Cover Sheet

Symposium: 23rd International Command and Control Research & Technology Symposium, November 2018

Topic: 10: Cyberspace Challenges

Title: Options for Persistence of Cyberweapons

Author information:
Carissa G. Hall and Neil C. Rowe
U.S. Naval Postgraduate School
GE-328, CS/Rp, 1411 Cunningham Road, NPGS
Monterey, CA 93943 US
carissaghall@outlook.com, ncrowe@nps.edu

Options for Persistence of Cyberweapons

Carissa G. Hall and Neil C. Rowe
U.S. Naval Postgraduate School
GE-328, CS/Rp, 1411 Cunningham Road, NPGS
Monterey, CA 93943 US
carissaghall@outlook.com, ncrowe@nps.edu

Abstract

A cyberweapon is weaponized software code that exploits flaws in software. It is only effective while the flaw still exists. Because of this, there is usually only a small window of time when a particular cyberweapon can be used. Some have argued 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 for a while. 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 making it harder to locate and remove the weapon. This can be accomplished by incorporating survivability ideas into the weapon's design requirement. This paper 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.

1. Introduction

Exploitable flaws in software code that are unknown to the software manufacturer are 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 and 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 and 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, cyberweapons generally have only a window of time when they can be effectively used or reused.

A factor driving the time-sensitive decision of when to use a cyberweapon is the principle of obsolescence. The longer the attacker holds onto the cyberweapon, the more likely the vulnerability is to be discovered and patched, thus rendering the cyberweapon obsolete (Huntley, 2016). A vulnerability could be discovered prior to its use if an insider exposes the vulnerability, or it is discovered during a penetration test, or it is 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

However, neutralization of a cyberweapon 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. Then they (or the software vendor or some agency working on their behalf such as an incident response team) must locate the vulnerability that was exploited. This process includes reverse-engineering the cyberweapon to identify signatures and the cyberweapon's targets within software, and the weapon may have been designed to erase its tracks by 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 it. 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.

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, even a higher-level malware can be reused against the same or different targets, if for example the cyberweapon could target the same vulnerability on platforms that are geographically dispersed. 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, so that a reuse will have a better chance of success.

2. Previous work

2.1. Cyberweapons and vulnerabilities

(Herr, 2014) introduced a model for describing cyberweapons which they refer to as the PrEP framework. The model identifies a propagation method, an exploit, and a payload. The propagation can be done through removable media, compromised certificate authorities, compromised email attachments, or compromised websites, and could have multiple stages. The exploit is code that takes advantage of a vulnerability in the target in the operating system, the browser, the firmware, or even in the hardware. The payload is the weapon itself, the code that accomplishes a goal such as espionage, persistence, or denial of service.

Cyberweapons exist because of the vulnerabilities in software code. To explain the lifecycle a vulnerability undergoes, (Arbaugh et al, 2000) introduced a lifecycle model that consists of vulnerability birth, discovery, disclosure, correction, publicity, scripting, and death. Birth refers to when the flaw was written into the code; discovery when an actor finds the vulnerability; disclosure when the actor reveals information about the vulnerability; and correction when the vulnerability is patched by the software vendor. Publicity refers to when the public learns of the vulnerability; scripting refers to when an attacker exploits the vulnerability; and death refers when the vulnerability is no longer exploitable.

(Wilson et al, 2016) introduce the concept of a vulnerability ecosystem of actors who are interested in finding software vulnerabilities: independent agents, small teams, larger teams, and governments. Independent agents could be security researchers or amateur hackers; small teams are academic labs or security firms; larger teams are the large technology companies; and governments are state-sponsored agencies. Any of these could be white-hat hackers who are seeking to find and patch vulnerabilities, or black-hat hackers who are finding vulnerabilities for destructive purposes. Some actors may be driven by financial incentives such as selling vulnerabilities to a third party, since there is a market for buying, selling, and renting vulnerabilities.

Whether vulnerability information should be made public prior to the release of a patch is the topic of debate. (Shazad et al, 2012) conducted a large-scale study and determined that, since 2008, the average number of public disclosures of vulnerabilities has been decreasing. In general, once an actor finds a vulnerability, there are three options. They could choose to keep the vulnerability secret in the hopes of exploiting it themselves, either selling it or using it for attacks. They could also choose to privately disclose it to the vendor so it can be patched before information is released to the public. Or

they could choose to fully disclose the vulnerability to the public without first informing the vendor, to put pressure on the vendor to create a patch. Publicly released vulnerabilities are recorded, among other places, in the Common Vulnerabilities and Exposures (CVE) database which is sponsored by the United States Computer Emergency Readiness Team (US-CERT) (Arora et al, 2004).

The time between vendor discovery and patch distribution varies. (Bilge and Dumitras, 2012) found 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 over two years before they were discovered. (Shazad et al, 2012) analyzed 46310 vulnerabilities discovered between 1988 and 2011 and concluded that patching has been gradually improving since 2005, and now 80% of vulnerabilities are patched before their disclosure dates due to the monetary incentives for reporting advertised by software vendors. However, once a patch is released by the software vendor, the user still needs to install it, and can be vulnerable in the meantime (Wilson et al, 2016) as we will discuss.

2.2. Cyberweapons over time

Most writers refer to cyberweapons as "single-use" or "perishable" weapons (Smeets, 2017). 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, whereas perishable means that cyberweapons become gradually ineffective even without use. Smeets suggests there is a limited period of time in which cyberweapons can cause harm, since there is an average of 312 days between a vulnerability being exploited and implementation of its patch.

(Schneier, 2000) referred to the transitory nature of cyberweapons as a "window of exposure." Figure 1 summarizes this framework which includes when the vulnerability was introduced into the code, when the vulnerability was discovered, when the vulnerability was exploited, when the vendor became aware of the vulnerability, when the vulnerability was disclosed to the public, and when the patch is released to the public respectively. These events will not necessarily happen in this order. Ultimately, a cyberweapon can be effective so long as it remains within the window of exposure, which exists between when the vulnerability was introduced and when it was patched. Though not shown, another important event after disclosure is the identification of signatures or anomalies associated with the exploits attacking the vulnerability, which serve in intrusion-detection systems to prevent attacks, particularly before the patch is available.

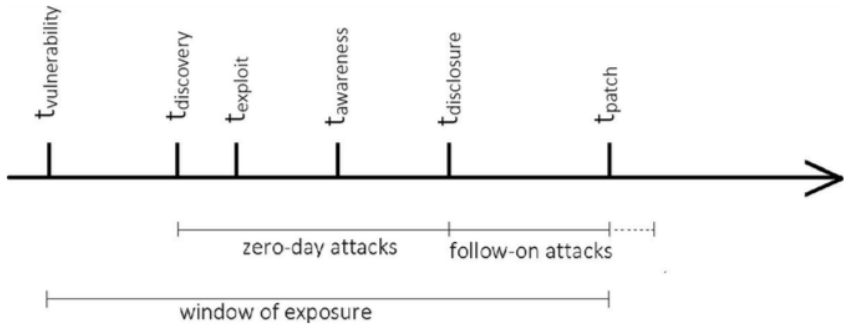

Figure 1. Lifecycle of a cyberweapon's effectiveness (Smeets, 2017).

Once a cyberweapon is used, its effectiveness decreases against other targets (Smeets, 2017). But this varies with the cyberweapon. A cyberweapon that stops an entire system will likely cause a short time between $t_{exploit}$ and $t_{awareness}$. But if the cyberweapon is for espionage it likely will not be found as quickly. Effectiveness also changes with the target pool size since more targets mean a

greater chance of discovering the attack quickly, although with many targets, there is a better chance that some users who slowly patch their system will still be vulnerable.

(Ablon and Bogart, 2017) proposed a theory of how long zero-day exploits can be kept 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 (patched) or private (unpatched) knowledge. Longevity refers to how long it is likely to remain undisclosed. The collision rate is the probability that someone else will discover it. (Herr and Schneier, 2017) refer to this as "rediscovery rate", and an example rediscovery is the Heartbleed vulnerability which was discovered by two companies within just days of each other. Cost is the financial cost of development of the cyberweapon. 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. They estimated that 25% of the exploits would not survive more than 1.5 years, and another 25% would not survive 9.5 years, with an average of 6.9 years. However, this study did not include the time for patching, during which exploits can still be successful after rediscovery.

(Moussouris and Siegel, 2015) proposed a system-dynamics model of the discovery, rediscovery, and stockpiling processes for vulnerabilities. Defenders want to find undiscovered vulnerabilities and patch them quickly. However, they must also assume that an attacker also knows of them and is stockpiling attacks, so patching the vulnerabilities will render the stockpiled vulnerability information useless, what they call "discovery correlation".

(Axelrod and Iliev, 2014) introduced a mathematical model for the best time to use a cyberweapon using variables for "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 because greater stakes mean 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 time before the weapon is 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.

(Hichkad and Bowie, 2012) analyzed the concept of secret weapons using some similar ideas. To evaluate the effectiveness of cyberweapons during military operations, the authors used the factors of 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. However, secret weapons are hard 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 causes a greater uncertainty as to whether or not the cyberweapon will actually work once it is deployed.

3. Factors affecting reusability

3.1. Malware signatures and anomalies

A malware signature is a distinctive byte sequence for a particular malware (Ask, 2006) that should rarely appear in other files. It often represents critical steps or data for the malware functionality and cannot be easily obfuscated. Ikarus Software GmbH is a company that writes malware signatures from the samples they receive. They have an automated process to look for those not having a pre-

existing signature. A human analyst then generates a signature by looking for syntactic signatures (Bonfante et al, 2008) in patterns expressed as "regular expressions" (Yu et al, 2006).

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

Once signatures or detailed anomaly descriptions are published for a particular malware, it becomes much less effective. However, it may still be reusable if contributory negligence occurs or the malware is well hidden.

3.2. Contributory negligence

Most system exploitations today are a result of unpatched operating systems or out-of-date antivirus software (Arbaugh et al., 2000). Users 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.

3.2.1 Unpatched systems

An example of negligence in patching was the Heartbleed vulnerability CVE-2014-0160 in the OpenSSL software library. The US-CERT (2014) released a technical alert Ta14-098A addressing this vulnerability on April 8, 2014, and announced 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). CVE-2014-6332, a vulnerability that was reported in Microsoft's Object Linking and Embedding (OLE) technology in Windows (Microsoft, 2015), 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 the user 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.

Another 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, -0144, -0145, -0146, and -0148 (Microsoft, 2017). One month later, an exploit known as EternalBlue was publicly released which took advantage of the SMB vulnerability that was patched by MS17-010. On May 12, 2017, the 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, a scan across 80 million IP addresses looking for the vulnerability exploited by EternalBlue identified 50,000 systems (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 attackers.

Many of these exploits succeeded due to failure of users to set their systems for automatic updates. Since the default setting on Windows machines is to have automatic updates selected, a significant number of Microsoft users are turning this function off or it is getting turned off by malware. (Kingsley-Hughes, 2017) suggests that many users choose to turn this function off because of the disruption and unreliability of the patches. 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.

Militaries or government organizations should be more diligent about applying patches quickly than home users. However, that does not necessarily mean that a significant window of opportunity cannot be exploited by an attacker. It only takes one administrator at one organization to fail to install a patch properly or on time to compromise an entire system.

3.2.2 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 May 4 and 5 in 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 version of a Windows application with a Trojan Horse. This allows the application to retain full functionality while containing embedded malware. A user who has the most basic protection from up-to-date free antivirus software could detect the modified application before it launches, yet substantial portions of the world's population do not have antivirus systems or keep them up-to-date.

Users are also defenseless against vulnerabilities that have been publicly disclosed but unpatched by the vendor. An example is the Hot Potato exploit which exploited three 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 has not been patched by Microsoft. Some 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 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 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 addressed 18 security bulletins, 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.

3.2.3 Social engineering

Humans can be a key component in allowing attackers to reuse malware multiple times because an attack can involve deceiving them repeatedly (Symantec, 2017). Despite continued warnings to users about opening links in suspicious emails, users continue to fall prey to phishing, which is a good way to deliver malware. Social engineering can gather information about a specific target to tailor phishing emails directly to them, and can allow an attacker to gain information to blackmail an individual, target their home network with the intention to pivot to their work network, or brute-force their credentials. CERT updated their definition of an insider threat in 2017 to include unintentional non-malicious actors (US-CERT, 2017).

3.3. Hiding and disguising malware

Attackers have several techniques to continue to gain access to a digital system once their malware has been emplaced. Most involve deception of some kind (Rowe and Rrushi, 2016). The secondary storage of digital devices is non-volatile, meaning data is stored until explicitly deleted. This is a common place for attackers to install malware on target systems. An effective technique of some malware is to insert itself in the operating system. 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 and remove it.

Malware can also hide in main memory if it is loaded from encrypted secondary storage. An example of a memory-resident attack was the Code Red worm which targeted Microsoft's IIS web service (Moore et al, 2002). "Fileless" malware can hide in the registry (McAfee, 2015). It gives the attacker persistence after the system reboots because the operating system loads the registry when the computer restarts. 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 it is still running (Gossett, 2015). Rootkits are another way to give attackers persistence on target systems, a subversion of the entire operating system. Rootkits can subvert normal OS behavior by intercepting and responding to questions about files and registry keys to hide their existence on machines. This allows the attacker to continue to reuse a single exploit well after the initial installation. Malware can also be highly persistent if it hides in hardware as by subverting the supply chain for a digital product (Inserra and Bucci, 2014). However, such malware is "brittle" in that it is hard to install and hard to replace if discovered.

One way to extend the window of malware is to vary its signatures through obfuscation techniques. In 2016 alone, 357 new malware variants were seen (Symantec, 2017). One technique is to encrypt the malware with a different key each time. Each time the malware infects a new target, the malware can be encrypted with a different key so that it looks different each time and its signatures will be different. The malware must be decrypted with a key in main memory to enable it to be used, and antivirus software can detect it there. Furthermore, if the decryptor machinery remains constant from one infection to the next, an antivirus signature can be written for the decryptor (You and Yim, 2010). Also, encrypted files are also easy to detect because they have very high entropy (data randomness), so they are suspicious even without a recognized signature.

Other techniques can be used to vary malware at each appearance (what is called "polymorphic" malware) such as dead-code insertion, register reassignment, subroutine reordering, instruction substitution, code transposition, 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. Attempts to write "metamorphic" malware, where the main body is never exposed in memory so a signature cannot be written against it, are difficult (You and Yim, 2010).

4. Strategic implications of a reusable cyberweapon

Cyberweapons differ from cybercriminal attacks in that they are often tailored to specific systems, and carry out a specific effect in a single strike much like conventional armaments. Due to their specificity, cyberweapons are not usually designed for reuse since they tend to have 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. This may cause the window of opportunity for reuse to close quickly for the original target and also for any other susceptible targets.

Nonetheless, reusable cyberweapons could be useful. To design one, it is helpful to refer to the military concept of survivability. That includes both allowing it to accomplish the mission prior to being detected and being able to accomplish more than one mission while keeping its window of opportunity open.

4.1. Aspects of survivability

(Ball, 2003) defines survivability of an aircraft as the probability it will survive its mission. He models survivability as one minus the product of susceptibility and vulnerability. Susceptibility is the inability to withstand a hostile environment; vulnerability is the inability to avoid or counter threats. Susceptibility can be estimated by simulating possible sequences of events and determining the likelihood that the aircraft will encounter an obstacle preventing a successful mission. Vulnerability depends on the predicted threats that the aircraft may face during the mission, and the threats' capabilities to hit critical components on the aircraft.

Applying the aircraft survivability model 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 to 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. Both kinds are valuable to cyberattackers.

A cyberweapon can be "killed" by vulnerability patching prior to mission accomplishment or by 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.

To reduce susceptibility, a cyberweapon must persist on the target system, meaning it must deceive the target and conceal its 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 its organizational routines (Gartzeke and Lindsay, 2015). To reduce vulnerability, a weapon can use redundancy, analogously to how an aircraft with multiple engines is less 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. Section 3.3 suggested some techniques to do this.

It requires time, effort, and resources to make a cyberweapon more resilient. When deciding where to target the software, there is a tradeoff between the level of difficulty and the resilience. For instance as mentioned, it is difficult to embed a cyberweapon in the hardware. The attacker would not only need to insert malicious code into the hardware design or implementation, but ensure that the device that was altered gets installed on the equipment that is later going to be targeted. This could however provide a persistent cyberweapon that would be difficult to locate. 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 from which to mount an attack, but they are volatile and the system will delete the malware when the system is powered off, therefore requiring that they be reloaded 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.

4.2. Designing a survivable cyberweapon

Cyberweapon design needs to vary with the mission objective and type of adversary. 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 survivability assessment including vulnerability and susceptibility. A diagram like Figure 2 can be helpful.

Wargaming can simulate the forces with which the cyberweapon may come in contact while on mission. They 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.

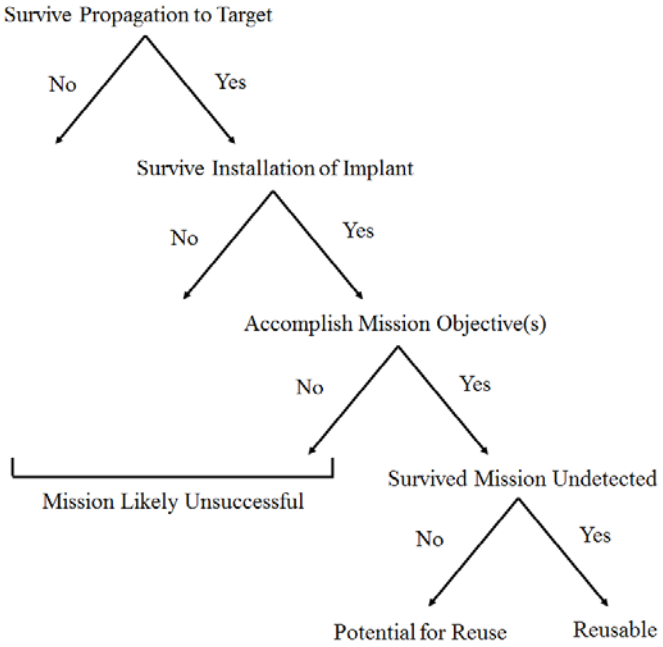

Figure 2. Tree diagram for static mission success, adapted from (Ball, 2003).

A cyberweapon should be continuously reevaluated to ensure it is still accomplishing its intended objectives and has not been detected (Figure 3). This is especially important when a cyberweapon can be reused against the same target or others. If the weapon is detected by the target, there is still an opportunity to continue the mission but it becomes riskier. 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 if there is a hope of future reuse. In addition, lengthy missions may also decide to terminate a cyber operation prior to the cyberweapon being detected.

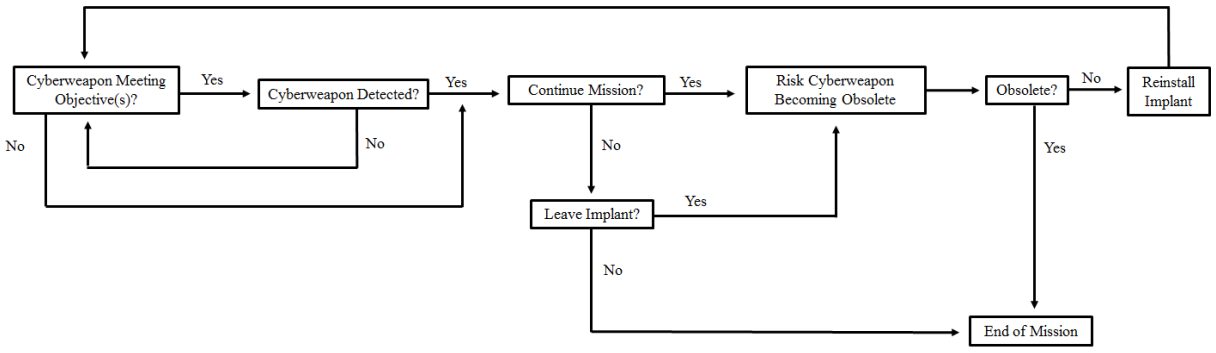

Figure 3. Flowchart for continuous mission reevaluation.

5. Policy recommendations

When the cyberweapon is under development, it must be decided whether it will be designed as a traditional overt single-use cyberweapon or a stealthy reusable cyberweapon since the design considerations are different. A case for when a reusable cyberweapon is indicated is when a target organization is developing a capability that the cyberattacker wants to prevent; then a cyberweapon could infiltrate the supply chain and disable equipment in a non-alarming manner that would appear to be a general machine malfunction and would persist for quite a while. Alternatively, a reusable cyberweapon could 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 for a while because they cannot find all the methods, and the attack could persist for some time until they do.

A reusable cyberweapon has a lower rate of being killed in use than a non-reusable cyberweapon. Therefore, a key issue is whether it has a higher chance of being killed in use or by obsolescence (its vulnerabilities being patched and its signatures recognized). Therefore, a special risk-assessment decision is 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 cannot immediately kill it. In principle, large stockpiles are not needed of highly reusable weapons.

Title 10 of the U.S. Code defines Department of Defense (DOD) and military operations, and Title 50 defines the operations conducted by the intelligence community. Military operations must publicly acknowledge the United States' role in the operation or it must be apparent as per Title 10 (Wall, 2011). Title 50 has no such requirement. 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 as to which authority would take precedence depending on whether there is a greater benefit to the intelligence community or to the military. For example, if the cyberweapon began as a covert action, 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 it and reduce the ability of it to be reused. However, 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.

Another issue is that 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, consider an attacker operating under Title 50 authority against one target and Title 10 authority against another; only when the effects are acknowledged by the attacker against the second target may the first target become aware that 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.

Title 18 of the U.S. code is also a consideration. Among other things, it prohibits cyber-sabotage and unauthorized access to computers, not necessarily in the United States, that relate to financial activities and interstate commerce. Many methods discussed here risk collateral damage of this kind.

International laws about cyberattacks are also increasing. The second edition of the proposed international law in the Tallinn Manual (Schmitt, 2017) goes further than U.S. law and suggests a number of restrictions on cyberattacks, many of which are now de facto law (Rowe, 2018), that could apply to reusable attacks. Particularly important is Rule 99, "Civilian objects should not be made the object of cyber attacks. Cyber infrastructure may only be made the object of attack if it qualifies as a military objective." It is hard to design a cyberattack to avoid civilian objects and cyber

infrastructure, as discussed in (Rowe, 2017), and since it is hard to get feedback from a cyberattack, a reusable one could easily continue damage to civilians unknown to the attacker.

6. Conclusions

Cyberweapons can be reused in some circumstances. This will add to their value and provide more benefit for the development and implementation costs. However, there are many difficult obstacles to reuse, some tactical and legal, but the most serious are technological. Military operations need high reliability, so they cannot depend on negligence for cyberattacks, the primary facilitator of most cybercriminal cyberattacks. They must use concealment and other forms of deception and these methods will need to be carefully designed. On the other hand, well-concealed cyberattacks may have little tactical or strategic effect on an adversary. Even a well-designed reusable cyberattack may suddenly become obsolete because of discovery or software patches. Thus, reusable cyberattacks will always be difficult to achieve and military planners need to be aware of this.

Acknowledgements

Statements in this paper are the opinion of the authors and do not represent the U.S. Government. This work was supported in part by the U.S. National Science Foundation under the Secure and Trustworthy Cyberspace program.

References

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

Arbaugh, W., Fithen, W., and 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., and 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., and 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., and 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

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

Gartzke, E., and 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

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

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

Hichkad, R., and Bowie, C. (2012). Secret weapons and 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., and 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

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

Moore, D., Shannon, C., and 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., and 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/

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

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. (2017). Challenges of civilian distinction in cyberwarfare. Chapter 3 in M. Taddeo and L. Glorioso (Eds.), *Ethics and Policies for Cyber Warfare: A NATO Cooperative Cyber Defense Centre of Excellence Initiative*, Philosophical Studies Series Vol. 124, Springer, pp. 33-48.

Rowe, N. (Winter 2018). A taxonomy of norms in cyberconflict for government policymakers. *Journal of Information Warfare*, 17(1).

Rowe, N., and Rrushi, J. (2016). *Introduction to cyberdeception*. Springer, New York.

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

Schmitt, M. (ed.) (2017). *The Tallinn Manual on the international law applicable to cyber operations*, Cambridge University Press, Cambridge, UK.

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

Shazad, M., Shafiq, M., and 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., and 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

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

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

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, and 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., and Herr, T. (2016). Bugs in the system. Open Technology Institute. Retrieved from https://www.newamerica.org/oti/policy-papers/bugs-system/

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

You, I. and 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., and 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