largest scale with minimal investment. This includes tools such as renting a botnet or other less-sophisticated tools. Another aspect of vulnerability discovery is how quickly the software vendor becomes aware of the vulnerability and creates a patch. Because of this, some software vendors will purchase vulnerabilities for their own products.

2. Reporting Vulnerabilities

Once an actor finds a vulnerability, Wilson et al. (2016) explain that there are three different reporting options. First, they could choose to keep the vulnerability secret, referred to as non-disclosure. This is typically done by government actors or market brokers who rely on the principle that if few know about the vulnerability, few can patch it, which makes it more valuable to buyers or governments. An actor could also choose to responsibly disclose it to the vendor so it can be patched before information is released to the public. Or the actor could choose to fully disclose the vulnerability to the public without first informing the vendor. The last two options are designed to pressure the vendor into creating a patch for the vulnerability. Whether vulnerability information should be made public prior to the release of a patch is the topic of a contentious debate which will be further discussed in Section D. Interestingly, Shazad, Shafiq, and Liu (2012) conducted a large-scale study on the trends in software vulnerability disclosure between 1990 to 2010. They determined that between 1997 and 2006, monthly public disclosures of vulnerabilities were exponentially increasing. However, since 2008, the number of monthly disclosures has been decreasing.

A list of every publicly released vulnerability is maintained by the Common Vulnerabilities and Exposures (CVE) database which is sponsored by the United States Computer Emergency Readiness Team (US-CERT). Arora, Krishnan, Nandkumar, Telang, and Yang (2004) explained the sequence of events following a vulnerability disclosure to CERT. First, CERT will contact the vendor and provide them a prescribed amount of time to patch the vulnerability. After the time has elapsed, CERT will send out a public advisory, and a CVE identifier will be assigned to the vulnerability.

3. Patching Vulnerabilities

The duration between vendor discovery and patch distribution differs per vulnerability depending on when the attack is discovered by the vendor so they can begin the patch development process. Bilge and Dumitras (2012) were able to conclude that attacks can last anywhere between 19 days and 30 months with an average zero-day attack lasting up to 312 days before it is discovered. Some attacks have taken up to 2.5 years before they were discovered.

Shazad et al. (2012) conducted a pattern analysis of 46310 vulnerabilities discovered between 1988 and 2011 and were able to conclude that up until 2005, "the percentage of vulnerabilities patched on or before disclosure dates consistently decreased." This is attributed to the overall lack of security concerns during this period in time. However, this has been gradually improving since 2005 and now "vendors have been providing patches for more than 80% of total vulnerabilities till their disclosure dates." This is attributed to the monetary incentives advertised by the software vendors to motivate vulnerability discoverers to report their findings directly to the vendors. Future research will prove interesting to see whether this trend continued from 2012 to current day (Shazad et al., 2012, p. 777).

However, once a patch is released by the software vendor, the user still needs to install it. A recent study determined that, "most hacking attacks in 2015 exploited known vulnerabilities that the targets had failed to address despite fixes having been available for months or even years" (Wilson et al., 2016, p. 4). Thus, a vulnerability's life cycle

ultimately depends both on how quickly the vendor discovers it and how quickly the user implements the patch.

D. THE DEBATE ON PUBLIC DISCLOSURE

There is a debate over whether there should be instantaneous or delayed ("responsible") disclosure of software vulnerabilities. Instantaneous disclosure is publicly releasing the vulnerability information at the time of vulnerability discovery prior to vendor patch development; responsible disclosure is waiting to release vulnerability information until the patch is ready. Those who argue for instantaneous disclosure maintain that this puts pressure on software vendors to release patches. Responsible disclosure supporters argue that public disclosure of unpatched vulnerabilities directly leads to more attacks. Both sides agree that the vulnerabilities need to be patched quickly or they risk being rediscovered by malicious actors.

Ozment (2005) examined vulnerability disclosure and vulnerability rediscovery. The study sought to determine whether vulnerability discovery depleted the number of vulnerabilities in a system and reduced the probability of vulnerability rediscovery, by using software-reliability growth models. For the study, the author used data from vulnerabilities found in the OpenBSD operating system. 39 vulnerabilities had been discovered over two years. To determine whether vulnerabilities were being depleted, the vulnerabilities were analyzed using two different software-reliability growth models, "Brooke's & Motley's Discrete SR Binomial Model" and "Yamada's S-Shaped Reliability Growth Model" (Ozment, 2005, p. 10). Both models had an acceptable goodness-of-fit. The results suggested that vulnerability disclosure is socially beneficial as it provides a hardened product. The article also contradicts prior research which stated that vulnerabilities could not be rediscovered by providing numerous instances of vulnerability rediscovery using Microsoft Security Bulletins and an analysis of potential reasons for rediscovery. This article assumed that everyone who is discovering or rediscovering vulnerabilities were "white-hat" (non-malicious) researchers, so from an attacker's perspective several things are not addressed. For instance, this article fails to

address the gap in time between patch readiness and a fully patched system and the security concerns with rediscovery.

Bilge and Dumitras (2012) conducted an empirical study on zero-day attacks. The data used were taken from the Worldwide Intelligence Network Environment (WINE), a database that was compiled by Symantec from 11 million hosts across the world. The hosts targeted by the attacks were all real computers (not honeypots). First, the authors identified executable files that were targeted in specific exploits. The authors then retrieved the executable-file public disclosure date. They could use these two data points to look for instances of an executable file existing on a system prior to the public-disclosure date. This would identify a zero-day attack. The results showed 18 vulnerabilities were exploited prior to their public disclosure. From this, they could determine that after the vulnerabilities were publicly disclosed, attacks increased in volume up to 5 orders of magnitude. They also found that 42% of the vulnerabilities were used within 30 days after the public disclosure date. These findings support the majority thinking that public disclosure of vulnerabilities cause attacks to increase. However, participants of the debate "disagree about whether trading off a high volume of attacks for faster patching provides an overall benefit to society" (Bilge & Dumitras, 2012, p. 842).

Arora et al. (2004) tested how quickly software vendors release a patch following public disclosure and the probability that attackers find and exploit unpublicized vulnerabilities. Their goal was to provide evidence for a good public policy that would end the contentious debate over vulnerability disclosure. The data for the test was taken from 14 honeypots utilizing four different operating systems: Linux, Solaris, OpenBSD, and Windows. To test the frequency of attacks on the honeypots, the researchers selected 308 vulnerabilities from the CVE database. They classified the vulnerabilities based on whether they were secret, published, or patched. Over a period of nine weeks, there were 2772 observations taken from the honeypots. The results showed that public disclosure forces the software vendors to patch earlier than they otherwise would have, but public disclosure also increases the frequency of attacks on users. However, vulnerabilities that

are kept secret tend to remain secret and are never publicly disclosed, resulting in fewer attacks on users.

E. PERISHABILITY OF CYBERWEAPONS

Smeets (2017) discusses the transitory nature of cyberweapons. Most scholars refer to cyberweapons as single-use or "perishable" weapons. However, the concepts are different. Single-use means that once the cyberweapon is used, the vulnerability will be patched and can no longer be exploited. Perishable means that cyberweapons become ineffective after first use because use exposes the vulnerability that was exploited. Smeets suggests there is a temporary period in which cyberweapons can cause harm, which is what makes them transitory. They can be stockpiled, and there is an average of 312 days between a vulnerability being exploited and its patch implementation, which means that there is still an opportunity to reuse the cyberweapon after first use.

Schneier (2000) referred to the transitory nature of cyberweapons as a "window of exposure." His framework consists of five phases of the vulnerability life cycle. Smeets (2017) took this framework and added in additional events to make it more comprehensive. Figure 1 summarizes the new framework which includes $t_{vulnerability}$, $t_{discovery}$, $t_{exploit}$, $t_{awareness}$, $t_{disclosure}$, and t_{patch} to account for the date the vulnerability was introduced into the code, the date the vulnerability was discovered, the date the vulnerability was exploited, the date the vendor became aware of the vulnerability, the date the vulnerability was disclosed to the public, and the date the patch is released to the public respectively. These events will not necessarily happen in this order. This framework is designed to help explain the transitory nature of cyberweapons because there is only a short period of time during which the weapon can be effective. Ultimately, a cyberweapon can be effective so long as it remains within the window of exposure, which exists between $t_{vulnerability}$ and t_{patch}. Once the vulnerability has been patched, the cyberweapon can no longer be effective.

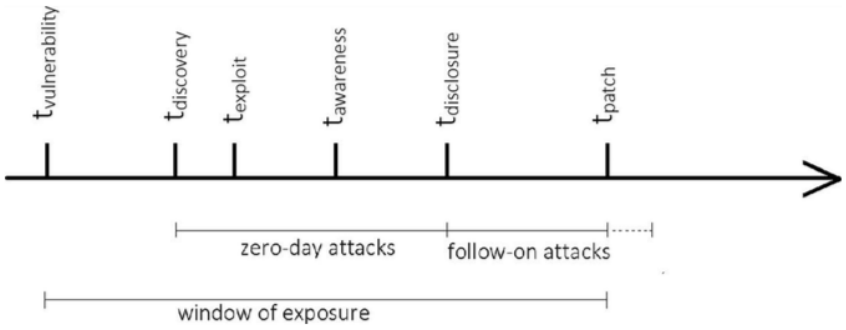

Figure 1. Life cycle of a cyberweapon's effectiveness. Source: Smeets (2017).

Smeets (2017) argues that once a cyberweapon is used, its effectiveness decreases against other targets. This is due in part because of the payload design. A payload that takes down an entire system will likely cause the duration of time between $t_{exploit}$ and $t_{awareness}$ to decrease. But if the payload is an espionage tool, it may not be found as quickly, causing the time interval to increase. Effectiveness also changes with the target pool size. If the goal of the attacker is to infect as many vulnerable users as possible, then the attacker could use the cyberweapon and potentially affect thousands of users. However, more targets mean a greater chance of discovering the attack quickly. It is true that with many targets, there will be some users who never patch their system after it is released. This allows the same vulnerability to be re-exploited by the same or different cyberweapon. However, for the users who install the patch, reusability of the cyberweapon has ended.

1. Risk of Rediscovery

Ablon and Bogart (2017) proposed a set of factors for how long governments should keep zero-day exploits in their arsenals before they risk rediscovery. The factors are life status, longevity, collision rate, and cost. Life status refers to the vulnerability and whether or not it is public or private knowledge, meaning whether the vulnerability is considered a zero-day vulnerability or is fully patched. A private vulnerability has a higher persistence value than a public vulnerability. Longevity refers to how long it is

likely to remain publicly undiscovered or undisclosed. A long longevity means that the vulnerability would have a high persistence value. Collision rate is the probability that someone else will discover the zero-day. A high collision rate would decrease the persistence value of the weapon. The cost refers to the development of the cyberweapon from the zero-day vulnerability which would not have a significant impact on the cyberweapon's persistence. These four criteria were used to analyze 200 zero-day exploits from 2002–2016 to determine which zero-day exploits the actor should stockpile, assuming they had to choose which to keep and which to disclose so they can be patched by the vendor. One of the major findings was that "exploits have an average life expectancy of 6.9 years after initial discovery; but roughly 25 percent of exploits will not survive for more than a year and a half, and another 25 percent will not survive more than 9.5 years" before they are rediscovered (Abalon & Bogart, 2017, p. 52). However, this study does not address the time between vulnerability discovery, patch development, patch implementation, and a fully patched system, during which exploits can still be successful after rediscovery. Also, an additional factor could be the risk versus reward. For example, a factor that addresses the cyberweapon's capability and the risk associated with holding on to the cyberweapon for future use versus the probability of vulnerability discovery which would render the cyberweapon useless with the creation of a patch.

Herr and Schneier (2017) refer to "collision rate" as a "rediscovery rate"; an example case is the Heartbleed vulnerability which was discovered by two different companies within just days of each other. When an undisclosed vulnerability is discovered, the discoverer often will not know if they are the only ones who know about it. The software vendor could already know about the vulnerability and a patch could be imminent, or there could be another individual who knows about the vulnerability and plans to make it public. Also as time elapses, there is more chance of a vulnerability rediscovery. Vulnerability rediscovery is an important factor in deciding whether a cyberweapon is useful against a particular target at a particular moment, since an easily discovered vulnerability is probably not useful to attack.

Moussouris and Siegel (2015) proposed a system-dynamics model of the discovery, rediscovery, and stockpiling processes for software vulnerabilities. The model addresses the offensive and defensive capabilities associated with vulnerabilities and the impact this has on the model. For instance, defenders want to find the undiscovered vulnerabilities and patch them quickly. However, they must also assume that an attacker also knows of this vulnerability and is stockpiling attacks on it. Therefore, the defender's ability to patch the vulnerability renders the stockpiled attacks useless. Moussouris and Siegel refer to this as discovery correlation. The authors also suggest there is a smaller discovery correlation rate with hardened software.

F. TIMING THE USE OF CYBERWEAPONS

All this analysis can be used to plan the use of cyberweapons. Axelrod and Iliev (2014) introduced a mathematical model for the best time to use a cyberweapon. In their model, the authors use variables called stakes, stealth, persistence, and use threshold. The stakes variable is the risk of the current situation, or the chance that you could lose the weapon's capability due to discovery of a fix or patch. Increasing the stakes will decrease the persistence of the weapon value because greater stakes mean it would have a greater impact if discovered. Stealth is the probability of using but not losing the reuse capability of the weapon. Increasing the stealth will increase the persistence of the weapon value because it will increase the amount of time before the weapon has been discovered. Persistence is the degree that delaying use of the weapon will not affect its usability later. Stealth and persistence are heavily affected by the network security and capabilities of the target. The use threshold is the tipping point that causes the cyberweapon to be used when the stakes are high enough.

The model computes a function of the variables noted earlier for the cyberweapon that rates the benefit of using the weapon against a particular target at a particular time. The model outputs the value by introducing "a trade-off between waiting until the stakes of the present situation are high enough to warrant the use of the resource, but not waiting so long that the vulnerability the resource exploits might be discovered and patched" (Axelrod & Iliev, 2012, p. 1298). The model provides a systematic way to weigh the

considerations surrounding whether is it more beneficial to use the cyberweapon or wait. It reduces the complexity of the decision-making process by automating some its steps, which provides an objective way of determining when cyberweapons should be used.

Hichkad and Bowie (2012) analyze a concept of secret weapons using some similar ideas. The authors developed a system to explain the importance of secrecy in maximizing cyberweapon effectiveness. To evaluate the effectiveness of cyberweapons during military operations, the authors used the following factors: operational effectiveness, degree of impact, longevity of the weapon once employed, and fragility of the weapon's use. Operational effectiveness measured the success of the weapon to meet objectives. Degree of impact measured whether the weapon had tactical or strategic implications. Longevity measured the long-term persistence of the effects. Fragility measured the complexity of the weapon, and how easily it could be reverse-engineered and patched by the enemy, rendering it obsolete. The authors concluded that cyberweapons need to remain secret to maintain efficacy, which also generates an inherent fear of using the weapon. Secret weapons are difficult to test because you do not want an enemy to get clues to them and patch the vulnerability before the secret weapon can exploit it. This brings a hesitation to test the cyberweapon in realistic settings and thus, a greater uncertainty as to whether or not the cyberweapon will actually work once it is deployed. However, not mentioned is the capability for a wealthy nation state to create a rather faithful replica of the realistic setting, whereas an entity with fewer resources would not have this luxury.

THIS PAGE INTENTIONALLY LEFT BLANK.

III. REUSABILITY

A. OVERVIEW

During a vulnerability's lifetime, there is a window of opportunity which begins once an attack exploits a specific vulnerability, and ends once a patch has been implemented that removes the vulnerability. The window will vary per vulnerability. For example, a vulnerability that is found quickly and then publicly disclosed will put pressure on the vendor to create and release a software patch, so this kind of vulnerability will only have a short window of opportunity. However, a zero-day vulnerability not disclosed immediately has a larger window of opportunity because it requires the target to recognize the attack, locate the vulnerability that was exploited, reverse-engineer the attack, and then create, test and implement a software patch.

During this window of opportunity, malware and cyberweapons can be reused effectively until the vulnerability has been patched and the window of opportunity has closed. However, these scenarios do not account for other factors that may influence a vulnerability's window of opportunity, such as an attacker's ability to influence the window directly. This chapter discusses malware signatures and how an attacker can manipulate them to extend the window of opportunity.

B. MALWARE SIGNATURES

A malware signature is a distinctive byte sequence that has been "extracted from the body of a specific malware strain" (Ask, 2006, p. 21). It is a distinctive pattern of bytes that should rarely appear in other files. It is also typically the part of the malware that contains critical steps for malware functionality and cannot be easily obfuscated.

New malware signatures are developed every day. Ikarus Software GmbH is one company that writes malware signatures from the samples they receive from "other anti-virus companies, customers, honeypots and many other sources" (Ask, 2006, p. 26). They have developed their own automated process to sort through and classify the malware they receive as either containing a pre-existing signature or as a new variant of pre-

existing malware that does not contain a pre-existing signature. Once the sorter tool makes the initial classification, a human analyst generates a signature for the new variants of malware (Ask, 2006). This is done by looking for syntactic signatures, which are particular patterns in sequences of bytes in files or main memory (Bonfante, Kaczmarek, & Marion, 2008). Fixed patterns can be generalized to "regular expressions" which allow a single expression to represent a set of similar signatures (Yu, Chen, Diao, Lakshman, & Katz, 2006). Once the human analyst writes the new signature, it must be tested and then uploaded into a database so that the users of the antivirus product will be protected from this new malware variant.

An alternative to signature-based detection is anomaly-based detection where intrusions are defined by abnormalities in system behavior. Here, malware analysts must define what the normal system behaviors are first, and then anomalous behaviors can be defined (Yang, Usynin, & Hines, 2006). Anomaly based detection can recognize some zero-day attacks, unlike signature-based detection, but it can have a high false-alarm rate.

C. NEGLIGENCE

Most system exploitations today are a result of unpatched operating systems or out-of-date antivirus software (Arbaugh et al., 2000). Users and administrators who fail to keep their operating systems and antivirus software up-to-date may be defenseless against attacks and re-attacks which are addressed by previous security updates. However, there are also cases where the vulnerabilities are public knowledge yet the vendor fails to release a patch, putting users at risk. Both scenarios allow the attacker to reuse malware because the window of opportunity remains constant. Attackers can also encourage negligence directly by coercing users into downloading malware directly to their computers.

1. Unpatched Systems

As an example of negligence in patching, the Heartbleed vulnerability, CVE-2014-0160, was in the OpenSSL software library. The US-CERT (2014) released a technical alert, Ta14-098A, addressing this vulnerability on April 8, 2014, and

announcing that a patch had been released. Then there were 600,000 websites at risk. But two months later, only half of the websites had implemented the patch (Vatu, 2017). As of January 22, 2017, a Shodan search reported that 199,594 devices were still vulnerable to Heartbleed (Kovacs, 2017). Microsoft Security Bulletins MS14-064 and MS17-010 identify two other vulnerabilities that were exploited immediately following their public disclosure. Both exploits were largely effective because many operating systems were not updated as security updates became available. Another example is Conficker, CVE-2008-4250, a remote code-execution vulnerability that was first disclosed in Microsoft Security Bulletin MS08-067 on October 23, 2008. Ten years later, Conficker has exploited almost eleven million Windows XP machines and continues to infect devices (Olenick, 2016).

Another example is CVE-2014-6332, a vulnerability that was reported in Microsoft's Object Linking and Embedding (OLE) technology in Windows (Microsoft, 2015). The vulnerability was privately reported to Microsoft and was followed by the release of a patch on November 11, 2014 in Microsoft Security Bulletin MS14-064. The following day, an exploit taking advantage of CVE-2014-6332 was publicly released. By November 14, 2014, there was evidence of the exploit being used. If users failed to install the security patch within the first three days after it was released, then they were defenseless against an attack targeting OLE. This shows how quickly attackers can create an exploit that targets a specific vulnerability.

A third example occurred on March 14, 2017, when Microsoft Security Bulletin MS17-010 explained how to patch Server Message Block (SMB) remote code-execution vulnerabilities CVE-2017-0143, CVE-2017-0144, CVE-2017-0145, CVE-2017-0146, and CVE-2017-0148 (Microsoft, 2017). One month later, an exploit known as EternalBlue was publicly released which takes advantage of the SMB vulnerability that was patched by MS17-010. On May 12, 2017, the large WannaCry ransomware attack began taking advantage of the EternalBlue vulnerability (Rudis, 2017). On July 12, 2017, two months after the patch was released by Microsoft, security researcher Elad Eraz conducted a scan across 80 million IP addresses looking for the vulnerability exploited by EternalBlue and he identified "50,000 Internet-connected systems that have the SMB_v1 flaw" (Schwartz, 2017). Despite the widespread media coverage of the WannaCry attack,

the vulnerability is still present in some systems and the same exploits can be reused by an attacker.

Many of these exploits succeeded due to failure of Microsoft users to set their systems to do automatic updates. The default setting on Windows machines is to have automatic updates selected, which means that a significant number of Microsoft users are intentionally turning this function off. Kingsley-Hughes (2017) suggests that many users choose to turn this function off because of the disruption and unreliability of the patches themselves, even though it leaves them completely vulnerable. For instance, some organizations do not allow automatic updates on workstations, and do not install them until their information-technology departments have tested them themselves. Depending on the vendor, patches can require users to download and install an entire new operating system or reboot their system multiple times, which is time-consuming. A justification is that faulty patches have been occasionally released by vendors which break the normal functionality of the computer or device and disallow users from reverting to previous updates.

As another example, Petya was introduced a little over one month after WannaCry, on June 27, 2017. Petya also takes advantage of the EternalBlue vulnerability that was patched by MS17-010. It was designed to target large companies across several different countries (Symantec Security Response, 2017). EternalBlue affected most versions of Windows operating systems. This is significant because as of April 2014, Microsoft does not support Windows XP operating system. This means there are generally no new patches being developed or promulgated to the seven percent of computer users worldwide who still have Windows XP operating systems. But this case forced Microsoft to create and release a patch for the SMB vulnerability on the Windows XP operating system on May 12, almost two months after the initial patch was released for newer versions of Windows operating systems (Dunn, 2017). Although Windows XP has a patch to protect it from the SMB vulnerability, this does not protect the Windows XP user from other vulnerabilities discovered in the three-year period after Microsoft stopped supporting Windows XP.

These case studies are focused on individual users who failed to install patches on time and do not address nation-state militaries or government organizations. The latter should be more diligent about applying patches quickly. However, diligent patching does not necessarily mean that the window of opportunity cannot be exploited by an attacker. It takes one administrator at one organization to fail to install a patch properly or on time to compromise an entire system.

2. Outdated Antivirus Software

A study by Microsoft estimates that, on average, 24 percent of the world's population is unprotected from malware because of outdated antivirus software, which leaves them five times more likely to become infected with malware (Meisner, 2013). Antivirus software is designed to check for previously identified malicious patterns (US-CERT Publications, 2009). For example, the email worm known as the ILOVEYOU bug infected almost 45 million Windows machines between May 4–5, 2000 (Ward, 2010). Symantec reported that as of May 31, 2001, all 82 variants of the worm were updated in virus definitions and would be detected by antivirus software (Chien, 2000). Still, some systems were infected after that date.

Tools for penetration testing can also maliciously launch attacks against unsuspecting unprotected targets. For example, Metasploit has a program known as MSFvenom which allows an attacker to create a trojaned version of a Windows application. This will still allow the application to retain full functionality but it will also contain embedded malware. A user who has the most basic protection from up-to-date free antivirus software could detect the trojaned application before it launches, yet substantial portions of the world's population do not have antivirus systems or keep them up-to-date which allows an attacker to have continued success by reusing the same malware.

3. Publicly Known Vulnerabilities

Users are also defenseless against vulnerabilities that have been publicly disclosed but unpatched by the vendor. An example is the Hot Potato exploit, which chains together

three different vulnerabilities in the Windows operating systems. The first exploit was released in 2014 by Google's Project Zero. There are also several readily available exploit tutorials on websites such as Reddit and YouTube about how to exploit this vulnerability using Metasploit's exploit module or a Powershell script. Hot Potato is still prevalent because it still has not been patched by Microsoft. Some of the vulnerabilities used in Hot Potato were reported in 2000 and Microsoft claims that they cannot patch it because it would "break compatibility between the different versions of their operating system" (Prabhu, 2016).

Another heavily used exploit takes advantage of the vulnerability CVE-2017-0199. It was first discovered in July 2016 and privately reported to Microsoft in October 2016 by Ryan Hanson. In January 2017 attacks began, and there was still no patch available. McAfee discovered the vulnerability, notified Microsoft on April 6, 2017, and then publicly released the vulnerability information the next day. On April 9, 2017, an exploit was for sale on the black market. Then on April 25, 2017 a patch was released by Microsoft, almost 9 months after discovery. Microsoft claimed that their delay in patching the vulnerability was because they wanted to "identify other potentially similar methods and ensure that our fix addresses more than just the issue reported" (Menn, 2017). The delay between discovery, public disclosure, and patch provided a large window of opportunity for attackers.

Open-source software-sharing websites such as Github allow attackers to post exploits that take advantage of known vulnerabilities. An example was an exploit for CVE-2017-0016 which was "a denial of service vulnerability that can crash Windows when connecting to a malicious SMB share" (Robinson, 2017). The vulnerability was initially found in September 2016, and an exploit was released on GitHub in February 2017. Microsoft issued a patch for the vulnerability on March 14, 2017 (Spring, 2017). The patch came alongside 17 additional security bulletins regarding completely different vulnerabilities, eight of which were critical and three of which had been publicly disclosed and had accompanying exploits (Robinson, 2017).

Few vulnerabilities remain unpatched for a significant amount of time. However, the window of opportunity provided by the public sharing of exploits that target known

unpatched vulnerabilities give attackers a major advantage, and allow for a greater window of opportunity to reuse the exploit.

4. The Human Element

Humans can be a key component in allowing attackers to reuse malware multiple times because an attack can involve deceiving them repeatedly (Symantec, 2017, p. 8). In 2016, there was a significant increase in email malware rates, targeted spear-phishing, and ransomware. Despite continued warnings to users about opening links in suspicious emails, users continue to fall prey to these kinds of methods which provide attackers with good ways for the delivery of malware. Attacks which leverage the human element do not need to rely on malware signatures bypassing anti-virus software to gain access to a system. They just need to coerce a user to click a malicious link which takes them to a website, which then downloads malicious software directly to the target's computer.

Social engineering is a timeless technique to gather information about a specific target to tailor phishing emails directly to them. Social engineering could also allow an attacker to gain information to be able to blackmail an individual, target their home network with the intention to pivot to their work network, or brute-force their credentials. Thumb drives can also be used to install malicious software directly on to a target network. Whether they are dropped in a parking lot and an unsuspecting user plugs it into the network out of curiosity or an insider deliberately plugs it in to a network, USB thumb drives are an effective means to target isolated networks.

In fact, CERT updated their definition of an insider threat in 2017 to include unintentional non-malicious actors (CERT, 2017). This is an important distinction because there is a significant difference between errors in source code from a mistake and software engineers who are deliberately writing errors into source code with the intent to later exploit it as a zero-day exploit. Similarly, users are not clicking on links in emails with the intention of compromising their company's network, but because they are cleverly crafted by an attacker to entice the user. Users will continue to provide an imperishable attack vector which allows for the window of opportunity to remain open and malware to be reused.

D. PERSISTENCE

Attackers can also gain continuing access through memory-based malware, fileless malware, or by manipulating the supply chain. Persistence can be defined as "any access, action, or configuration change to a system that gives an adversary a persistent presence on that system" (Mitre, 2017). A persistent presence on a system impedes the window of opportunity on the system from closing.

1. Memory-Resident Malware

Memory inside a computer is divided into cache, primary, and secondary memory. Cache memory is designed for temporary use and is faster than primary memory. Primary memory, or main memory, maintains the data and instructions the computer is currently using, and is divided into RAM (random-access memory) and ROM (read-only memory) (Tutorials Point, 2017). Malware that resides in memory "loads its code into that memory space and remains resident until it is accessed or reactivated" (McAfee, 2015). However, primary memory and cache are volatile meaning that memory-resident malware stored in either of these locations is lost when the power to the computer is turned off (Wueest & Anand, 2017, p. 10). The secondary memory is non-volatile meaning data is stored until explicitly deleted (Tutorials Point, 2017). This is a common place for attackers to install malware on target systems. Malware in secondary storage can also be encrypted, unlike malware in main memory, so it can be more difficult for antivirus software to detect it.

An example of a memory-resident attack was the Code Red worm which targeted "a vulnerability in Microsoft's IIS web servers" (Moore, Shannon, & Claffy 2002, p. 273). On June 18, 2001, Microsoft released the vulnerability information in Security Bulletin MS01-033 which urged system administrators to patch their systems (Dolak, 2001). Then on July 13, 2001, the Code RedI v1 worm was first discovered in use. This variant of the worm resided in primary memory, so a reboot would disinfect the machine. On July 19, 2001, a second variant called Code RedI v2 appeared which differed in the pseudo-random number generator it used to spread to other hosts. A machine could be infected multiple times unless the vulnerability was patched. A third variant known as

Code RedII began to spread on August 4, 2001 by exploiting the same vulnerability with completely different code. Code RedII created a "backdoor" giving itself remote access with administrator privileges, which gave it the persistence it needed to remain through system reboots (Moore et al., 2002).

Another technique used by attackers to establish persistence on a target machine is to insert itself in the operating system, often into the dynamic link libraries (DLL) within the Microsoft Windows operating system. M-Labs (2010) described how an attacker could gain persistence on a target machine by using DLL search-order hijacking. To get the program to upload the malware DLL instead of the legitimate DLL, the attacker can insert the malware DLL into the directory containing the program. If the malware DLL is in the directory with the same name as the legitimate DLL, the program will install the malware DLL and never check the folder where the legitimate DLL actually is. Microsoft is aware of the vulnerability, but claims that patching this would cause compatibility issues (M-Labs, 2010). This allows attackers to establish and maintain persistence through a single exploit.

2. Fileless Malware

Fileless malware can use the CPU registers which hold the instructions and data the CPU is currently working on. Fileless malware can evade "detection by hiding in the Microsoft Windows registry" (McAfee, 2015a, p. 3). Fileless malware gives the attacker persistence after the system reboots because the operating system loads the registry when the computer restarts (Semantic Security Response, 2015). An early example was Poweliks which uses a naming method to prevent users from finding it, along with a Watchdog process which continually verifies that "Poweliks is still running and that its registry subkeys have not been deleted" (Gossett, 2015).

Similarly to Windows registry malware, rootkits can give attackers persistence on target systems. Rootkits are "software that acquires and maintains privileged access to the operating systems (OS) while hiding its presence by subverting normal OS behavior" (Smith and Harrison, 2012, p. 2). Rootkits can subvert normal OS behavior by intercepting and responding to questions about files and registry keys to hide their

existence on machines. Both these techniques allow the attacker to remain on the targeted machine and continue to reuse a single exploit well after the initial installation.

3. Supply Chain Manipulation

Inserra and Bucci (2014) concluded that a big cybersecurity challenge is the vulnerability in the components that make up the supply chain by which digital devices and software are acquired. This includes hardware, firmware, and software. The cyber supply chain is a global market where components are developed in countries around the world, and oftentimes a single component requires elements from multiple countries. Shackleford (2015) states that supply chain issues include malware distributed to end users in software and firmware updates, disclosure of information to unauthorized personnel, and stolen vendor credentials that can later be used against an end user. There could also be product tampering where software or firmware errors are injected into the code, or hardware trojans or backdoors are installed into the hardware.

Inserra and Bucci (2014) suggest that what makes supply chain attacks so difficult to detect is that the end user ultimately has to identify what component led to the intrusion. This becomes difficult when testing must examine all of the software, firmware, and hardware used by the system. Hardware trojans are challenging to find because they are often designed to hide from testing procedures; they can be designed to only activate after certain actions or events.

E. VARIATIONS

Once a variant of malware is used and analyzed, antivirus software developers release updated signatures to prevent the reuse of the same variant of malware. Malware signatures are generally difficult to modify because they are usually created from the most critical portion of the malware that is difficult to change. However, a skilled malware author can vary the signatures of the malware through obfuscation techniques. In 2016 alone, there were “more than 357 million new variants observed” (Symantec, 2017, p. 38). Varying the malware’s signature may temporarily fool host-based antivirus

software. This allows the malware to have a greater window of opportunity for reusability.

1. Encryption and Packing

The ongoing arms race between antivirus companies and attackers continues as the antivirus companies try to keep up with the evolving methods used by attackers to obscure malware (Graveland, 2011). A technique used to bypass signature-based antivirus software is to encrypt the malware. Encrypted malware contains the decryptor and the encrypted main body. Each time the malware infects a new target, the malware can be encrypted with a different key. This means the malware looks different each time and its signatures will be different. However, the malware must be decrypted in main memory to enable it to be used, and antivirus software can detect it there. Furthermore, the decryptor usually remains constant from one infection to the next, so an antivirus signature can be written for just the decryptor (You & Yim, 2010). Malware can also be compressed or packed to try to evade signatures, but then it is easier to restore the original malware. Encrypted files are also easier to detect because they have very high entropy (data randomness), so even without a recognized signature, the target will know their system is under attack.

Another technique used by malware authors to hide their signatures from anti-virus software is packing. Software packers are designed to "compress and encrypt other executable files in a disk and restore the original executable images when the packed files are loaded into memories" (Yan, Zhang, & Ansari, 2008, p. 72). To detect malware after it has been packed, it needs to be unpacked so that the anti-virus software or security researcher can scan it for executable signatures. Most packing or compressing algorithms are well known; nonetheless, this approach allows for a window of opportunity where the malware could be reused (Yan et al., 2008).

2. Polymorphic and Metamorphic Malware

Malware authors also use techniques which permit malware to change over time. This is referred to as polymorphic malware. Examples of obfuscation techniques used

include dead-code insertion, register reassignment, subroutine reordering, instruction substitution, code transportation and code integration. Dead-code insertion adds instructions to change the appearance of the code but not the code's behavior. Register reassignment switches register assignments when this does not change functionality. Subroutine reordering changes the order of the subroutines, which rarely changes functionality. Instruction substitution replaces the original instruction with an equivalent instruction. Code transposition changes the sequence order of the instructions, and code integration attaches and intertwines to a target program. But as discussed earlier, it is difficult for obfuscation to change key aspects of the malware which provide the best signatures. Another technique to combat obfuscation is to run the malware in a safe area or "sandbox," allowing it to decrypt itself and reveal its signatures.

Polymorphic malware developers can try to combat detection with "metamorphic" malware which uses obfuscation techniques, but where the main body is never exposed in memory so a signature cannot be written against it (You & Yim, 2010). However, writing such malware is difficult. Christiansen (2010) suggests that malware which has a morphing mechanism inside, such as polymorphic malware, gives the antivirus developer an advantage because once they have the malware, it just has to be reverse engineered to find a specific signature or behavior.

IV. STRATEGIC IMPLICATIONS OF A REUSABLE CYBERWEAPON

A. OVERVIEW

The previous chapter identified techniques the attacker can leverage to increase the window of opportunity between vulnerability discovery and patch implementation. However, using some of these techniques on generic malware differs from using them on a highly tailored cyberweapon. Generic malware is designed to be used on a large scale and distributed to many people simultaneously. It can often be reused because the attackers can rely on mistake-prone ordinary people. The success of the attack depends upon the likelihood that a portion of the population will be enticed to click links embedded with malware, or that enough people will fail to patch their operating systems, install anti-virus software, or keep it up-to-date with the latest malware definitions.

However, attackers cannot rely on random user errors to get system access when using cyberweapons. Most cyberweapons contain payloads that are highly tailored to a specific system and are designed to carry out a specific effect in a single strike, much like conventional armaments. Due to their specificity, cyberweapons are not traditionally designed to be used more than once, but some could be. A cyberweapon intended for single use tends to have overt payload effects. They are distinct and dramatic effects which bring government and media attention, encourage allocation of resources for better understanding, and prompt investigation to determine exactly what vulnerabilities led to the effects. In turn, this may cause the window of opportunity for reuse to close quickly for the original target and also for any other susceptible targets. However, usually it is irrelevant whether or not the vulnerability is discovered and patched after the cyberattack because the mission will have already been accomplished, such as with missions that demonstrate strength or provide temporary area denial.

A reusable cyberweapon should be designed with its survivability in mind. Its stealthy effects could allow it to accomplish the mission prior to being detected. A cyberweapon that goes undetected allows the window of opportunity to remain open.

B. SURVIVABILITY

Knowing a cyberweapon's probability of surviving an attack is especially critical when offensive actions must be continued over a period to accomplish the objective. Knowing a cyberweapon's survivability metric is also beneficial if the intention is to later reuse this weapon against other targets. A cyberweapon with a high probability of survivability has a high suitability for reuse.

A model can be adapted from the aircraft combat survivability model created by Ball (2003). Survivability is measured by how killable the aircraft is based on the aircraft's susceptibility and vulnerability. Susceptibility is defined as "the inability of an aircraft to withstand the man-made hostile environment" (Ball, 2003, p. 1). Vulnerability is defined as "the inability of an aircraft to avoid the guns, approaching missiles, exploding warheads, air interceptors, radars, and all of the other elements of an enemy's air defense that make up the man-made hostile mission environment" (Ball, 2003, p. 1).

Survivability can be measured using a probability on a scale from 0 to 1. It could refer to the probability the aircraft will survive the entire mission or the probability it will survive an attack from a specific threat. Killability, the opposite of survivability, could refer to a downed aircraft following an attack or an aborted mission where the aircraft was damaged from an attack and had to return to base. Ball models survivability as one minus the product of susceptibility and vulnerability. To measure susceptibility of an aircraft, a model is needed of the sequence of events between when the aircraft encounters a threat and until there is a successful attack against the aircraft. This is known as a flyout model. The higher the capability of the threat to detect the aircraft and successfully target it, the higher the killability of the aircraft. The aircraft's vulnerability depends on the predicted threat that the aircraft may face during the mission. It also depends on the threat's capability to hit critical components on the aircraft. The higher the effectiveness of the threat, the higher the killability of the aircraft (Ball, 2003).

1. Applying Survivability to Cyberweapons

The aircraft survivability model can be applied to cyberweapons. Survivability of a cyberweapon is its capability to avoid or withstand target-detection systems. Detection

systems include antivirus software, host-based or network-based intrusion-detection systems, and other software looking for indicators of compromise. It also includes manual detection capabilities such as the training and overall level of cyber-security knowledge of the personnel at the target and their capability to understand their logs and equipment and realize that they have been attacked.

There are two kinds of survivability: a cyberweapon that survives an attack on the system it targeted, and a cyberweapon that survives over multiple uses without becoming obsolete. Some objectives will be tailored around one kind of survivability, and some will be tailored to fit both. However, a survivable cyberweapon of either kind is valuable.

A cyberweapon can be killed in two ways: vulnerability patching prior to mission accomplishment, or removal of the cyberweapon from the target system prior to mission accomplishment. Killability is contingent on how susceptible and vulnerable the cyberweapon is to the specific target and the effectiveness of the cyberweapon's capabilities. Susceptibility is the inability of a cyberweapon to avoid target-detection systems. Vulnerability is the inability of a cyberweapon to withstand target-detection and neutralization systems.

2. Deception and Concealment of Cyberweapons

To survive, a cyberweapon must persist on the target system, meaning it must deceive the target and conceal its effects. Therefore, the ideal cyberweapon is one in which susceptibility and vulnerability are as close to zero as possible, thus minimizing the possibility for the cyberweapon to be killed.

Susceptibility is lowest when a cyberweapon fully avoids target-detection systems. Several attributes can contribute to lower susceptibility, but the most effective for a cyberweapon is stealth or its capability to conceal effects. Stealth could refer to the cyberweapon's propagation method going unnoticed by the target detection systems, its inability to be located on the target system, or the effects of the payload after it is installed as an implant on the target system. Stealth could also refer to the cyberweapon's capability to deceive the target's operators by concealing the effects of organizational routines (Gartzke & Lindsay, 2015).

Vulnerability is lowest when a cyberweapon can be discovered by the target's detection systems but still accomplish the mission. Redundancy can lower the vulnerability, analogously to how an aircraft with a single engine is more likely to be disabled with a single-shot attack. Redundancy in a cyberweapon could mean that discovery of one method of the weapon will not prevent it from accomplishing its mission with another method. Vulnerability can also be reduced by hardening the cyberweapon to make it more persistent between detection and vulnerability patching. Chapter 3 listed several techniques to harden malware or cyberweapons to make them more difficult to remove, such as obfuscating the cyberweapon's code by using encryption or packing techniques, inserting cyberweapons into the cache or software registry so they will be reinstalled if deleted, attacking obsecure programs or services that are not invoked often, or using zero-day attacks so the weapon is harder to find (Rowe, 2015). There is also an opportunity to influence the supply chain by tampering with hardware, firmware, and software to install hardware trojans or backdoors through deliberate errors that may be inputed into the code.

These techniques could extend the lifetime of the cyberweapon considerably. Once the cyberweapon is discovered, the target needs to reverse-engineer the cyberweapon, determine how persistence was established, determine what vulnerability was exploited, and patch the vulnerability. The cyberweapon could try to make all these difficult. For example, it could delay the amount of time before the weapon is first detected, the amount of time before it is found on the target, the amount of time before it is removed, and so on. This would allows the window of opportunity to remain open for a longer period, which could allows for reuse. Reuse could include attacking different aspects of the same target or sending different payloads.

A great amount of costs in time, effort, and resources are necessary to make a cyberweapon more resilient. When deciding where in the software the cyberweapon will mount its attack from, there is a tradeoff between the level of difficulty and resilience. For instance, a cyberweapon can attack the hardware, but it would require a difficult process that would likely involve the supply chain. The attacker would need to insert malicious code into the hardware design or implementation, then ensure that the device

that was altered gets installed on the equipment that is later going to be targeted. This requires much time and effort and has a high level of risk, but would provide an extremely persistent cyberweapon that would be very difficult to locate on the system. Conversely, it is easy for an attacker to mount an attack from secondary storage, but anti-malware software scans such storage repeatedly. Main memory and cache memory are also easier to mount an attack from, but they are volatile and the system will delete traces of the malware when the system is powered off, therefore requiring that they be reloaded with malware when power is restored. The registry and the boot memory are good for malware to use when it needs stealth and persistence, but they are difficult to modify.

C. DESIGNING A SURVIVABLE CYBERWEAPON

Depending on the mission objective and type of adversary, each cyberweapon needs a different design approach. Figure 2 has been adapted from the aircraft survivability model (Ball, 2003) to create a new method which accounts for survivability in the cyberweapon design process. Design begins by describing what the mission objective is, including its intended effects. A cyber capability is designed to exploit a specific vulnerability on the target system. A determination will be made of the best place to hide the cyberweapon on a target system according to the type of adversary and the type of persistence that is desired. The next step is a killability assessment including vulnerability and susceptibility. In subsequent wargaming, it is necessary to simulate forces with which the cyberweapon may come in contact while on mission. These include vulnerability discovery and public disclosure by a third party which results in a patch or the inadvertent discovery of the system intrusion, or a target-system configuration change that exposes the cyberweapon or nullifies its effect. The result of wargaming is a mission-success assessment. If the cyberweapon does not meet the survivability requirements that are necessary to accomplish the mission, the weapon can be refined.

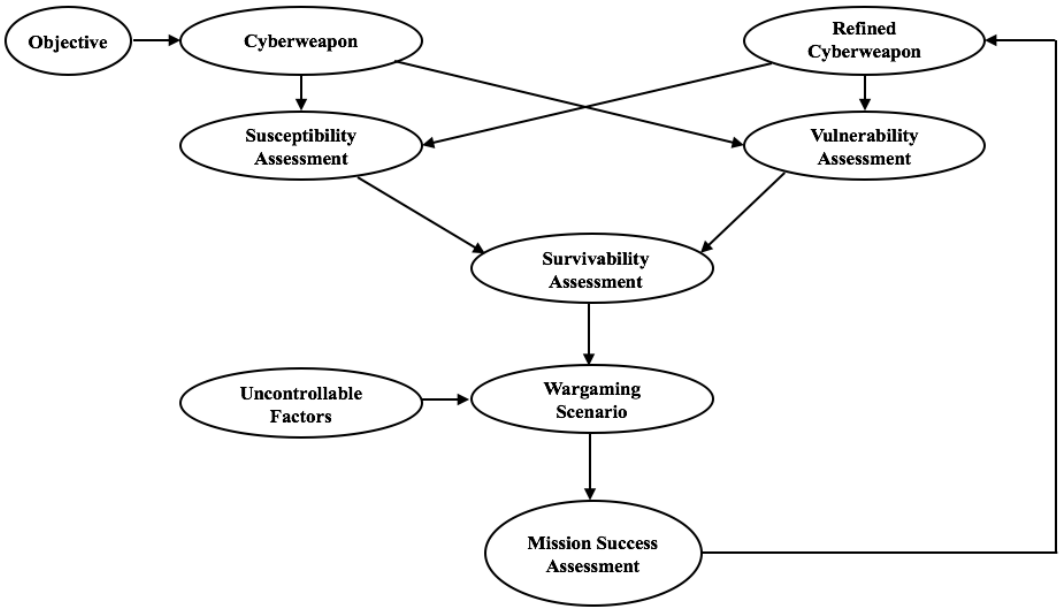

Figure 2. Survivability design model. Adapted from Ball (2003).

1. Determining Probability of Mission Success

To determine the overall probability of mission success, the cyberweapon must survive the propagation to the target and installation. Then if it remains undetected, it may be available for reuse on the same system or others. If the cyberweapon is discovered prior to mission accomplishment, there is a good probability that it will be killed prior to accomplishing its objective. However, a cyberweapon may still be able to accomplish the objective after being discovered depending on how long the window of opportunity remains open on the target system. Note that a detected cyberweapon may still be available for reuse against other targets if the first target does not ever disclose it.

The tree diagram in Figure 3 has been adapted from the aircraft mission success model that was developed by Ball (2003) to demonstrate what conditions must be met for a cyberweapon to survive a mission and be available for reuse.

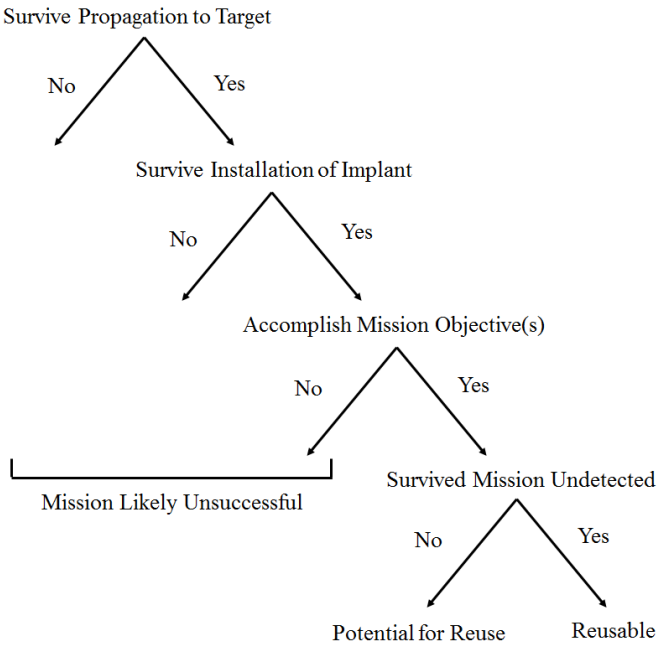

Figure 3. Tree diagram for static mission success. Adapted from Ball (2003).

2. Determining Measure of Effectiveness over Time

A cyberweapon should be continually reevaluated to ensure it is still accomplishing the intended objective and has not been detected. This is especially important when a cyberweapon can be reused against the same target or others. If the weapon was detected by the target, there is still an opportunity to continue the mission but it becomes risky. The options are to either discontinue the mission and potentially forego reuse of the cyberweapon in the future, or leave the weapon in place and use the foothold to install a different form of persistence. If the decision is made to discontinue the mission, the implant can be fully removed or just rendered inactive. This is an important distinction because an inactive implant could be reused in the future. In addition, lengthy missions may also decide to terminate a cyber-operation prior to the cyberweapon being detected. Figure 4 provides a flowchart for continuously reevaluating a cyberweapon's mission effectiveness.

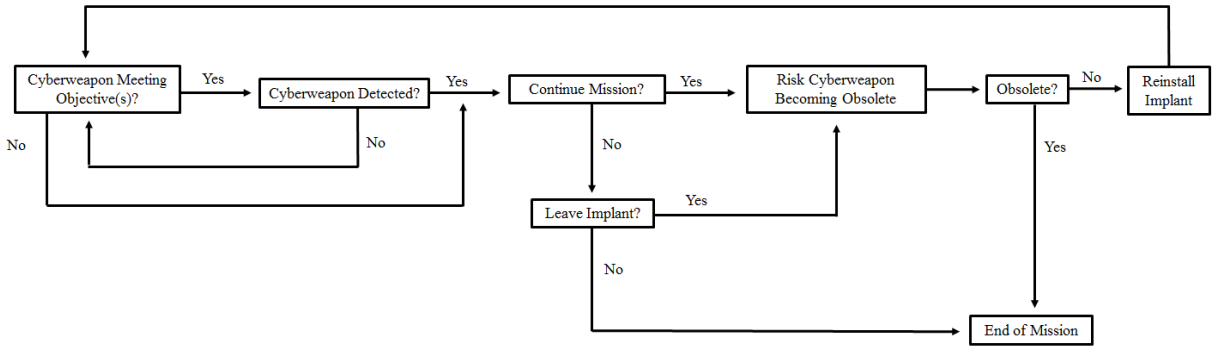

Figure 4. Flowchart for continuous mission reevaluation

V. CONCLUSION

A. OVERVIEW

Cyberweapons are traditionally thought of as perishable, use-and-lose weapons where once the capability has been used, it is revealed to the public, and soon a patch will be released that removes most of the value of the cyberweapon in the future. However, this thesis showed that cyberweapons can be effectively used more than once. In a vulnerability's life cycle, there is a window of opportunity to re-exploit the vulnerability after it is discovered and before it is patched. An attacker can prolong the window of opportunity through obfuscation and concealment of the cyberweapon's signatures. The ability for a cyberweapon to survive a mission undetected is enhanced by stealth since that gives it a higher the probability of surviving.

To begin patching and cause the window of opportunity to close for an attacker, the target must realize they have been attacked. This includes recognizing the attack, finding the vulnerability that was exploited, and then creating and implementing a patch for the vulnerability. This can take time. Until discovered, there is still an opportunity to reuse the cyberweapon against the same target by re-exploiting the unpatched vulnerability. Then until the attack has been publicized, there will be opportunities to reuse the cyberweapons against different targets. Patching itself may be difficult for key infrastructure systems that must be kept running continuously, so the vulnerability window may be even larger for them. It is not realistic for a major government installation to bring down their entire infrastructure while they search for, create, and implement a patch on all affected systems.

B. POLICY RECOMMENDATIONS

Historically, cyber attacks by nation-states have yielded effects that have been very overt and are highly destructive or disruptive. Attacks can be overt to assert dominance, deter, insert influence, for revenge, or for political effect. Oftentimes overt weapons with noisy effects cannot be reused. However, if the desire is to retain the

capability of the cyberweapon and reuse it, the cyberweapon must be designed with survivability in mind.

1. The Benefits of a Reusable Cyberweapon

When the cyberweapon is under development, it must be considered whether it will be designed as a traditional overt single-use cyberweapon or a stealthy reusable cyberweapon. There are benefits to both, but a reusable cyberweapon with stealthy effects can provide an attacker with a strategic advantage over an overt single-use cyberweapon. For instance, if a target organization is developing a capability that the cyberattacker wanted to prevent, a cyberweapon could infiltrate the supply chain and disable equipment in a non-alarming manner that would appear to be general machine malfunction. Alternatively, an attacker could slow down an adversary network so that it would appear to the system administrators as routine updates or software malfunctioning. A cyberweapon could also place a multi-method implant on a target system so that the target could realize they have been attacked, but be unable to stop it because they cannot find all the methods. Being stealthy and persistent is a tactical advantage for the commander because it allows the window of opportunity to remain open for a longer period which enables meeting follow-on objectives using cyber or non-cyber means.

2. The Debate on Cyberweapon Stockpiling

Chapter 2 introduced the issue of whether there is a greater benefit to stockpiling cyber capabilities versus developing the capabilities as they are required. The decision to use versus stockpile the weapon depends on the ability to predict when the cyberweapon will be killed. A cyberweapon on the shelf risks becoming obsolete prior to use, but a cyberweapon in use has the potential to become perishable because the vulnerability will be patched.

A reusable cyberweapon has a lower risk of being killed in use. Therefore, the issue with a reusable cyberweapon is whether it has a higher chance of being killed on the shelf or in use, and also whether the target is capable of killing the cyberweapon. Therefore, the risk-assessment decision becomes whether it is more beneficial to use the

weapon against a specific target, or delay the use of the weapon until it can be assured that it is being used against a target that is not capable of killing it. The cyberweapon's susceptibility and vulnerability assessments aid in this decision. If all cyberweapons are designed with survivability in mind, it decreases the need to stockpile them.

3. Title 10 and Title 50 Policy Discussion

Title 10 of the U.S. Code defines Department of Defense (DOD) and military operations, whereas Title 50 defines the operations conducted by the intelligence community. Distinguishing between the two depends on whose authorization the mission commander is operating. If the effect of the action is not apparent and is never intended to be known or publicly acknowledged, then it is considered covert action which is a Title 50 authority. Military operations, or Title 10 authorities, must publicly acknowledge the United States' role in the operation or it must be apparent (Wall, 2011). This is an important distinction because cyberweapons can be designed with or without the intention to be attributed to the United States.

Cyberweapon reusability impacts the Title 10 and Title 50 policy discussion in two ways. First, the cyberweapon can be reused in a single mission to send a variety of payloads to the target. This allows a cyberweapon to potentially switch authorities mid-mission from a Title 50 authority to Title 10, or vice versa. A determination would need to be made on which authority would take precedence depending on whether there is a greater benefit for the intelligence community or the military. For example, if the cyberweapon began as a covert action, then there would need to be a risk determination on whether it is more strategically beneficial to switch to a Title 10 authority and acknowledge involvement. The disadvantage in acknowledging it is that this would allow the target to begin searching for the cyberweapon and then patch the vulnerability that was exploited, which could prevent the cyberweapon from being reused. However, Wall (2011) argues that a military operation does not require a press release announcing United States involvement. Therefore, if the operation goes undetected and the United States is never asked about their involvement, then they will not need to acknowledge it.

The intent to acknowledge the operation if asked is different from the actual acknowledgement itself.

Second, the reusable characteristic of a cyberweapon allows it to be used under different authorities each time it is used against a new target. Determining the authority that the cyberweapon falls under will depend on the desired mission goal and whether the effects are intended to be overt or stealthy. For example, suppose an attacker operates under Title 50 authority against the first target and Title 10 authority against the next target. When the effects are acknowledged by the attacker against the second target, the first target may not even be aware they have been attacked. To fully take advantage of the benefits a reusable cyberweapon can have in the Title 10 and Title 50 discussion, there needs to be an updated policy documenting the authorization process specific to reusable cyberweapons.

C. FUTURE RESEARCH

The reusability characteristic of a cyberweapon can alter a number of policy discussions and operating requirements. Future research could focus more specifically on how to integrate the issues discussed here with current doctrine. Also, it would be helpful to focus on developing a quantitative way to measure cyberweapon survivability through measures of effectiveness and measures of performance, and then integrate these measurements into current cyberweapon development and monitoring. Lastly, future research could compare costs and benefits between specific single-use cyberweapons and reusable cyberweapons to see which is more cost-effective.

LIST OF REFERENCES

Ablon, L., & Bogart, A. (2017). *Zero days, thousands of nights* (Research Report No. 1751). Retrieved from http://www.rand.org/t/RR1751

Arbaugh, W., Fithen, W., & McHugh, J. (2000). Windows of vulnerability: A case study analysis. *Computer*, 33(12), 52–59. http://dx.doi.org/10.1109/2.889093

Arora, A., Krishnan, R., Nandkumar, A., Telang, R., & Yang, Y. (2004). Impact of vulnerability disclosure and patch availability—an empirical analysis. *Third Workshop on the Economics of Information Security*, 24, 1268–1287. Retrieved from https://www.dtc.umn.edu/weis2004/telang.pdf

Ask, K. (2006). *Automatic Malware Signature Generation*. Gecode. Retrieved from http://www.gecode.org/~schulte/teaching/theses/ICT-ECS-2006-122.pdf

Axelrod, R., & Iliev, R. (2014). Timing of cyber conflict. *Proceedings of the National Academy of Sciences of the United States of America*, 111(4), 1298–1303. http://dx.doi.org/10.1073/pnas.1322638111

Ball, R. E. (2003).*The fundamentals of aircraft combat survivability analysis and design, second edition*. Reston, VA: American Institute of Aeronautics and Astronautics, Inc.

Bilge, L., Dumitras, T. (2012). Before we knew it: An empirical study of zero-day attacks in the real world. *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, 833–844. http://dx.doi.org/10.1145/2382196.2382284

Bonfante, G., Kaczmarek, M., & Marion, J. (2008). Architecture of a morphological malware detector. Journal in Computer Virology, 5(3), pp. 263–270. http://dx.doi.org/10.1007/s11416-008-0102-4

CERT. (2017). CERT Insider Threat Center. Retrieved from http://www.cert.org/insider-threat/cert-insider-threat-center.cfm

Chien, E. (2000). *VBS.LoveLetter.Var*. Symantec Corporation. Retrieved from https://www.symantec.com/security_response/writeup.jsp?docid=2000-121815-2258-99&tabid=3

Christiansen, M. (2010). *Bypassing malware defenses* [White Paper]. SANS Institute. Retrieved from SANS Institute InfoSec Reading Room website: https://www.sans.org/reading-room/whitepapers/testing/bypassing-malware-defenses-33378

Cylance Threat Guidance Team. (2017, April 4). Threat spotlight: The truth about fileless malware [Blog Post]. Retrieved from https://www.cylance.com/en_us/blog/threat-spotlight-the-truth-about-fileless-malware.html

Dolak, J. (2001). *The Code Red worm* [White Paper]. SANS Institute. Retrieved from SANS Institute InfoSec Reading Room website: https://www.sans.org/reading-room/whitepapers/malicious/code-red-worm-85

Dunn, J. (2017, May 15). A huge number of PCs still use ancient Windows software that puts them at risk. Retrieved from http://www.businessinsider.com/how-many-people-use-windows-xp-chart-2017-5

Gartzke, E., & Lindsay, J. (2015). Weaving tangled webs: Offense, defense, and deception in cyberspace. *Security Studies*, 24(2), 316–348. http://dx.doi.org/10.1080/09636412.2015.1038188

Gossett, K. (2015, June 9). Poweliks click-fraud malware goes fileless in attempt to prevent removal [Blog Post]. Retrieved from https://www.symantec.com/connect/blogs/poweliks-click-fraud-malware-goes-fileless-attempt-prevent-removal

Graveland, B. (2011, January 20). Fully undetectable cryptors and the antivirus detection arms race [Blog Post]. Retrieved from https://www.symantec.com/connect/blogs/fully undetectable-cryptors-and-antivirus-detection-arms-race

Herr, T. (2014). PrEP: a framework for malware & cyber weapons. *The Journal of Information Warfare*, 13(1). http://dx.doi.org/10.2139/ssrn.2343798

Herr, T., & Armbrust, E. (2015). Milware: Identification and implications of state authored malicious software. *Proceedings of the 2015 New Security Paradigms Workshop*, 29–43. http://dx.doi.org/10.1145/2841113.2841116

Herr, T., & Rosenzweig, P. (2014). Cyber weapons and export control: Incorporating dual use with the PrEP model. *Journal of National Security Law and Policy*, 8(2), 301–319. http://dx.doi.org/10.2139/ssrn.2501789

Herr, T., & Schneier, B. (2017). Taking stock: estimating vulnerability rediscovery. *Social Science Research Network (SSRN).* http://dx.doi.org/10.2139/ssrn.2928758

Hichkad, R., & Bowie, C. (2012). Secret weapons & cyberwar. *Armed Forces Journal*. Retrieved from http://armedforcesjournal.com/secret-weapons-cyberwar-2/

Huntley, W. (2016). Strategic implications of offense and defense in cyberwar. *2016 49th Hawaii International Conference on System Sciences*. http://dx.doi.org/10.1109/HICSS.2016.691

Inserra, D., & Bucci, S. (2014). Cyber supply chain security: A crucial step toward U.S. security, prosperity, and freedom in cyberspace. *The Heritage Foundation*. Retrieved from http://www.heritage.org/defense/report/cyber-supply chain-security-crucial-step-toward-us-security-prosperity-and-freedom

Kingsley-Hughes, A. (2017, May 15). Stop disabling automatic updates, people. Retrieved from http://www.zdnet.com/article/stop-disabling-automatic-updates-people/

Kovacs, E. (2017, January 23). Heartbleed still affects 200,000 devices: Shodan. Retrieved from http://www.securityweek.com/heartbleed-still-affects-200000-devices-shodan

M-Labs. (2010, July 15). Malware persistence without the Windows registry [Blog Post]. Retrieved from https://www.fireeye.com/blog/threat-research/2010/07/malware-persistence-windows-registry.html

McAfee. (2015). *McAfee Labs Threats Report*. McAfee Inc. Retrieved from https://www.mcafee.com/us/resources/reports/rp-quarterly threats-nov-2015.pdf

McAfee. (2015a). *Protecting Against Fileless Malware*. McAfee Inc. Retrieved from https://www.mcafee.com/us/resources/solution-briefs/sb-quarterly threats-nov-2015-1.pdf

Meisner, J. (2013, April 17). Latest security intelligence reports shows 24 percent of PCs are unprotected [Blog Post]. Retrieved from https://blogs.microsoft.com/blog/2013/04/17/latest-security-intelligence-report-shows-24-percent-of-pcs-are-unprotected/

Menn, J. (2017, April 26). Hackers exploited Word flaw for months while Microsoft investigated. *Reuters*. Retrieved from http://ca.reuters.com/article/technologyNews/idCAKBN17S32G-OCATC

Microsoft. (2015). *Microsoft security intelligence report, volume 18*. Microsoft Corporation. Retrieved from https://www.microsoft.com/en-us/download/details.aspx?id=46928

Microsoft. (2017). *Microsoft security bulletin MS17-010—critical*. Microsoft Corporation. Retrieved from https://technet.microsoft.com/en-us/library/security/ms17-010.aspx

Mitre. (2017). *Persistence*. Mitre Corporation. Retrieved from https://attack.mitre.org/wiki/Persistence

Moore, D., Shannon, C., & Claffy, K. (2002). Code red: a case study on the spread and victims of an Internet worm. *Proceedings of the 2nd ACM SIGCOMM Workshop*

on the Internet Measurement 2002, France, pp. 273–284. http://dx.doi.org/10.1145/637201.637244

Moussouris, K., & Siegel, M. (2015). The wolves of Vuln Street: The 1st dynamic systems model of the 0day market. Retrieved from RSA Conference USA 2015 website: https://www.rsaconference.com/events/us15/agenda/sessions/1749/the-wolves-of-vuln-street-the-1st-dynamic-systems

Olenick, D. (2016, November 21). Happy birthday Conficker: malware hits 8. Retrieved from https://www.scmagazine.com/happy-birthday-conficker-malware-hits-8/article/574419/

Ozment, A. (2005). The likelihood of vulnerability rediscovery and the social utility of vulnerability hunting. Retrieved from https://pdfs.semanticscholar.org/1e83/3bab15a975dd71af5fcbbaed4a8df3a5be8a.pdf

Prabhu, V. (2016, January 24). Windows 7/8/8.1/10 vulnerable to Hot Potato exploit by hackers. Retrieved from https://www.techworm.net/2016/01/windows-7-8-8-1-10-vulnerable-to-hot-potato-exploit-by-hackers.html

Rid, T., & McBurney, P. (2012). Cyber-weapons. *The RUSI Journal*, 157:1, 6–13. http://dx.doi.org/10.1080/03071847.2012.664354

Robinson, T. (2017, March 14). Patch Tuesday: Microsoft releases 18 security bulletins, 18 critical. Retrieved from https://www.scmagazine.com/patch-tuesday-microsoft-releases-18-security-bulletins-8-critical/article/644148/

Rowe, N. (2015). Finding contextual clues to malware using a large corpus. *3rd IEEE International Workshop on Security and Forensics in Communication Systems 2015*, pp.229-236. http://dx.doi.org/ 10.1109/ISCC.2015.7405521

Rudis, B. (2017, May 12). Wanna decryptor (WNCRY) ransomware explained [Blog Post]. Retrieved from https://community.rapid7.com/community/infosec/blog/2017/05/12/wanna-decryptor-wncry-ransomware-explained

Schneier, B. (2000). Full disclosure and the window of exposure. *Crypto-Gram*. Retrieved from https://www.schneier.com/crypto-gram/archives/2000/0915.html

Schwartz, M. (2017, July 13). Eternally blue? Scanner finds EternalBlue still widespread [Blog Post]. Retrieved from http://www.bankinfosecurity.com/blogs/eternally blue-scanner-finds-eternalblue-still-widespread-p-2512

Shackleford, D. (2015). *Combatting cyber risks in the supply chain* [White Paper]. SANS Institute. Retrieved from SANS Institute InfoSec Reading Room website: https://www.sans.org/reading-room/whitepapers/analyst/combatting-cyber-risks-supply chain-36252

Smith, S., & Harrison, J. (2012). *Rootkits* [White Paper]. Symantec Corporation. Retrieved from Symantec Corporation website: https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/rootkits-12-en.pdf

Shazad, M., Shafiq, M., & Liu, A. (2012). A large scale exploratory analysis of software vulnerability life cycles. *Proceedings of the 34th International Conference on Software Engineering*, 771–781. Retrieved from http://dl.acm.org/citation.cfm?id=2337314

Smeets, M. (2017). A Matter of time: On the transitory nature of cyberweapons. *Journal of Strategic Studies*, 1–28. http://dx.doi.org/10.1080/01402390.2017.1288107

Spring, T. (2017, February 23). Publicly disclosed Windows vulnerabilities await patches. Retrieved from https://threatpost.com/publicly disclosed-windows-vulnerabilities-await-patches/123833/

Stockton, P., & Golabek-Goldman, M. (2013). Curbing the market for cyberweapons. Yale Law and Policy Review, 32(1), 101–128. Retrieved from http://digitalcommons.law.yale.edu/ylpr/vol32/iss1/11

Symantec. (2017). *Internet Security Threat Report Volume 22*. Symantec Corporation. Retrieved from https://www.symantec.com/security-center/threat-report

Symantec Security Response. (2015, September 24). Kovter malware learns from Poweliks with persistent registry update [Blog Post]. Retrieved from https://www.symantec.com/connect/blogs/kovter-malware-learns-poweliks-persistent-fileless-registry-update

Symantec Security Response. (2017, June 27). Petya ransomware outbreak: Here's what you need to know [Blog Post]. Retrieved from https://www.symantec.com/connect/blogs/petya-ransomware-outbreak-here-s-what-you-need-know

Tutorials Point. (2017). Computer—Memory. Tutorials Point. Retrieved from https://www.tutorialspoint.com/computer_fundamentals/computer_memory.htm

U.S. Joint Chiefs of Staff. (2013). *Cyberspace Operations*. Joint Publication 3–12R. Washington, DC: U.S. Joint Chiefs of Staff, February 5, 2013. Retrieved from https://cle.nps.edu/access/content/group/0b41fea9-fc21-4d06-8afd-fe9533799b20/JointPubs/jp3_12r.pdf

US-CERT. (2014). *Alert (TA14-098A)*. United States Computer Emergency Readiness Team. Retrieved from https://www.us-cert.gov/ncas/alerts/TA14-098A

US-CERT Publications. (2009). *Security tip (ST04-005).* United States Computer Emergency Readiness Team. Retrieved from https://www.us-cert.gov/ncas/tips/ST04-005

Vatu, G. (2017, January 23). OpenSSL bug heartbleed still affects some 200,000 websites. Retrieved from http://news.softpedia.com/news/openssl-bug-heartbleed-still-affects-some-200-000-websites-512117.shtml

Wall, A. (2011). Demystifying the Title 10-Title 50 debate: distinguishing military operations, intelligence activities & covert action. *Harvard Law School National Security Journal*, 3(1), pp. 85–142. Retrieved from http://harvardnsj.org/wp-content/uploads/2012/01/Vol-3-Wall.pdf

Ward, M. (2010, May 4). A decade on from the ILOVEYOU bug. *BBC News*. Retrieved from http://www.bbc.com/news/10095957

Wilson, A., Schulman, R., Bankston, K., & Herr, T. (2016). Bugs in the system. Open Technology Institute. Retrieved from https://www.newamerica.org/oti/policy-papers/bugs-system/

Wueest, C., & Anand, H. (2017). *ISTR Living off the land and fileless attack techniques* [White Paper]. Symantec Corporation. Retrieved from Symantec Corporation website: https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-living-off-the-land-and-fileless-attack-techniques-en.pdf

Yan, W., Zhang, Z., & Ansari, N. (2008). Revealing Packed Malware. *IEEE Security & Privacy*, 6(5), pp. 72–76. http://dx.doi.org/10.1109/MSP.2008.126

Yang, D., Usynin, A., & Hines, J. (2006). Anomaly based intrusion detection for SCADA systems. *American Nuclear Society*. Retrieved from https://pdfs.semanticscholar.org/1af8/4c9c62fb85590c41b7cfc9357919747842b2.pdf

You, I. & Yim, K. (2014). Malware obfuscation techniques: a brief survey. *Proceedings of the Fifth International Conference on Broadband and Wireless Computing, Communications and Applications*, Japan, pp. 297–300. http://dx.doi.org/10.1109/BWCCA.2010.85

Yu, F., Chen, Z., Diao, Y., Lakshman, T., & Katz, R. (2006). Fast and memory-efficient regular expression matching for deep packet inspection. *Proceedings of the 2006 ACM/IEEE Symposium on Architecture for Networking and Communications Systems*, USA, pp.93-102. http://dx.doi.org/10.1145/1185347.1185360

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California