

Amid the VIPERS
Establishing Malware's Position Within the Information Ecosystem

A Senior Project

presented to

the Faculty of the Computer Science

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science

by

Shawn Louis Everett Robertson

March, 2011

© 2011 Shawn Louis Everett Robertson

Table of Contents

1.0 Introduction: Malware in the World Today	3
1.1 <i>Definition of Terms</i>	3
1.2 <i>Types of Malware</i>	5
1.3 <i>Historical Examples</i>	7
1.4.0 <i>Analyzing the Attackers</i>	9
1.4.1.0 <i>Objectives</i>	9
1.4.1.1 <i>Personal Power</i>	10
1.4.1.2 <i>Profit</i>	10
1.4.1.3 <i>Political/Military Goals</i>	12
1.4.1.4 <i>Other Goals</i>	15
1.4.2 <i>Capabilities</i>	16
1.4.3 <i>Behaviors</i>	16
1.5 <i>Why Malware is so Prevalent</i>	17
1.6 <i>Why Malware is Hard to Defend Against</i>	20
1.7.0 <i>To Build or Not to Build</i>	24
1.7.1 <i>Reasons to Create Malware for Testing</i>	24
1.7.2 <i>Reasons Not to Create Malware (for Testing or Otherwise)</i>	26
2.0 Background: “Good” Pieces of Malware	29
2.1 <i>Current Standard: CVSS</i>	29
2.2.0 <i>Alternate Standard: VIPERS Classification</i>	32
2.2.1 <i>Virulence</i>	33
2.2.2 <i>Infiltration</i>	33
2.2.3 <i>Payload</i>	34
2.2.4 <i>Exclusivity</i>	34
2.2.5 <i>Remoteness</i>	35
2.2.6 <i>Strength</i>	36
2.3.0 <i>Justifying VIPERS</i>	37
2.3.1 <i>From Biology to Binary</i>	37
2.3.2 <i>The Importance of Subtlety</i>	39
2.4 <i>Enter the Plaguebringer</i>	42
2.5 <i>A Stumbling Block</i>	43
3.0 Implementation: A VIPER Examined	45
3.1 <i>Virulence: Cross Site Scripting with Flash</i>	45
3.2 <i>Infiltration: Disguising Malicious JavaScript</i>	47
3.3 <i>Payload: Heap Spraying and Shellcode</i>	49
4.0 Conclusion: The Internet Used To Be A Nice Neighborhood	51
Works Cited	52

1.0 Introduction: Malware in the World Today

Before identifying what makes a “good” piece of malware, it is important to identify the roll that malware performs in the world today. First, some basic terminology will be defined in order to ensure complete understanding of the subsequent material. The next step to examining malware is to understand the several different types and what their particular strengths and weaknesses are.

Additionally, by examining multiple historical examples of malware, one can determine strategies that have worked in the past. After examining the technological artifact itself, one can then investigate who is creating malware and why, enabling them to understand the motivations behind this virtual epidemic. Next will be an examination of why malware is so prevalent in the world today, as well as several reasons why it is so hard to defend against. Finally, this section will attempt to compare and contrast reasons for either creating or not creating malware for testing purposes.

1.1 Definition of Terms

Attacker: A malicious user who is attempting to compromise a system's confidentiality, integrity, availability, or some combination thereof.

Confidentiality: The system only gives information to individuals or processes who are authorized to access that information.

Integrity: The system functions as intended *and nothing else* (Nico).

Availability: The system can be utilized by authorized users when they desire to use it.

Malware: A series of malicious instructions executed in order to exploit either a logic error or a coding error (Orton).

Payload: The portion of a piece of malware that actually performs a function. While the rest of the code is generally dedicated to infiltrating a system or obfuscating the presence of the malware, the payload is what actually accomplishes goals like opening a backdoor or stealing information (Heidari).

Code Error: An error in the way that the software was written while implementing a correct piece of logic. Code errors are the most frequently encountered errors but they are also the easiest to fix (Orton).

Logic Error: An error inherent in the underlying functionality of a piece of software. These errors are rarer than code errors, but they are also much more difficult to discover and fix (Orton). For example, attackers exploit the way that web servers respond to requests by conducting Denial of Service attacks.

Denial of Service (DOS): Issuing repeated resource-intensive commands to a computer or server in order to deny its availability to normal users (Heidari).

Distributed Denial of Service (DDOS): Utilizing a network of computers to commit a massive DOS attack. Examples include the Low-Orbital-Ion-Cannon, a piece of software which can be operated in “hivemind” mode; this allows all connected computers to function as a single controlled network to attempt a DDOS attack (Johnson). DDOS attacks are harder to defend against than traditional DOS attacks as there is no single attacker to block all communications from.

Zero Day Exploit: An exploit which is unknown to anyone but the attacker. The name is due to the exploit existing on the 0th day of the vendor's response cycle. These exploits are jealously guarded and sold for a premium on the black market (Miller).

Rooted: A machine that has been infected by a rootkit. A rootkit is a piece of malware that allows privileged access to the attacker while also masking its presence from the user. They allow undetected control over a system and can be particularly insidious to remove (McAfee).

Zombie: A remotely controlled system which is connected to the Internet. This is typically used to launch attacks that are more difficult to trace back to the original attacker, or as components of DDOS attacks (Heidari).

Zombie Army: A collection of individual zombie system that is controlled from a central entity. Also known as a botnet (McAfee).

Exfiltrate: Secretly getting data out of a system through either unauthorized media or covert channels

(Rowe, *Counterplanning*). This is the opposite of infiltration, which is to secretly access a system.

1.2 Types of Malware

While the general public generally refers to all malware as a virus, there are actually several distinct classifications of malware. One of the fundamental differences which serves to characterize different forms of malware is whether or not it requires a host program in order to execute. A piece of code which requires a host program can be further differentiated by whether or not the hostile code was intended to be a part of that program. If the code has been injected by the original author, it is most likely to be either a trap door, a logic bomb, or a Trojan horse. A trap door, also known as a back door, is used to allow the original author of the code access to the program while bypassing the usual security features. These can be left for valid reasons, such as allowing for updates, testing, or access in the event of errors. However, these can also be exploited in order to achieve complete control of a system without the end-user's permission (Heidari). While a trap door can have valid reasons for its existence, a developer would have to do substantial amounts of backpedaling in order to justify including a logic bomb in a piece of software. A logic bomb is malicious code which is placed within a piece of software and activates once a set of predefined conditions is met (Heidari). It can be something as simple as the system connecting to the Internet for the first time, or it can be a complex precondition which requires a specific date, time, and IP address. Once the precondition is met, the malicious code is executed with the legitimate software's set of permissions. In order to be truly classified as an exploit, the logic bomb generally executes some destructive or security-compromising action while informing the original author that a victim is ready for exploitation (Heidari). The final preconfigured exploit is known as a Trojan. A Trojan horse, more commonly referred to as a Trojan, is named after the famous military gambit from Greek mythology. It masquerades as a useful program while actually containing additional exploit functionality. The difference between a Trojan and a logic bomb is the fact that a Trojan

generally does not have any preconditions to activating its payload other than first execution (Heidari).

The final form of host dependent malware, the ubiquitous virus, is different from the previous classifications by two important factors. First, the virus code is generally not placed in the host program by the original host author, but injected by an attacker at a later time. Additionally, viruses are able to replicate on their own, which makes them particularly dangerous (McAfee). A computer virus mimics a biological virus in the manner that it attaches to an otherwise healthy program and injects its own code into the existing program (Wassenaar, Blaser). A typical virus will add its exploit code to the end of a program as well as two separate jump instructions (Heidari). The first jump will be placed at the start of the program and will cause it to execute the exploit code before performing its normal duties. The second jump instruction will hop back to the normal program, functioning as if nothing untoward had happened (Heidari). Viruses are characterized by what they infect and how they propagate through an infected system (Heidari). An important fact to consider is that all viruses require some interaction from the user in order to begin their appointed mission (McAfee). This is why computer viruses have been compared to sexually transmitted diseases, as they require specific behavioral practices in order to spread (Wassenaar, Blaser).

The next type of malware is unique in the fact that it requires neither a host program nor any form of specific user interaction in order to achieve infection (McAfee). The innocuously named worm has been likened to socially transmitted diseases such as influenza, as it spreads to and infects all vulnerable systems (Wassenaar, Blaser). A worm is a fully self-contained program which utilizes specific security exploits and propagation techniques in order to achieve the maximum infection across the world (Heidari). While the actual payload of a worm may only be activated when it satisfies a certain set of preconditions, the typical worm will infect everything possible. As they do not require user interaction to achieve infection and utilize each infected machine as a new distribution point, the spread of worms is typically exponential and can pose “serious risks to the internet infrastructure as a

whole” (Heidari). Worms generally exist along a continuum with speed on one end and stealth on the other. A worm which spreads as fast as possible will conduct many scans of potential victims and cause network communications to appear anomalous when viewed over time (Weaver, Paxson, Staniford, Cunningham). The stealthier worms rely on passive propagation, only passing themselves along to new victims by embedding themselves within normal communications (Weaver, Paxson, Staniford, Cunningham). These stealthy worms cause infected machines to appear normal for longer periods of time, delaying any potential reaction to the malware. Additionally, worms can be distributed with or without payloads. While it may seem useless to have a worm which reaches half the world without actually doing anything, these apparently harmless worms are often used to test new infection vectors (Knapp, Boulton).

Now that the different types of malware have been examined, it is possible to evaluate a few historical examples to learn more about them.

1.3 Historical Examples of Malware

On November 29th, 2010 Iran's government confirmed that its nuclear program had been damaged by malware known as the Stuxnet worm (Nicoll, Delaney). The worm was a self-replicating piece of software which utilized intelligence of the particular industrial controller used by Iran's Natanz plant to locate its target. This act of premeditated sabotage was designed to delay Iran's attempt at acquiring enriched Uranium and appears to have been a joint effort by Israel and the United States (Gross). The worm was incredibly technically sophisticated, it contained four zero day exploits to help it accomplish its goal of sabotage. Stuxnet spread throughout the globe, infecting Windows 64-bit machines and checking each system it infected to see if it had found its true target. The worm spread to any USB thumb drives which were plugged into an infected system enabling it to infiltrate networks with no connection to the public Internet. Relying on lax thumb-drive discipline and poor IT security in

Iran, the worm was then transferred from infected personal networks to the closed network at the Natanz plant (Gross). Once the worm verified that it had reached its target, it rooted the Programmable Logic Controller manufactured by the Siemens corporation and used at the plant (Nicoll, Delaney). The worm then rapidly oscillated the operating frequency of the gas-enrichment centrifuges while reporting nothing amiss to technicians monitoring the system (Gross). This caused a rapid deterioration of the centrifuges, destroying an unknown number utilized for enriching Uranium before finally being discovered (Gross). Outgoing Israeli Intelligence Chief Meir Dagan commented on the setback to Iran's nuclear program, claiming "Iran won't reach its nuclear capabilities before 2015" (Nicoll, Delaney). IT Security Analyst Ralph Langer claimed that the cyber attack was, "...nearly as effective as a military strike, but even better since there are no fatalities and no full-blown war" (Nicoll, Delaney). This piece of malware serves to demonstrate the ability for software to affect items in the physical world as well as herald an oncoming age of true cyber warfare.

Other examples can be used to examine particular methods of manipulating vulnerable users. The ILOVEYOU worm of 2000 originated in the Philippines and employed social engineering to spread around the World. The worm arrived as an attachment in an email with the subject line "ILOVEYOU" (Wassenaar, Blaser); this enticed users to open the attachment, wanting to see someone professing their love. In reality, the attachment contained no love letter and instead spread by sending itself to individuals on the infected system's email list, purporting to be from the victim (Rikstep). The cycle then repeated itself as acquaintances of the newly infected victim let their curiosity get the better of them and opened the attachment. This particular piece of malware shows how effective social engineering can be, as the worm employed both an intriguing message and appeared to be from someone that the user knew (Rikstep).

The SQLslammer worm stands out due to the incredible speed of its infection. Indeed, the worm ended up infecting over 90% of all vulnerable hosts within 10 minutes of its initial release (Heidari).

SQLslammer was an incredibly small worm, it fit into a single 404 byte UDP packet; this allowed for the worm to simply send itself out to a potential victim and not have to wait for a response before attacking (Heidari). The worm's random number generator, utilized to determine where to attack next, ended up having a bug which left a large number of potential victims unscanned and unattacked (Heidari). The worm serves as an important example of how knowledge of technical processes allows for the creation of a true terror of the Internet. It also demonstrates the importance of testing all code for bugs, including malware. Had the creator fully tested the random number generator, the attack could have been even more devastating.

1.4.0 Analyzing the Attackers

According to Scott Orton, Raytheon's Director of System Integrity Programs, attackers are composed of three separate attributes: objectives, capabilities, and behaviors. Objectives are what the attacker hopes to accomplish through their malfeasance and can be categorized into the following: Personal Power, Profit, Political/Military, and Other. The capabilities of attackers are defined by what they can and cannot do to a system. Finally, the behaviors of attackers are human psychological tendencies that Mr. Orton claims are largely the same between attackers and remain mostly constant for each individual over time.

1.4.1.0 Objectives

By analyzing an attacker's objectives, one can finally answer what many users must wonder after they have been attacked by malware: “why did this happen to me?” When viewing an attack as a several stage process, it is easier to realize some of the collateral damage that occurs when otherwise inconsequential civilians become infected with malware. Indeed, it appears that while many civilians are impacted by malware, it is often in pursuit of some greater goal.

1.4.1.1 Personal Power: The desire for personal power is a motivation that can be easily understood by an outsider. Whether it is an individual seeking to improve their station in life through illicit means, or to acquire power over other individuals, malware can be utilized to further these goals. As an example, if an attacker compromised a powerful individual's system and discovered incriminating information within, they could then extort that individual to do what they wanted. In a work environment, the attacker could force the victim to give them a raise or other benefits that would advance their station in life.

Another potential way to increase personal power is by refining knowledge, thereby making future attacks more likely to succeed. If an attacker sends out a test attack to verify that a particular exploit exists and functions as intended, this can be seen as gathering power in order to make a future attack successful. Additionally, an attacker can compromise a system and create a zombie in order to utilize it in future attacks. Zombies are useful for numerous activities such as bypassing spam filters, attempting DDOS attacks (Heidari), attacking allegedly secure systems from unexpected origins, and making proper attribution for future attacks difficult (Dhamankar, Dausin, Eisenbarth, and King). As such, increasing personal power through malware attacks appears to be a valid, if unethical, motivation.

1.4.1.2 Profit: Recent years have caused a shift in potential motivations as the increased capabilities of malware have allowed for attackers to generate a monetary profit from their creations. There are two major categories of achieving profit from malware, one is directly malicious while the other is only questionably so. One utilizes knowledge in the form of a cyber attack in order to attain a direct monetary benefit; the other acts as an enabler for other individuals who may or may not have malicious intentions.

Those attackers who choose to engage in directly malicious actions have a plethora of

opportunities available to make a profit. The ability to extract a profit from compromised systems has allowed for malware to finally be considered a successful biological parasite (Furnell, Ward, Malware). By compromising a system, the attacker gains several assets. At the most basic level, they gain access to the data stored on the system. If that data is incriminating, they could use it to blackmail the owner of that system. If the data is particularly sensitive, the attacker could instead sell it to a different party in order to make a quick profit. This appears to be quite common because in 2002 a survey of 498 members of various organizations reported that 40% claimed to have “received confidential information belonging to other companies via the Internet” (Knapp, Boulton). In addition to exfiltrating sensitive information, competitors could pay for attackers to introduce errors into their opponent's products, or perform other acts of sabotage.

Another potential for profit treats the compromised machine as a resource to utilize for future attacks. By turning the infected machine into a zombie, the attacker gains another member in a virtual army. After a zombie army of suitable size has been assembled, attackers can attempt DDOS attacks to bring down servers of online businesses. Online gambling sites appear to be common targets, with some reporting demands from Russian crime syndicates of \$50,000 in order to call off the attack (Furnell, Ward, Malware). Additionally, attackers can instead sell or rent out their zombie armies to spammers or other organized crime syndicates. Attackers can rent out the processing power and bandwidth of their zombie army for as little as \$100 an hour (Furnell, Ward, Malware).

While all of the above actions are unquestionably malicious, the other option available for profit resides in a gray area. Instead of utilizing the exploit that the attacker has developed by creating a piece of malware and directly attacking other systems, the attacker can simply sell the exploit itself. This raises a whole issue of other problems, as they must somehow demonstrate to potential buyers that the exploit works without giving away any information that would allow the buyers to recreate the exploit on their own (Miller). Additionally, at some point the attacker must provide their exploit to the buyer

and the buyer must provide the money to the attacker; this causes a whole new set of problems.

Without the presence of a trusted third party who will hold both assets in escrow, either the attacker must provide the exploit and trust that the buyer will pay, or the buyer must first provide the money and trust the attacker to provide the exploit (Miller). Currently there is a distinct lack of a trusted third party for exploits, therefore it is almost impossible to guarantee simultaneous delivery of goods (Miller).

The reason that the sale of exploits resides in a gray area of maliciousness is due to the fact that the original attacker has no control over what is done with the exploit after selling it. While it could be sold to a legitimate source such as several penetration testing firms or a government agency (Miller), it can also be sold to criminal organizations. Even the supposedly legitimate buyers can utilize the exploits for dubiously ethical purposes or sell it to someone else. Therefore profiting from the sale of exploits is unclear regarding its maliciousness.

1.4.1.3 Political/Military Goals: There are four major forms of attacks which are motivated by political or military goals; these appear to be differentiated by whether they are primarily conducted by states or nonstate actors. Whereas states are typically concerned with cyber war and economic espionage, nonstate actors appear to concern themselves with cybercrime and cyber terrorism (Nye). By examining each form individually, it is possible to better understand the whole.

Cyber warfare is defined as “...cyber-actions that have effects outside cyberspace that amplify or are equivalent to physical violence...” (Nye). There are few public examples of cyber warfare in practice under this definition, yet the Stuxnet worm serves as a perfect case. As the interconnectedness provided by the Internet increases worldwide, so do potential targets for cyber warfare (Nugent, Raisinghani). Power grids, water sanitation plants, government communication channels, and military security stations all provide tempting targets for cyber war planners. However, there are two major arguments against attempting cyber warfare in an attempt to destroy an opponent's infrastructure. First,

“[evidence] seems to suggest that it is unlikely that cyber terrorists or other war planners could cause meaningful damage to critical infrastructures through cyber warfare tactics” (Owen). However, this can be remedied by instead attacking several non-critical services in order to both plant a foothold for future attacks and to erode public confidence in the system itself. After all, “decreased use of a system due to lack of confidence is in many ways the same as if a part of the system had been destroyed in a huge single attack” (Owen). The total cost of malicious software in 2003 alone was approximately \$13 billion; attacking nation states can take note and conduct cyber warfare with the goal of economic damage rather than the destruction of critical infrastructure (Owen). The second problem inherent in states building an arsenal of exploits is that maximum effectiveness will be achieved by immediately releasing the exploit. Each day the exploit isn't used, the chances increase that someone else will discover the exploit and bring attention to it (Miller). Additionally, the exploit could be fixed in a seemingly unrelated patch, causing the stockpiled exploit to become useless against updated systems (Miller). The only way to effectively mitigate this fact is to attack as soon as an exploit is discovered. Machiavelli maintains that attacking as soon as a state has the capability is a just action by claiming that, “...war is not to be avoided, it is only to be put off to the advantage of others...” (Machiavelli). However, an immediate strike with cyber weapons may be less effective than waiting for a later date as individual cyber attacks are generally highly effective only once; this is because survivors will be able to adapt to the methods utilized for that specific attack (Rowe, *Ethics*). This leaves war planners in a difficult position as they must gamble for the correct time to unleash their new weapons.

Economic Espionage is the other form of cyber attack which states engage in. As first world nations find their economies increasingly dependent upon the Internet and other information networks, they become more vulnerable to outside influence (Nye). It appears that currently the private sector is actually the primary target of information warfare; this is partially due to the difficulty inherent in any military retaliation (Knapp, Boulton). While some individuals might believe that it is only developing

nations who are conducting economic espionage in order to gain an advantage, data simply does not support this. Indeed, the former director of French Intelligence publicly admitted that French Intelligence acquires confidential information about competitors and forwards this data on to French companies (Knapp, Boulton). Additionally, the CIA released a warning to US companies outsourcing the fix for the Y2K bug that offshore companies were installing backdoors to allow their host governments remote access to the updated software (Nugent, Raisinghani). These are just two proven events of economic espionage, but one can extrapolate that other nations with information warfare capabilities are also engaging in these actions.

Actions commonly undertaken by nonstate actors are cybercrime and cyber terrorism. While cybercrime is politically or militarily motivated mostly in the choice of targets, cyber terrorism is entirely different. Cyber terrorism is defined as “the premeditated, politically motivated attacks against information, computer-systems, computer programs, and data which result in violence against non-combatant targets by sub-national groups or clandestine agents” (Curran, Concannon, McKeever). Many of the factors which lead to the prevalence of malware and cyber attacks also benefit cyber terrorism and will be examined in a future section. Academics believe that the combination of cyber terrorism and conventional physical terrorism is the most effective use of cyber terrorism (Curran, Concannon, McKeever). Potential combinations include the delay of crisis response teams following a traditional attack, which would further exacerbate the damage and chaos caused. Another option is to utilize cyber terrorism in order to allow for traditional terrorism to strike at otherwise protected targets; this can be by either drawing response teams away from the target or allowing individuals to infiltrate past existing security measures. Historical cyber terrorism events are not as sophisticated and have generally been classified as low-level information warfare because they mostly consist of website defacing, spamming to disrupt communications, and intercepting communications in order to weaken organizations (Curran, Concannon, McKeever). The European Union has responded to the possibility

of more severe cyber terrorist attacks by mandating that “attacks through interference with an information system” be punishable as a terrorist action if the goal is “seriously altering or destroying the political, economic, or social structures” (Curran, Concannon, McKeever). This growing awareness of cyber terrorism as a valid form of attack will doubtlessly increase the number of attacks in upcoming years.

1.4.1.4 *Other Goals*: The remaining motivations can be broken down according to human emotion. These can be curiosity, pride, rage, and others. Steven Furnell and Jeremy Ward claim that “Malware writers have never been known for their public-spirited activity, so if they are electing not to directly harm our systems there must be something in it for them”(Furnell, Ward, *Malware*). While they mostly draw attention to the profit available for the creators of malware, these benefits can also be psychological. While the original hackers were seen as mostly curious individuals, one can imagine other psychological benefits individuals can gain from conducting cyber attacks. The original pieces of malware appear to have been motivated by a combination of curiosity as to how systems work as well as ego. This is demonstrated by the original payloads being rather flashy, with the objective of being seen and gaining recognition for the attacker(Furnell, Ward, *Malware*). As there were few prior examples to compare to, relatively simple attacks were enough to gain recognition(Furnell, Ward, *Malware*). However, the increase in sophistication has resulted in attackers creating more devious exploits and obsessively guarding their methods in order to gain respect from other attackers (Rowe, Duong, Cutsy). Indeed, it seems that the number of machines exploited is a measure of the skill of a hacker within the community (Rowe, Duong, Cutsy).

In addition, there is the chance that attackers are responding out of rage or vengeance for a perceived slight. The organization known as Anonymous is famous for declaring crusades against particular groups due to both perceived slights and “for the lulz” (Davies). This has caused some

attackers to be labeled as “socially maladapted manchildren” due to responding to online insults with cyber attacks (Unknown). While this does not paint a flattering image of attackers, it is easy to imagine certain immature individuals utilizing their knowledge to exploit the machines of those who have wronged them.

1.4.2 Capabilities

Capabilities are what the attacker is able to do to a system. Some of these capabilities come from failure analysis tools, where the attacker experiments with access controls in unique or novel ways and finds an edge case which will allow access to the system (Orton). Attackers seek to discover either Code Errors or Logic Errors, with the latter being more valuable (Orton). It is important to note that attackers and defenders are locked in a virtual arms race which results in capabilities staying in lockstep with advances in security (Orton). However, traditional access controls have been studied for a substantial amount of time, which results in major innovation being quite rare (Rowe, *Designing*). Capabilities vary widely between attackers, ranging from “determined adversaries with serious motivations” to simple “script kiddies” (Rowe, *Counterplanning*). In spite of this, the presence of a database of over 6,000 sites which contains only “a part of the better hacker tools” allows even unsophisticated attackers to quickly learn more (Knapp, Boulton). This increase in knowledge will allow for an increase in capabilities, which is something all defenders should worry about.

1.4.3 Behaviors

Behaviors are the psychologically motivated traits which are present in attackers. While attackers are by no means a homogenous group with uniform characteristics, one can still analyze traits which appear prevalent within the more successful members of the group. Attackers avoid monitoring due to the fact that they thrive on subtlety and concealment in order to achieve maximum effectiveness

(Rowe, Duong, Cutsy). Due to the access of training tools, attackers can also be highly skilled (Orton). They also tend to be methodical, making repeated attempts at a first compromise (Orton). This is exemplified by an embittered Australian employee who took 45 attempts to release 1 million liters of raw sewage into the river and coastal waters of Queensland (Curran, Concannon, McKeever). At the same time, once attackers have successfully accomplished an attack, they tend to create a process which they follow for future attacks (Orton). These processes are generally automated in the form of attack scripts; however these automated scripts often lack sufficient conditional statements to react correctly to any unexpected deviations from the plan (Rowe, Duong, Cutsy). Additionally, attackers generally trust in their senses and what systems report to them (Rowe, *Counterplanning*). Attackers generally assume that if they encounter a system that resembles something they have seen before, it will be the same as what they have encountered previously (Orton). This can be exploited by defenders, who can utilize deception against attackers in order to mislead and prevent them from accomplishing their objectives (Rowe, *Counterplanning*). As psychological traits generally remain constant within individuals while capabilities change (Orton), defenders should target specific behaviors rather than specific capabilities in order to achieve maximum return for the least amount of effort.

1.5 Why Malware is so Prevalent

Several factors combine to lead to the prevalence of malware in the world today. These run the gamut from lax laws preventing it to the difficulty inherent in correctly locating an attacker or responding in real time. One of the primary reasons for the abundance of malware is the extremely low cost of entry inherent in a cyber attack (Nye). This low cost is due to a variety of reasons. First, the cost of the delivery mechanism, the Internet, is much lower than the cost associated with a traditional attack such as guns and individuals who will use them. Additionally, some attacks require less technical knowledge than in the early days of cyber weaponry; the availability of downloadable graphical user

interface tools directly led to the removal of many barriers for entry (Knapp, Boulton). As an example, guides exist that allow unsophisticated users to create electromagnetic pulse bombs for as little as \$400 in parts (Knapp, Boulton). While not technically malware, these devices allow attackers to target systems in a similar manner by denying availability for legitimate users. Another source of cost reduction is in the fact that the Internet allows for attackers to target systems anywhere in the world without paying transportation costs traditionally associated with such gambits. Furthermore, the presence of approximately 1.7 billion individuals on the Internet (Nye) allows for an attacker to choose a valid target within a wide array of options without paying an additional cost (Nugent, Raisinghani). The costs inherent in undertaking a cyber attack are primarily limited to the development and initial seeding of the attack, as both the Internet and infected servers are available to all attackers with skill and patience (Nugent, Raisinghani).

Another factor leading to the prevalence of malware is the fact that correct attribution is nearly impossible in cyberspace (Rowe, *Ethics*). While several methods exist to attempt to trace a signal with varying degrees of legality (Arnes), even these can be bypassed by clever attackers. Many sites which maintain information about IP addresses, DNS addresses, and user information lack any form of editorial control (Arnes); clever attackers can seed incorrect information about their location in order to throw off searches. Indeed, it is possible to perform a “false-flag attack” in which an attacker seeds information and appears to be from a different nation in order to throw suspicions to an innocent group (Rowe, *Ethics*). This is similar to hiring mercenaries to wear the uniforms of another nation and is prohibited in normal warfare but not in cyberspace (Rowe, *Ethics*). Clever attackers can utilize other methods of obfuscation such as TOR networks and anonymous IP addresses to further hinder tracing (Arnes). Additionally, attacks made through zombie systems will allow for yet another degree of separation between the attacker and the final target (Heidari). All of these systems combine to ensure that discovering who actually made an attack is exceedingly difficult. An attacker motivated by pride

may sign their work, but there is still no guarantee that the signature is actually from the author and not someone masquerading as them. This anonymity has allowed attackers to not fear potential repercussions from their actions, causing Machiavelli's advice to only attack when you will completely crush the enemy and prevent any chance of retaliation to hold less importance (Machiavelli). This lack of attribution has been analyzed by academics as well as fiction writers (Suarez) and remains yet another reason that malware is so prevalent.

The international community lacked any form of unified cybercrime laws for quite some time, which further motivated the creation of malware. The Convention on Cybercrime has sought to establish these laws through a treaty between numerous nations, but it has drawn its own shares of detractors (Kierkegaard). Ratified in 2006 by the US, the treaty makes several actions illegal, including “to create, possess, or acquire any computer program designed to crack or disrupt systems illegally” (Kierkegaard). It is important to note that writing malware for the “testing or protection of a computer system” would not impose criminal liability, but it is questionable how this will be upheld in court (Kierkegaard). While it may seem that the ratification of this treaty would drive the production of malware down, a vast majority of states have not ratified the treaty, citing the fact that the process excluded legal experts and human rights advocates (Kierkegaard). Furthermore, the treaty is under scrutiny by several states who have expressed a desire to impose procedural safeguards for limiting surveillance powers (Kierkegaard). This will likely result in many attackers ignoring the treaty as it may be changed radically before being adopted worldwide. Finally, even if the treaty were to be completely adopted, the lack of guaranteed attribution would most likely cause only unskilled attackers to fear the laws. This could result in the only malware authors being sophisticated users who take lengthy steps to prevent their identities from being established, while making sure to cause other users to appear to be the ones responsible for attacks they commit.

Another factor contributing to malware's preeminent presence in the online ecosystem is the fact

that it is impossible to truly prevent attacks, one can only delay or mitigate them (Schneidewind). Professor Neil C. Rowe of the Naval Postgraduate Institute maintains that “it is unfeasible for a military organization... to think their systems cannot be compromised by determined adversaries with serious motivations” (Rowe, *Counterplanning*). As such, academics argue that if attacks cannot be prevented one should instead focus on mitigating them (Rowe, *Designing*). Malware writers acknowledge this fact, but seek to be efficient with their time and generally target the machines that are easiest to compromise unless motivated to target a specific system (Orton). If an attacker is willing to invest the time and resources into compromising a system, they generally will be able to accomplish this goal (Orton). Simply looking at examples of polymorphic shellcode online demonstrates the effort attackers are willing to invest in compromising specific systems (Offensive Security). Additional reasons for the difficulty of preventing attacks will follow in the next section.

1.6 Why Malware is Hard to Defend Against

Part of the reason that attacks are so difficult to prevent is due to the fact that users of commercial software cannot do much to protect themselves. As a majority of the infrastructure of the Internet is controlled by private sector entities such as the ISPs or ICANN (Schneidewind), there are few countermeasures available for end users to employ. Even were an organization to utilize the most up-to-date security protocols, a single unpatched flaw in a commercial operating system or frequently used program could render them all moot (Schneidewind). Malware writers have realized this fact and adjusted their tactics accordingly, exploiting client-side vulnerabilities in commercial software as the primary infection vector against systems with internet access (Dhamankar, Dausin, Eisenbarth, and King). These infected machines are then used to spread the malware to other machines within the network, which were incorrectly thought to be protected from outside influence (Dhamankar, Dausin, Eisenbarth, and King). Once a company has discovered a vulnerability, it typically takes twice as long

for a patch to be applied to client-side software than an operating system (Dhamankar, Dausin, Eisenbarth, and King). Some vulnerabilities have remained unpatched for up to two years after their initial discovery (Dhamankar, Dausin, Eisenbarth, and King).

This time delay for patching brings up yet another difficulty in defending against malware and cyber attacks, the inability to respond in real-time. According to IEEE, the definition of real-time is, “Pertaining to a system or mode of operation in which computation is performed during the actual time that an external process occurs, in order that the computation results can be used to control, monitor, or respond in a timely manner to the external process” (Schneidewind). In order to respond to a malware attack, several steps must be accomplished. First, the attack must be detected. As many pieces of malware exercise extreme subtlety, this can be more difficult than expected (Furnell, Ward, *Malware*). Next, an effective defense would have to understand exactly what is being attacked in order to correctly choose what to protect (Schneidewind). Finally, the actual countermeasures must be deployed in order to defeat or mitigate the attack (Schneidewind). As this last step involves a decision of what countermeasures are most applicable and appropriate as well as the actual bandwidth to accomplish them, this is a definite time investment. All of this adds up to a substantial amount of time even without any human input. As a human decision process will be assumed to take at least longer than one second, this is an eternity in computational time. With a 1GHz processor, that one second delay is an additional billion cycles for the malware to accomplish its goal. Taking all of these steps into account, it is highly unlikely that a system will be able to respond to a cyber attack in real time.

Another valid strategy for preventing attacks is to inspire fear into potential enemies (Machiavelli). As has been established earlier, the lack of a valid counterattack strategy causes attackers to lose much of the fear which would be inherent in a traditional physical attack. In addition to being illegal in most jurisdictions, counterattacking in cyberspace is laced with so many difficulties relating to collateral damage against civilians that it is simply not an effective or moral strategy (Rowe,

Ethics). This combines with the fact that it appears immoral to react too strongly to cybercrime (Kierkegaard) to leave defenders with a dearth of valid options other than mitigation. All of these factors combine to prevent the cyber equivalent of mutually assured destruction, which was used as a form of nuclear deterrence during the Cold War (Dunn).

A different factor which causes cyber defense to be so difficult is the relative stagnation of security technologies. While attacker capabilities seem to advance daily, new innovations in access controls remain rare (Rowe, *Designing*). Indeed, there has been a significant worldwide increase in the number of zero day vulnerabilities discovered in the past several years (Dhamankar, Dausin, Eisenbarth, and King). There are several potential reasons for this increase, ranging from a shortage of skilled researchers (Dhamankar, Dausin, Eisenbarth, and King) to the fact that there has already been substantial research into the “first line of defense” (Rowe, *Designing*). However, one particularly fascinating rationalization explains this gap by analyzing the Internet as a biological ecosystem. If we assume that attackers occupy the role of predators in a traditional ecosystem and that the defenders are prey, an interesting relationship is demonstrated. Selection pressure is the process through which preferable traits are passed down within a group as a result of interaction between entities within an ecosystem (Furnell, Ward, *Jungle*). As there are around 1.7 billion individuals online (Nye), it makes sense to determine that the number of predators is much less than the number of prey. As such, the majority of prey will never have an interaction with a predator (Furnell, Ward, *Jungle*). Therefore they will see no reason to adapt their current activities online and wonder why so many individuals are worried about these online predators. Indeed, as many items of malware have very subtle effects, some prey won't even realize they have been successfully attacked until it is too late (Furnell, Ward, *Malware*). This results in a comparatively small selection pressure on each member of the prey group, as it is distributed across such a large population (Furnell, Ward, *Jungle*). The small number of predators compared to the number of prey ensures that each feels the effect of selection pressure more

acutely. As such, “the predator will always need to improve faster than the prey” (Furnell, Ward, *Jungle*). However, online predators are granted additional benefits that their biological counterparts lack: They are able to target multiple groups of prey with a single attack, they are not constrained by location or time due to the interconnectedness of the Internet, and they can easily learn from successful techniques employed by other predators and share knowledge efficiently (Furnell, Ward, *Jungle*). These factors all combine to cause predators to be a potent force on the Internet, making it even harder to defend as a member of the prey population. This accounts for the ability of attackers to keep their capabilities in lockstep with advancements in security technology, as well as rapidly adopting new infection vectors as they grow popular with the population as a whole. It seems that as something grows popular, such as social media sites like Facebook or Myspace, attackers quickly react and find a way to utilize these new vectors to attack prey (Furnell, Ward, *Jungle*). After examining the biological comparisons, the rapid response and obsessive attention to detail demonstrated by attackers is understood easier.

Yet another source of difficulty in effectively defending from malware and cyber attack is the fact that organizations must be ever vigilant, even against their own members. As attacks are likely to come from both within and without (Machiavelli), organizations must also put security measures in place to defend against their own members. Though it may be easier to think of an attacker as someone from without, they are still able to be a disgruntled employee rather than a competitor attempting corporate espionage. Indeed, it was an employee who failed to secure full-time employment that attacked his company's system and dumped 1 million liters of sewage into the water (Curran, Concannon, McKeever). According to a poll of over 600 organizations, insider attacks consisted of 21% of all cyber security incidents in 2010 (Henricks). This causes organizations to have to prioritize exactly what will be defended against, as they must pay the costs to defend against both outsiders and their own members (Orton).

These are some of the factors leading to why malware and cyber attacks are hard to defend against, yet relatively easy to perpetrate. This problem poses a distinct disadvantage for nation states seeking to extend their dominance into this digital frontier, as it is so unlike the other four domains associated with military power of land, sea, air, and space (Nye). Due to the myriad reasons explained above, offensive maneuvers appear to be more effective than defensive maneuvers in cyberspace. As nations or organizations gain more and more capabilities by integrating with the Internet, they also expose vulnerabilities that would be otherwise unknown (Nye). This is why it is important to discover as much as possible about both offensive and defensive maneuvers in a digital frontier, so that the limitations and vulnerabilities can be understood and mitigated.

1.7.0 To Build or Not to Build

Before creating a potentially damaging piece of software, one should examine the pros and cons inherent in their actions. As such, the benefits of creating a new piece of malware will be examined first, these will serve as potential reasons to act in this manner. To provide a counterpoint, the drawbacks inherent in creating a new piece of malware will then be examined.

1.7.1 Reasons to Create Malware for Testing

Of the potential reasons to create a new piece of malware for testing, some stem from the benefits associated with gaining knowledge. Others are in attempts to prevent individuals from accomplishing nefarious goals. Still others are in an attempt to result in better software overall. According to Scott Orton, one must be able to think like an attacker in order to defend (Orton). It makes sense then that the best way to become proficient in defending systems is to learn how to attack them. As such, the creation of malware in order to learn how to be a better defender should be considered a noble goal. It is easy to imagine many attackers using this fact to their advantage and

claiming to be developing malware for purely academic reasons. Therefore, it is important that those who are actually developing malware for academic reasons take steps ensuring that their creations never escape into the wild.

Another reason to research and create malware is to prevent other individuals from exploiting systems for nefarious purposes. The SANS Institute reports that different teams are discovering the same zero day exploits without collaborating (Dhamankar, Dausin, Eisenbarth, and King). As such, it is understandable that a researcher who discovers an exploit while writing a new piece of malware may not actually be the first individual to do so. By sharing the data associated with the exploit they ensure that other teams who also discovered this vulnerability are unable to utilize it in secrecy for malicious purposes. However, this goal can be accomplished simply by analyzing software and determining an exploit without actually implementing a payload or propagation mechanism. Therefore if the goal is to simply discover vulnerabilities it should only be considered moral when creating malware that lacks these facilities.

Yet another reason to create malware is that it can be argued to create a more robust information ecosystem. As attackers and defenders are locked in a perpetual arms race to continuously improve their capabilities (Orton), this may result in better programs overall. Some individuals argue that “the long-term outcome of a continuing arms race is an improved retort against the unfriendly actions already performed by the opponent, and probably reducing the number of obvious new exploiting behaviors” (Carlsson, and Davidsson). It is important to note that in order to have viruses be a beneficial agent to the system as a whole, the final result should be one in which there is a balance between attackers and defenders and “viruses only in exceptional cases do harm” (Carlsson, and Davidsson). However, this goal seems unlikely at the current rate, as the effects of selection pressure are much greater on attackers than defenders (Furnell, Ward, *Jungle*). Therefore it seems that unless there is a massive shift in the way software is developed by placing security first, this balance is

unlikely to result. As such, this particular reason to create malware is questionable at best.

1.7.2 Reasons Not to Create Malware (For Testing or Otherwise)

Now that potential reasons to create a piece of malware for testing have been examined, it is important to consider the consequences of such an action. Creating a piece of malware in an effort to become a better defender appears to be morally justified. The next idea, creating an exploit without any payload, seems promising at first glance. However, upon examining several pieces of malware in the wild, it becomes apparent that this is a primary method for attackers to test attack vectors that will later be reused with payloads (Nugent, Raisinghani). Furthermore, revealing an exploit to the general public does not actually appear to be a good method of preventing others from utilizing it for nefarious purposes. The SQLslammer worm, the fastest spreading worm ever, utilized an exploit that had been discovered almost one year prior and still managed to infect over 75,000 hosts (Heidari). This combines with the delayed response time to the revelation of zero day exploits mentioned earlier (Dhamankar, Dausin, Eisenbarth, and King) to call into question whether publishing an exploit so that others cannot use it is valid. If anything, it appears that it may be used to allow certain individuals to create pieces of malware by grabbing published exploits and adding their own payload and propagation mechanisms.

The next question is whether it is right for states to create cyber weapons as an effective nonviolent arsenal. There are several contradictions in this area, but one thing that can be agreed upon is the fact that the Geneva conventions forbid attacks whose effects cannot be controlled or whose damage to the civilian population is disproportionate (Rowe, *Ethics*). One of the primary infection vectors appears to seed through many intermediate networks and systems on the way to the final target; this is in order to make attribution more difficult and defeat existing security measures. As such, it appears that the effect of the malware is quite difficult to control. This can be remedied by ensuring

that the payload only activates upon reaching the final target, but the intermediate systems still undergo damage in the form of wasted bandwidth and cpu cycles in an effort to spread the attack (Rowe, *Ethics*). Additionally, while the direct effect of the piece of malware may be nonviolent, such as disabling a power plant without firing a shot, the ramifications could lead to violence. This can include lost lives due to hospitals being unable to provide services to patients, accidents and general chaos that results from citizens being scared, among others (Rowe, *Ethics*). Indeed, some of the more promising targets for a cyber attack using malware (power plants, financial institutions, communication systems) are actually civilian targets. An additional question arises over how much damage to cause. As many attacks are against data, it is tempting to go overboard when causing damage in an attempt to ensure that some lasting damage is done (Rowe, *Ethics*). This can be solved by ensuring that the attack taking place is well-researched, precise, and utilizes only the minimum amount of force. However, as software interactions are an imprecise science at best, even the most well researched and precise attack could have an error in the code which causes massive collateral damage to the victim's civilian population. A final question arises as to whether or not developing a cyber weapon is cost effective. While there are extremely low barriers to entry in the development of cyber weapons and malware (Knapp, Boulton), each particular weapon is at peak effectiveness only once (Rowe, *Ethics*). Unlike a new missile system, the time and cost that goes into creating and testing a cyber weapon is something that an organization will not be able to reuse when attempting another attack. However, the costs associated with the creation and testing of a cyber weapon are so low when compared to traditional weapons systems that it may be a moot point. Regardless of the cost-efficiency of a cyber attack, it is clear that states will have to be very careful when developing malware in order to ensure ethical behavior.

A final reason for an individual not to create malware is the emerging legal issue. With the ratification of the Convention on Cybercrime, both the creation and intention to create malware are now a crime (Kierkegaard). While creating malware for testing purposes on private machines is not

illegal, there are still legal questions which arise about publishing an exploit which someone else uses. If a researcher discovers a new zero-day exploit and reveals it to the world at large without utilizing it, this is not a crime (Kierkegaard). However, if the act is considered a crime in any nation which it occurred, the attacker is held liable regardless of their own location (Kierkegaard). Therefore, if the vendor is unable to get a patch and an attacker creates a particularly devastating piece of malware utilizing this exploit, it is unclear whether or not the original researcher will be held partially liable. As the treaty has a provision for “other criminal offenses not defined by the convention” (Kierkegaard), a nation could easily define publishing an exploit which is used for crime as aiding and abetting. This would cause publishing an exploit which is used for a crime to be illegal if the result affects any systems in the nation which introduced this particular provision. This puts nonstate actors who are looking to publish security research in a difficult position, as they lack the political clout of a state in order to defend themselves from legal ramifications.

All of these rationals combine to form a grim picture for security researchers. It is increasingly important to maintain logs of own actions in an effort to prove that there were no criminal activities being undertaken. Additionally, it appears that the only valid reason to create malware is to learn how to become a better defender. Publishing an exploit that is discovered is suddenly a questionable act; only further actions by governments who ratified the Convention on Cybercrime will establish precedents on what is permissible in this new era. While those who are creating malware for criminal purposes will likely ignore the results of these new prohibitions, it is the researchers who find themselves in a questionable position.

2.0 Background: “Good” Pieces of Malware

Now that the motivations and effects of malware have been established, it is possible to understand what makes an impressive specimen. By examining the current standard for vulnerability analysis and determining both pros and cons of such a system, the shortcomings can be realized. The alternative VIPERS system will then be analyzed as a potential replacement. Other effective characteristics of malware will be analyzed and placed within the VIPERS categories. Finally, the goal of the experiment will be examined: a cross-platform piece of malware. The problems inherent with creating such a piece of software will be explained as an addendum.

2.1 Current Standard: CVSS

The Common Vulnerability Scoring System (CVSS) version 2 is currently the standard used along with Common Vulnerability and Exposure (CVE) identifiers to define the name and impact of a particular IT vulnerability (Mell, Scarfone, Romanosky). CVSS v2 breaks each vulnerability into three distinct categories: Base, Temporal, and Environmental. The base metrics are utilized to enumerate items which will not change over time or environment; these include items such as how the vulnerability accesses the target system, how complex the vulnerability is to create or exploit, whether or not the vulnerability requires authentication in order to exploit, and how it effects the confidentiality, integrity, and availability of a system. The base metrics are the most essential components of the CVSS classification, and many vulnerabilities are simply reported as their overall weighted CVSS score and the base metric vector formed by combining all of the information within that category (Mell, Scarfone, Romanosky). Each sub item has three possible values; these provide gradation within the categories that the metric addresses. This category provides a good description of what the vulnerability actually does, as well as what is required for it to work correctly. This is very useful for organizations attempting to grade which vulnerability has the capability to be the most devastating and devote

resources accordingly.

The next category within the series of metrics is Temporal; these are items which may change over time. The first item is Exploitability, which determines whether or not the vulnerability has been proven to exist beyond theoretical abstractions and whether or not working example code exists. This factors into the overall threat of the vulnerability as an exploit with working code enables attackers with little skill to utilize it (Mell, Scarfone, Romanosky). Another subcategory within this metric is Remediation, which relates to whether or not there is a way to prevent this exploit from occurring. Vulnerabilities with no official fix are weighted as more severe than those which have workarounds published. The final subcategory within the Temporal metric is Report Confidence; this is whether or not the vulnerability has been confirmed by reputable agents. A fully confirmed vulnerability is considered more severe, as its existence is proven and attackers are able to acquire more information about how it works (Mell, Scarfone, Romanosky). All of these items are considered optional for filing a CVSS report. They have benefits to organizations prioritizing between different vulnerabilities, but offer little to no benefit to researchers attempting to determine the most effective pieces of malware. As such, they will be considered superfluous to the classification of a “good” piece of malware.

The third and final category within the series of metrics is Environmental; these vary by what particular information ecosystem a vulnerable system exists in. Example information ecosystems and their members include military, command and control infrastructures; financial, banking servers; commercial, inventory systems; and industrial, power plant controllers. Unlike the previous categories, some of the following are useful to determine whether or not a particular piece of malware is effective. As such, the analysis of whether or not each subcategory is useful will occur immediately after its description. The first subcategory within this metric is Collateral Damage Potential; it relates to the ability of an exploit to damage life, physical property, revenue, or productivity (Mell, Scarfone, Romanosky). Logically, those vulnerabilities that have the potential to inflict more damage are rated as

more severe than others. While this is interesting, the gradation within the category is inefficient for determining whether or not a piece of malware is effective; the only necessary factor relating to this category being if the specimen accomplishes its assigned objective. The next subcategory is Target Distribution, it represents the percentage of systems within the environment that could be affected by the vulnerability (Mell, Scarfone, Romanosky). The greater the proportion of systems within the particular environment that could be affected, the greater the severity of the vulnerability. This is another category that is partially useful for determining whether or not a particular specimen is well made. If the goal of the malware producer is to infect as many systems as possible in a particular environment, this category is important. If the attacker instead wants to assault a specific target, a low score in this category that includes the desired target is more valuable than a high score which does not include the desired target. The final three categories within the Environmental metric are the Security Requirements for confidentiality, integrity, and availability. Each of these items represents how important each facet of security is to a member of this environment, as they change dramatically between environments (Mell, Scarfone, Romanosky). For example, it makes sense that a clandestine service agency will care most about confidentiality compared to the other two facets of security. On the other end of the spectrum, public service utilities such as power plants will most likely care about integrity and availability most, confidentiality least. As these categories reflect how damaging a potential compromise of each of the facets of security is in a particular environment, they are not particularly useful when analyzing a specific piece of malware. Therefore these last categories are also deemed superfluous for the analysis of a well-crafted piece of malware.

While this system is valuable for organizations seeking to prioritize actions based on specific threats to their particular environment (Mell, Scarfone, Romanosky), it is overall less useful for researchers who are seeking to identify the paragon examples of malware. It does provide descriptions of what the exploit can actually accomplish and how it effects the three factors of security:

confidentiality, integrity, and availability. Additionally, establishing whether or not the vulnerability can be exploited remotely is an important piece of information. While the difficulty of creating the exploit and whether or not authentication is required does provide interesting information, they are unnecessary when determining the potency of a particular piece of malware. A glaring fault of the CVSS system is the lack of a benefit for being subtle. Utilizing this system, a piece of malware which changes the background to say “you have been pwned” would be weighted the same as a piece of malware which accomplished the same objectives without alerting the victim. Among other missing items, this demonstrates that the CVSS system is intended for vulnerabilities as a whole, rather than focusing on malware or cyber warfare. This demonstrates a need for an alternate system.

2.2.0 Alternate Standard: VIPERS Classification

In order to determine the gestalt of the effective characteristics of malware, it is necessary to create a different analysis standard. The VIPERS standard forms a vector without an overall score in an effort to enumerate the important facets of a piece of malware. The VIPERS system can be utilized to provide quick differentiation between individual examples by comparing along a specified axis or it can be used to gain a detailed view of a specific specimen. It is important to note that many of the sections require knowledge of the intention of the attacker, which can be problematic to acquire. As such, some sections will be extrapolations based on how the particular piece of malware interacted with its environment.

The classification is named VIPERS in an effort to provide a useful mnemonic to allow for quicker understanding of the system. The individual sections of the VIPERS classification are as follows: Virulence, Infiltration, Payload, Exclusivity, Remoteness, and Strength. Each section will be analyzed in order to establish how it benefits the classification system as a whole. Additionally, each section will be applied to an example in an effort to demonstrate an effective usage. As the VIPERS

classification is new, there are no preset numerical values associated with each section like CVSS.

These are outside of the scope of the current project, yet may be addressed in future research.

2.2.1 Virulence: The raw infectiousness of a piece of malware. This section is intended to analyze whether or not the attack can compromise the desired systems. It also measures whether or not the attack can spread effectively, if so desired by the attacker. This requires extrapolation as to the attacker's intentions in order to make an accurate judgment. By establishing the attacker's goals, a researcher can examine the aftereffects of the attack and determine how effective it was. The section provides important data for measuring the effectiveness of a piece of malware, as even the most sophisticated payload is rendered useless if it can never reach a target. As an example analysis, the SQLslammer worm can be considered highly virulent, as it was able to infect the desired windows machines. Furthermore, it was able to spread swiftly from those machines to other vulnerable systems. However, the worm's random number generator did have a bug which prevented it from attacking a large group of potential victims (Heidari). If this bug hadn't existed, the worm would have been considered extremely virulent.

2.2.2 Infiltration: The subtlety of a piece of malware. This section analyzes whether or not the attack is able to perform its mission while remaining undetected. This has many gradations, and is often opposed to virulence. For example, scanning worms such as the SQLslammer generate anomalous communications in an attempt to infect as many systems as possible, this increases its likelihood of being detected (Weaver, Paxson, Staniford, Cunningham). This is an important category for a piece of malware, as a subtle attack that goes unnoticed can be used for several purposes. It can silently report all data from the infected machine, it can be used to lace the infected system with backdoors to facilitate further attacks, or it can lie in wait for additional instructions in addition to performing its main objective (Weaver, Paxson, Staniford, Cunningham). One form of particularly subtle attack is the installation of a rootkit. Assuming that the initial compromise is handled stealthily, the actual rootkit

infects the system and then often changes the operating system reporting instructions to hide its presence (McAfee). Boot disk scans are redirected to copies of the uninfected boot disk in an effort to hide any sign of the system being compromised (McAfee). As such, many rootkits go completely unnoticed until they are used for further ends.

2.2.3 Payload: The ability of a piece of malware to accomplish its goal. This section is used to determine whether or not the function performed by the malware is truly effective. Like virulence, this section does require some extrapolation as to what the goal of the attacker was. In many cases, this can be determined by examining the code of the malware. If there is no payload, this means that the code is most likely being used to test a potential infection vector or demonstrate the existence of a vulnerability (Knapp, Boulton). Otherwise, the payload itself can be analyzed in an effort to determine what the goal of the attacker was. This is another important section to pay attention to for a piece of malware, as a well crafted payload can differentiate between a minor annoyance and a complete catastrophe. An example piece of malware with a known payload is the Stuxnet worm, which manipulated centrifuges in an effort to cause them to destroy themselves. While the complete impact of the worm will not be known for some time, if ever, the increased amount of mechanical failure in the Natanz plant demonstrates that the worm was effective (Nicoll).

2.2.4 Exclusivity: A multifaceted measurement made up of unique infections and system control guarantee. One of the base items within exclusivity is whether or not a piece of malware will waste resources by reinfecting a machine which has already been subverted. This is different than hiding copies of itself in an infected machine, as will be discussed in the strength section. The property of unique infection is important to consider as it both benefits subtlety and demonstrates a more efficient piece of malware. An example of this particular type of exclusive malware is a scanning worm which will not try to reinfect a system that it gets a connection to. This can be accomplished through a variety of methods, but an efficient and subtle version would be one which does not bother scanning that

particular address anymore. This will both cut down on potentially anomalous traffic as well as not wasting precious resources.

The other subcategory is that of system control guarantee. This treats each compromised system as an investment and seeks to ensure that the only piece of malware compromising the system is the one being examined. It is important to consider this when creating malware as the presence of other vulnerabilities prevents the attacker from guaranteeing the new functionality of the system. Additional pieces of malware can allow other attackers to exploit the newly controlled machines; they can even draw attention to the fact that the system is compromised if they behave less subtly than the original attack. Historical examples of malware which behave in this manner are the Bagel, MyDoom, and Netsky worms (Heidari). Each one installed their own payload and also removed instances of the other worms in an attempt to guarantee system control to the attacker (Heidari). Another example is the prevalent tactic among attackers to compromise a system and then patch it in an effort to prevent others from subverting the system in the same manner (Orton).

2.2.5 *Remoteness*: Another multifaceted measurement which consists of access requirements, remote activation, and remote control. Access requirements are one of the simpler factors, dedicated to determining whether or not the piece of malware is able to compromise systems from across networks. This is unashamedly mirrored from the Access Vector measurement of the CVSS base metric. As such, it is necessary for analysis due to the fact that an attack which can occur from anywhere in the world is substantially more powerful than one which requires local access such as a thumb-drive being inserted into a system. Examples of attacks which can occur across networks include web-sites which run malicious JavaScript in order to compromise a machine viewing the site (Dhamankar, Dausin, Eisenbarth, and King). At the opposite end of the spectrum, attacks which require local access include attacks over firewire or USB inputs (Mell, Scarfone, Romanosky).

The remote activation subcategory refers to whether or not the piece of malware requires

interaction with the user in order to be executed. If the attack is remotely activated, there is no user interaction required and the piece of malware is considerably more valuable. While effective social engineering can cause users to activate the piece of malware, as in the ILOVEYOU worm (Wassenaar, Blaser), these are considered less valuable as some users may not fall prey to the social engineering attack. Most worms appear to fall into the remotely activated category, whereas viruses are not considered to be remotely activated (Heidari).

The third and final subcategory relates to the ability of the attacker to modify or control the piece of malware remotely after distribution. Effective remotely controlled attacks allow for the attacker to change the way that the attack functions after the initial infection has been achieved. This allows for better adaptability to changing situations, such as the discovery of a new vulnerability that will allow for better propagation. Alternately, the payload mechanism may have been discovered and adjusting the attack after it has been released can help it remain effective. This adaptability inherited from remote control adds to the value of a piece of malware. It is important to note that the process of remote control can form several potential problems, maintaining secrecy being first and foremost. If there is a central entity sending updates, individuals analyzing infected systems can determine who is acting as the puppet master. Also, these anomalous communications could signal that a system is infected to any unaware individuals monitoring network traffic. A commonly seen example of a remotely controlled piece of malware is a zombie. These infected systems await instructions from a master attacker and are often utilized for other schemes at the attacker's whim (Heidari).

2.2.6 Strength: The ability of a piece of malware to resist attempts at removal. The strength or resilience of a piece of malware reflects any tricks bundled in to make it last as long as possible within compromised systems. This component comes into play after subtlety has failed and users attempt to reassert control over their own machines. It makes logical sense that a piece of malware which resists attempts to remove it should be considered more effective than one which is easily fixed. This can be

accomplished by seeding multiple copies of the malware throughout an infected system, or through even more devious methods. Polymorphic code changes automatically in an effort to prevent anti-virus programs from identifying it based on signature (McAfee). Additionally, some rootkits are able to survive reimaging the machine and simply reinfect the supposedly clean operating system installation by infecting the master boot record (McAfee). Other viruses utilize their strength to terminate processes associated with anti-virus programs; they ensure their survival by actively preventing attempts at removal (Furnell, Ward, *Malware*).

2.3.0 Justifying VIPERS

Now that the VIPERS classification has been defined, it is important to justify its existence. First, the comparison between the biological world and the digital world will explore two specific examples. Analysis of the successful criteria of a biological parasite and the VIPERS classification will shed light on the similarities between physical and digital malware. Additionally, the importance of adaptability in the physical and digital world will demonstrate further proof. Next, the importance of subtlety as a metric for success will be proved. These will all combine to form a picture of why the VIPERS classification is necessary in order to understand the best of what malware can be.

2.3.1 From Biology to Binary

The similarities between the biological and digital ecosystems provide important information about what makes an effective piece of malware. Biological malware has had millions of years of evolution to reach its current stage of efficiency, therefore it makes sense for attackers to strive to emulate it (Furnell, Ward, *Jungle*). The criteria of success in a biological parasite are as follows: “it spreads rapidly and effectively, it does not cause such a violent adverse reaction in a host that it is rapidly destroyed, and that it is able to extract valuable resources from the host” (Furnell, Ward,

Malware). These criteria will be addressed in order to explain their representatives in the VIPERS classification system. Virulence directly represents the ability of a piece of malware to spread to other systems effectively, as well as if it can infect the desired systems. As such, the first criteria for success in a biological parasite is explained clearly by the VIPERS classification. The next criteria for success is encompassed by a combination of the infiltration, exclusivity, and strength sections of the classification system. By infiltrating a system and keeping a host unaware, the attacker prevents the system from having a chance to react in a manner that destroys the malware. As the exclusivity section of the classification directly benefits infiltration, it can also be considered to help satisfy this requirement. Additionally, if the execution of the payload causes the system to react to the malware's presence, the strength and resilience of the code will prevent its destruction. Therefore a combination of the infiltration, exclusivity, and strength categories satisfies the second condition for success as a biological parasite. The final condition of success is encompassed by the payload section of VIPERS. An effective payload will take advantage of the compromised system and extract valuable resources by satisfying any of the motivations provided in the introduction section. While some consider valuable resources only as items which directly lead to monetary profit(Furnell, Ward, *Malware*), one can argue that any resource considered valuable by the attacker fits this requirement. Therefore the virulence, infiltration, payload, exclusivity, and strength sections of the VIPERS classification directly satisfy the requirements for success among biological parasites.

The field of biology relates another important factor for success in a predator or parasite: adaptability (Furnell, Ward, *Jungle*). Selection pressure is so much greater on predators that it is vitally important for attackers to adapt in order to survive (Furnell, Ward, *Jungle*). Digital attackers should follow suit, which is represented in the remoteness section of the proposed classification. The remote control function of a piece of malware allows it to be adapted on the fly, thereby increasing its chances to both survive and thrive in the digital ecosystem. Additionally, digital malware has begun to evolve

towards remote activation, which demonstrates the adaptability of attackers as a whole (Furnell, Ward, *Malware*). As such, this important factor of biological success is also mirrored in the VIPERS classification system by remoteness. This final addition means that all factors of the VIPERS classification system directly correspond to what makes malware successful in the biological world. Therefore, it can be said that the system is effective in describing what should be held up as good examples of digital malware.

2.3.2 *The Importance of Subtlety*

As Sun Tzu states in the very first chapter of *The Art of War*, “All warfare is based upon deception” (Tzu, Cleary). Seeing how malware infections are often referred to as cyber attacks, it makes sense that there will be similarities between traditional warfare and cyber engagements. Therefore, one should aim to deceive the victim as to their true purpose when attempting a cyber attack. The infiltration section of the VIPERS classification is primarily concerned with subtlety and deception, as has been explained above. However, it is important to prove the necessity of subtlety in order to justify the existence of the classification. In addition to the lessons of previous military advisers like Sun Tzu, current military personnel also stress the importance of deception. These claims will be examined and applied directly to cyber attacks. An attack that was crippled due to their lack of subtlety will be examined next to provide a counterpoint. Both of these examinations will serve to prove the importance of subtlety in malware.

Master Sun Tzu makes several claims regarding the effectiveness of deception, which mostly boil down to one should appear to be the opposite of what one actually is (Tzu, Cleary). This sentiment is built upon by professor Neil C. Rowe of the U.S. Naval Postgraduate School, who reiterates the importance of deception in malware. The nine military-relevant types of deception are concealment, camouflage, disinformation, lies, displays, ruses, demonstrations, feints, and insight (Rowe, Duong,

Cutsy). Most of these have direct applications in cyberspace and will be examined in the order they are mentioned.

Concealment is defined by the United States Army as protection from observation (U.S. Army); an attack which utilizes concealment would be a rootkit which has measures in place to prevent the system from reporting the infection. Camouflage is used to describe the process of blending into the surroundings and appearing as something different (U.S. Army); an example of a camouflaged attack is a Trojan. Disinformation involves “false and planted information” (Rowe, *Counterplanning*). An effective example of this is to seed false vulnerability reports in an effort to distract security researchers from actual exposures. Outright lies are easily understood and frequently take place as social engineering attacks. An attacker utilizing phishing techniques, such as claiming to need a user's account information in order to verify the security of an online bank account, is one prevalent form of this. Displays are “techniques to make the enemy see what isn't there” (Rowe, *Counterplanning*). A particularly devious example involves claiming that an infection was attempted, but foiled due to a user's anti-virus system, while secretly installing a rootkit. This increases user confidence in their existing protections while secretly subverting their machine. Ruses are defined as “using enemy equipment and procedures” (Rowe, *Counterplanning*) and are often employed as identity theft. They have lost some of their effectiveness overtime however, as “identity theft is easy and lacks surprise” (Rowe, *Counterplanning*); this constant exposure causes individuals to not easily trust claims of online identity. Demonstrations of capabilities are primarily used as a show of strength to discourage attack while not actually engaging the enemy (U.S. Army). These are less effective in cyberspace, as a demonstration of strength without actually making contact with a system is hard to accomplish (Miller). Feints are false attacks which engage the enemy in a different location than the actual attack (U.S. Army). These can be very useful in cyberspace when attacking human adversaries. By attacking a location and drawing attention to it, the adversary will traditionally make another point of their defense

weaker; this weakening allows the attacker to achieve a decisive victory at the desired location (Tzu, Cleary). The final category of military-relevant deception is insight; it is defined as “deceiving the opponent by out thinking him” (Rowe, *Counterplanning*). An example of insight employed in cyber attacks is a multiple stage attack. By bringing an organization's primary servers down utilizing a DDOS attack, an attacker can count on the defender bringing their backup servers online. As backup servers are traditionally less secure than primary servers (Think Smarter), the attacker can then utilize this knowledge to compromise the backups as they come online. The attacker has successfully anticipated the opponent's maneuvers and planned accordingly.

These aspects of military deception and their applications all serve to demonstrate the importance of subtlety in cyber attacks. As a counterpoint to effective operating methods, a historical example will demonstrate what happens when a piece of malware does not follow these guidelines. The Nimda worm combined effective strategies from two previous worms in order to become extremely virulent (Wassenaar, Blaser). The worm was released one week after the September 11th terrorist attacks on the United States and managed to spread faster than any previous worm (Wassenaar, Blaser). However, it was so virulent and released at such a reactionary time that the Internet community was able to realize the threat it posed and act quickly to stop it. Within hours of its initial release, system administrators were able to identify how to contain the spread of the worm and prevent a true epidemic (Wassenaar, Blaser). Had the worm employed subtlety in an effort to infiltrate systems before activating its payload, there is no telling how much damage it could have caused.

As these examples have shown, subtlety and deception are key to both physical and digital attacks. The VIPERS classification system is different from the currently used CVSS system in that it acknowledges this fact. Therefore it follows that researchers attempting to analyze what makes a piece of malware effective should utilize the VIPERS classification over the traditional CVSS system.

2.4 Enter the Plaguebringer

The initial goal of this senior project was the creation of a truly cross-platform piece of malware. Here cross-platform was defined as a piece of code which would infect systems regardless of what operating system or processor architecture was utilized. The idea behind the malware was that the general code would infiltrate a desired system and then call home in order to dispatch a tailored attack to that particular platform. As the general code would call in a specialized piece of malware to fully compromise the system it had infected, it was named the Plaguebringer.

The Plaguebringer would function in a similar manner to a military spotter. A spotter will travel light in order to avoid detection and infiltrate to an advantageous position as the companion to a sniper (Valdes). From there, they provide targeting information to the sniper and can radio in coordinates to artillery or close air support (Valdes). Likewise, the Plaguebringer was intended to infiltrate vulnerable systems and then radio in targeting information to a central entity. This central entity would then dispatch a tailored payload, equivalent to the sniper taking the assigned shot.

The justification behind creating a cross-platform piece of malware was the goal of being able to target the largest amount of systems. Doing so would also enable an attacker to catch a large portion of the Internet off guard. Just as biological viruses with broad host specificity are rare, so are pieces of malware which can impact multiple operating systems (Wassenaar, Blaser). This combines with ideas such as Linux being a more secure environment (Schneidewind) or that Apple OSX doesn't get viruses (Dilger) to form a large group of individuals who do not take security seriously. This appears to be a perfect storm which would allow an attacker to infect a large proportion of systems on the Internet.

The Plaguebringer would have accomplished this goal by attacking an application rather than a specific operating system. Applications often remain unpatched for longer than operating systems after a vulnerability has been discovered; thus applications have risen to become the primary infection vector for malware (Dhamankar, Dausin, Eisenbarth, and King). Furthermore, infected documents placed on

trusted sites are generally trusted by users; this would allow attackers to infect systems and use them as beachheads for attacking supposedly secure networks (Dhamankar, Dausin, Eisenbarth, and King). Due to the fact that Adobe Flash version 10 or below is utilized by 99% of Internet users in “Mature Markets” such as the US and Canada (Adobe Systems), it would make sense to utilize it as an attack vector.

2.5 A Stumbling Block

When researching the items necessary to create the Plaguebringer, several problems quickly emerged. First and foremost, malware is closed source and secrets of infection are jealously guarded by attackers (Rowe, Duong, Cutsy). Methods are employed to prevent decompilation and examination of the crafted malware in order to prevent other attackers or defenders from learning how the vulnerability was exploited. Additionally, vendors do not release working exploit code for fear of individuals taking advantage of systems which have not been patched successfully (Mell, Scarfone, Romanosky). As exploits written for Adobe flash are relatively recent developments compared to operating system specific worms, source code has yet to be released. This means that in order to develop the Plaguebringer, it would be necessary to write a new exploit.

There are a few problems inherent with exploiting a new vulnerability for a senior project. The first and most easily understood is a simple matter of it being outside the scope of a research project. More importantly, there are moral problems inherent with creating a new exploit. In order to prove the creation of such a worm, source code would need to be provided. However, providing source code would allow attackers to analyze how the exploit worked and use it for their own ends. Even if a worm was created without a payload, it would be a simple matter for other attackers to place their own within the code of the Plaguebringer. Even if the vulnerability was brought to the attention of the vendor, fixing vulnerabilities takes time. This gap would allow attackers free reign to exploit as many users as

possible before the vulnerability was fixed, as well as users still using older versions of the software. Seeing how some published vulnerabilities have taken up to two years to be fixed (Dhamankar, Dausin, Eisenbarth, and King), that is a substantial amount of users that can be damaged. Even using a utilitarian argument that the knowledge gained from creating the Plaguebringer would benefit the computing industry, the suffering generated by malicious attackers exploiting users would greatly outweigh this benefit. Therefore this action should be considered unethical under Utilitarian ethics (Mill). From a deontological perspective, Immanuel Kant argues that an act which harms other individuals can never be virtuous, regardless of the reasons behind the action (Kant). As the act of releasing source code that other attackers would exploit before the vendor was able to patch the software would harm users, the action can be considered unethical under deontological ethics.

If the creation of a new exploit for a senior project is unethical, there is one option left. Several sites sell exploits via a black market which directly supports cyber attacks. It is possible to buy an existing exploit and utilize it to generate a payload-free worm for academic reasons, but this also has problems associated with it. While the worm could be generated in a protected environment to ensure that any malware included in the purchased exploit doesn't spread, this does not address the primary issue. The purchase of an exploit through a black market would directly benefit criminals. In addition to the ethical questions relating to this purchase, there is the legal question. With the passing of the Convention on Cybercrime, it is unclear whether paying a criminal for information on how to commit a cyber attack for research is legal or not (Kierkegaard). While graduating with an amazing senior project is nice, graduating without going to jail is nicer.

These factors all combined to stop the creation of the Plaguebringer before it could begin. As such, the implementation section of this paper will be focused on several smaller examples which will not have catastrophic consequences for Internet users. While it would have been nice to apply the research into what makes a truly magnificent piece of malware, the research is still valid on its own.

3.0 Implementation: A VIPER Examined

This section will demonstrate a few of the available methods to attack individuals on multiple platforms. The first exploit is an infection vector which utilizes script code embedded in public websites to attack unaware individuals. The next item is not an exploit but a method to disguise malicious activities in an effort to increase the subtlety of an attack. The next item is a payload, which can benefit from both of the previous items. By combining all three items, a fully fledged attack is created.

3.1 Virulence: Cross Site Scripting with Flash

The first of the methods to be examined is known as a cross site scripting attack, often abbreviated as XSS. This particular version is performed by embedding JavaScript into a web page; users view the page and the script automatically executes. This is often performed on Web 2.0 sites such as forums which allow users to upload content. An attacker picks a site, uploads their malicious script, then counts on everyone else who utilizes the site to be infected. However, many sites are realizing this fact and attempting to escape JavaScript that is posted onto them. This can be avoided by instead placing the JavaScript into a flash file and placing the animation online.

An example use of this tactic is to create an animated signature for an online forum. Some sites allow flash animations to be embedded in posts in the form of signatures. This can allow unscrupulous individuals to embed the malicious JavaScript within the animation itself. Placing the script within an animation avoids the automatic filtering as it is inefficient to decompile every potential animation placed on the site and check it for JavaScript; therefore web servers will most likely not attempt to do this. There are two methods which will be examined to accomplish the goal of embedding malicious code within a flash animation; the first is for ActionScript 2 while the second is for ActionScript 3.

The first exploit utilizes the `getURL` method of ActionScript 2. `GetURL` is used to link to

another website, but can also be used to call JavaScript functions. The standard syntax of the function is `getURL(url [, window [, "variables"]]);` where `url` is the location to be directed to, `window` is optional and determines what the result will be loaded into, and `variables` are optional HTML `get` or `post` methods. However, this can also be used to execute JavaScript code by replacing the `url` variable with `"JavaScript: <script commands>"`. This allows for an attacker to run arbitrary JavaScript with the site's permissions. Attackers can utilize this fact to access user's session cookies, as well as activating numerous other payloads. While `getURL` is traditionally activated when a user clicks or interacts with an animation, devious users can instead cause the method to execute right as the animation loads. The following is an example of one such attack.

```
class xss extends MovieClip {
    // Constructor and other functionality removed
    private function onLoad() {
        getURL("JavaScript: alert(\'This could have been very bad\')");
    }
}
```

As both `getURL` and `onLoad` were removed in ActionScript 3, attackers have been forced to adapt. The fundamental action remains the same, the only difference is in the functions utilized. In order to execute a JavaScript function, one now calls the `navigateToURL` method; which takes a parameter of type `URLRequest`. This request can also be written as JavaScript in the same way that the URL string was created in the ActionScript 2 example. In order to execute immediately upon loading the animation, the malicious code can be placed within the constructor. Alternately, the code can be run by adding an event listener to the `MovieClip` which will execute when the clip is successfully added to the animation. The following is an example of an attack which utilizes the latter method.

```
// Ensure that all potential events will be recognized
import flash.events.*;
var myMovie:MovieClip = new MovieClip();
myMovie.addEventListener(Event.ADDED, xssFunc);

function xssFunc(event) {
    var xss:URLRequest = new URLRequest("JavaScript: alert(\'This could have been
    very bad\')");
    navigateToURL(xss, "_self");
}
```

Both of these example attacks rely on being able to post flash animations online, as well as users having flash player. However, they do not rely on a particular operating system or processor in order to achieve their goal. Both can also benefit from disguising the payload located within their JavaScript, which will be described in section 3.2. As they are fundamentally the same exploit, both are also blocked by browser add-ons such as NoScript; these prevent the execution of flash animations and JavaScript without explicit user permission. Even so, they demonstrate effective web-based attacks that leverage the emerging trends of user contribution inherent in Web 2.0.

3.2 Infiltration: Disguising Malicious JavaScript

JavaScript is a functional programming language, which allows for functions to be defined and used in the same line of code. This provides an excellent opportunity for disguising malicious code as something else entirely. If the malicious JavaScript can be named x , then there exists an encryption function f for which $f(x)$ does not resemble malicious JavaScript. If the creator of the encryption function then makes the inverse of that function g , such that $g(f(x)) = x$, they have the ability to disguise their code. By computing $f(x)$ before writing the final file, they have a block of encrypted text which they can place within their attack. This results in the original malicious code being written as a block of encrypted text with a decryption function defined and applied all in the same line. If the block of encrypted text is large enough, which is influenced by the encryption function and can be of arbitrary length, it will be very difficult to find the decryption function.

However, defenders have realized that a large block of gibberish will generally signify the presence of malware. This leads to the question of what the encrypted text should look like in order to not resemble disguised code. One potential option involves appearing incompetent rather than malicious. The encryption function will take a block of code that performs a legitimate function as a base argument. Then, the function will take the desired malicious code and traverse it, replacing every i th character of the base code with the next character of the malicious code. The decryption function will simply keep every i th character of the encrypted text and discard everything else. While it is extremely unlikely that the resulting block of text is executable code, it should remain close enough to pass a casual inspection. By formatting the resulting block extremely poorly, it will be even more difficult to read and allow for the decryption function to be embedded secretly. These factors will combine to result in code that appears to have been made by someone with little to no actual programming skill. The resulting opinion towards the code is that it is simply broken, which is an effective disguise as it will actually be a functional attack.

As a side note, attackers should be very careful in the selection of the legitimate base code. It should be complex enough that examination of changed characters will not easily demonstrate the maliciousness of the code, but should also have no association whatsoever with the attacker. If the attacker utilized published code which they were affiliated with, individuals analyzing the malicious code could have legitimate suspicions relating to the identity of the attacker. Selecting code published by an opponent is one option, but individuals analyzing the attack may correctly guess that someone is trying to pass blame to an enemy. As such, the safest choice for code is something with no relationship, good or bad, to the attacker.

3.3 Payload: Heap Spraying and Shellcode

One payload available is known as a heap spraying attack. This can be employed using JavaScript to fill a large portion of the heap with shellcode. In this instance shellcode is machine code which is used to perform malicious actions. Machine code is generally specific to individual platforms; a combination of operating system model, operating system version, as well as processor architecture (Offensive Security). However, it is possible to create a header which will operate on most machines and only execute the code tailored for that specific platform (Eugene). By filling the heap with multiple copies of the shellcode, attackers can exploit buffer or heap overflow errors and execute arbitrary code on a victim's machine.

By taking advantage of a heap overflow error, attackers overwrite memory and modify the pointer to a function; this is demonstrated in the aptly named Malloc Maleficarum (Phantasmal). Attackers change the pointer's address to the header of their shellcode (Phantasmal), which will then execute their malicious code instead of the normal function. Once the function is called, the shellcode executes and performs its duty. As the heap is dynamically allocated memory, the problem attackers face is knowing the address where their shellcode starts. If they point to an incorrect location, their attack will fail to work properly. One way to get around this is to guess an area and try repeatedly until it works or the program crashes (Sotirov), but this is inefficient and cannot guarantee proper functionality. It is possible to control the state of the heap using memory manipulation; this will provide a much greater chance of successfully executing the shellcode (Sotirov). While the example within Heap Feng Shui takes advantage of Internet Explorer (Sotirov), one can extrapolate that it is possible to commit the same actions on different browsers.

Because the heap spraying attack is executed in JavaScript, it benefits from the disguise method mentioned in section 3.2. Additionally, it can be delivered using multiple infection vectors, such as a flash animation on a forum. Combining all of the techniques yields an attack which is platform

independent, powerful, and easy to disguise. It requires user input in the fact that a user must view the infected page to become infected, but does not require local access in order to execute properly. The shellcode provided within the payload could allow for central control, as well as downloading and executing further malware such as a rootkit. The attack would need to be adapted in order to become exclusive and resilient, but this is possible to achieve using shellcode. Therefore the attack can be considered virulent, subtle, and remote; with the other factors of the VIPERS classification depending on the shellcode within the payload.

4.0 Conclusion – The Internet Used To Be A Nice Neighborhood

The field of malware has changed rapidly since the early days of computing. The emergence of the Internet has served as a catalyst for improvements in software and hardware; these influence the capabilities of attackers worldwide. Just as capabilities have evolved, potential motivations for creating malware have also emerged. Increased selection pressure on attackers drives continuous improvement in an effort to meet and surpass advancements in security technology. Whether this will end up leading to a more robust information ecosystem or simply become another battleground alongside land, sea, air, and space is yet to be seen.

Regardless of the final state of play, the field of malware appears here to stay. While existing vulnerability classification systems do an admirable job helping IT professionals prioritize time and effort, they are inadequate for determining what is an effective piece of malware. As such, it is necessary to create a new system of malware classification, VIPERS. This system is unique in the fact that it rewards subtlety while also gauging the success of a particular specimen of malware. While the system lacks weighted scores for individual categories, it is still in its infancy. As such, further research is necessary to make VIPERS a fully-fledged classification system.

While the creation of the Plaguebringer was never completed, not all was lost. The underlying research applied to the examination of several smaller facets, which served to demonstrate the VIPERS system. Additionally, the wealth of existing shellcode online allows for interested researchers to build upon the framework provided to create their own pieces of test malware. This small attack serves to demonstrate the importance of patching client software in order to prevent widespread user damage. It also brings up the question of whether it is more important to allow users to express themselves fully online, in the form of animated signatures, or to prioritize security. It is the author's opinion that freedom of expression ends when it damages another individual, therefore sites should err on the side of caution.

Works Cited

- Adobe Systems, Inc. "Flash Player Version Penetration." *Adobe*. Adobe, Dec 2010. Web. 15 Jan 2011.
- Arnes, Andre. "Identification and Localization of Digital Addresses on the Internet." *Cyber Warfare and Cyber Terrorism*. 'Ed'. Lech J. Janczewski and Andrew M. Colarik. Hershey, PA: Information Science Reference, 2008. Print.
- Carlsson, Bengt, and Paul Davidsson. "A Biological View on Information Ecosystems." CiteSeerX. Bleckinge Institute of Technology, 2001. Web. 29 Oct 2010.
- Curran, Kevin, Kevin Concannon, and Sean McKeever. "Cyber Terrorism Attacks." *Cyber Warfare and Cyber Terrorism*. 'Ed'. Lech J. Janczewski and Andrew M. Colarik. Hershey, PA: Information Science Reference, 2008. Print.
- Davies, Shaun. "The Internet Pranksters Who Started a War." *9 News*. MSNBC, 8 May 2008. Web. 15 Jan 2011.
- Dilger, Daniel. "The Mac Malware Myth." *RoughlyDrafted* 29 Jan 2009: n. pag. Web. 3 Mar 2011.
- Dhamankar, Rohit, Mike Dausin, Marc Eisenbarth, and James King. "Top Cyber Security Risks." *SANS.org*. SANS, Sep 2009. Web. 11 Dec 2010.
- Dunn, Lewis. France. *Deterrence Today*. Paris: IFRI Security Studies Center, 2007. Web. 11 Mar 2011.
- Eugene. "Architecture Spanning Shellcode." *Groar.org*. N.p., 09 05 2000. Web. 3 Mar 2011. <<http://www.groar.org/expl/intermediate/spanning.html>>.
- Furnell, Steven, and Jeremy Ward. "It's a jungle out there: Predators, prey, and protection in the online wilderness." *Computer Fraud & Security* 2008.10 (2008): 3-6. Web. 15 Jan 2011.
- Furnell, Steven, and Jeremy Ward. "Malware comes of age: The arrival of the true computer parasite." *Network Security* 2004.10 (2004): n. pag. Web. 15 Jan 2011.
- Gross, Michael J. "A Declaration of Cyber War." *Vanity Fair* Apr 2011: n. pag. Web. 3 Mar 2011.
- Heidari, Mohammad. "Malicious Codes in Depth." *SecurityDocs*. N.p., 11/29/2004. Web. 29 Oct 2010.
- Henricks, Mark. "Cyber Attacks Less Costly, More Common?." *Infosec Island*. 09 Feb 2011. Web. 23 Feb 2011.
- Johnson, Joel. "What is LOIC?." *Gizmodo* 8 Dec 2010: n. pag. Web. 15 Jan 2011.
- Kant, Immanuel. *Grounding for the metaphysics of morals; with, On a supposed right to lie because of philanthropic concerns*. 3rd ed. Hackett Publishing Company, 1993. Print.

- Kierkegaard, Sylvia M. "International Cybercrime Convention." *Cyber Warfare and Cyber Terrorism*. 'Ed'. Lech J. Janczewski and Andrew M. Colarik. Hershey, PA: Information Science Reference, 2008. Print.
- Knapp, Kenneth J, and William R Boulton. "Ten Information Warfare Trends." *Cyber Warfare and Cyber Terrorism*. 'Ed'. Lech J. Janczewski and Andrew M. Colarik. Hershey, PA: Information Science Reference, 2008. Print.
- Machiavelli, Niccoló. *The Prince*. SoHo Books, 2010. Print.
- McAfee. "Identifying and Thwarting Malicious Intrusions." *McAfee White Papers*. McAfee, 2010. Web. 15 Jan 2011.
- Mell, Peter, Karen Scarfone, and Sasha Romanosky. "A Complete Guide to the Common Vulnerability Scoring System Version 2.0 ." *CVSS*. CVSS, Jun 2007. Web. 15 Jan 2011.
- Mill, John S. *Utilitariansim*. London: Parker, Son, and Bourn, 1863. Print.
- Miller, Charles. "The Legitimate Vulnerability Market: the secretive world of 0-day exploit sales." *SecurityEvaluators.com*. Independent Security Evaluators, n.d. Web. 15 Jan 2011.
- Nico, Phillip. CPE 456. Computer Science Department. Cal Poly, San Luis Obispo. Lecture.
- Nicoll, Alexander, and Jessica Delaney. "Stuxnet: Targeting Iran's Nuclear Programme." *IISS Strategic Comments* 17.6 (2011): n. pag. Web. 11 Mar 2011.
- Nugent, John H, and Mahesh Raisinghani. "Bits and Bytes vs. Bullets and Bombs: A New Form of Warfare." *Cyber Warfare and Cyber Terrorism*. 'Ed'. Lech J. Janczewski and Andrew M. Colarik. Hershey, PA: Information Science Reference, 2008. Print.
- Nye, Joseph. "Cyberspace Wars." *New York Times* 27 Feb 2011: n. pag. Web. 3 Mar 2011.
- Offensive Security. "Exploit Shellcode." *Exploit DB*. Offensive Security, 6 Mar 2011. Web. 11 Mar 2011. <www.exploit-db.com/shellcode/>.
- Orton, Scott. Engineering Issues: Securing Cyberspace. eWeek 2011. Cal Poly, San Luis Obispo. 25 Feb 2011. Speech.
- Owen, Robert S. "Infrastructures of Cyber Warfare." *Cyber Warfare and Cyber Terrorism*. 'Ed'. Lech J. Janczewski and Andrew M. Colarik. Hershey, PA: Information Science Reference, 2008. Print.
- Rikstep. "The Love Bug: A Retrospective." *Rixstep Industry Watch*. Rixstep, 05 Apr 2004. Web. 15 Jan 2011. <<http://rixstep.com/1/20040504,00.shtml>>.
- Rowe, Neil C. "Counterplanning Deceptions To Foil Cyber-Attack Plans." *IEEE Computer Society* (2003): 203-210. Web. 11 Dec 2010.

- Rowe, Neil C. "Designing Good Deceptions in Defense of Information Systems." *Proceedings of the 20th Annual Computer Security Applications Conference (2004)*: 408-427. Web. 11 Dec 2010.
- Rowe, Neil C. "Ethics of Cyber War Attacks" *Cyber Warfare and Cyber Terrorism*. 'Ed'. Lech J. Janczewski and Andrew M. Colarik. Hershey, PA: Information Science Reference, 2008. Print.
- Rowe, Neil C, Binh Duong, and John Custy. "Fake Honeypots: A Defensive Tactic for Cyberspace." *Proceedings of the 7th IEEE Workshop on Information Assurance (2006)*: 223. Web. 11 Dec 2010.
- Schneidewind, Norman. "USA's View on World Cyber Security Issues." *Cyber Warfare and Cyber Terrorism*. 'Ed'. Lech J. Janczewski and Andrew M. Colarik. Hershey, PA: Information Science Reference, 2008. Print.
- Sotirov, Alexander. "Heap Feng Shui in JavaScript." *Black Hat Europe 2007*. Lecture. <<http://www.blackhat.com/presentations/bh-europe-07/Sotirov/Presentation/bh-eu-07-sotirov-apr19.pdf>>
- Suarez, Daniel. *Daemon*. E P Dutton, 2009. Print.
- Think Smarter. "Mint.com: Great concept, bad execution." *ThinkSmarter*. 10 Oct 2007. Web. 3 Mar 2011. <<http://think-smarter.blogspot.com/2007/10/mintcom-great-concept-bad-execution.html>>
- Tzu, Sun, and Thomas Cleary. *The Art of War*. Shambhala Publications, 2005. Print.
- Unknown. "Anonymous and Hacktivists." Online Posting to *SomethingAwful.com*. Web. 15 Jan 2011.
- U.S. Army. *MSL II: Foundations of Leadership*. BOLC I: Army ROTC. Pearson Custom Publishing, 2008. Print.
- Valdes, Robert. "How Military Snipers Work." *HowStuffWorks*. Discovery, n.d. Web. 6 Mar 2011. <<http://science.howstuffworks.com/sniper3.htm>>.
- Wassenaar, Trudy, and Martin Blaser. "Contagion on the Internet." *Emerging Infectious Diseases* 8.3 (2002): 335-336. Web. 15 Jan 2011.
- Weaver, Nicholas, Vern Paxson, Stuart Staniford, and Robert Cunningham. "A Taxonomy of Computer Worms." *Proceedings of the 2003 ACM Workshop on Rapid Malcode*. Washington, D.C., ACM. 2003. 11-18. Web. 11 Dec 2010.