An Assessment of the Contemporary Threat Posed by Network Worm Malware

Luc Tidy, Khurram Shahzad, Muhammad Aminu Ahmad and Steve Woodhead

Internet Security Research Laboratory
Faculty of Engineering and Science
University of Greenwich
Email: {l.j.tidy, k.shahzad, m.ahmad, s.r.woodhead}@greenwich.ac.uk

***Abstract*—The cost of a zero-day network worm outbreak has been estimated to be up to US$2.6 billion. Additionally zero-day network worm outbreaks have been observed that spread at a significant pace across the global Internet, with an observed infection level of more than 90 percent of vulnerable hosts within 10 minutes. The threat posed by such fast-spreading malware is therefore significant, particularly given the fact that network operator / administrator intervention is not likely to take effect within the typical epidemiological timescale of such infections. This paper presents a classification of wormable vulnerabilities, demonstrating a method to determine if a vulnerability is wormable, and presents a survey into the cause of the reduction of worm outbreaks in recent years, as well as their viability in the future. It then goes on to explore recent wormable vulnerabilities, and points out the issues with operating system security in relation to techniques used by zero-day worms.**

Keywords*—*Cyber Defence; Malware; Network Worm; Zero-Day Worm; Simulation; Modelling

I. INTRODUCTION

As a type of malware that exploits vulnerabilities that have not been patched or acknowledged at the point of an outbreak, with an automatic propagation method that can spread pervasively throughout a network, zero-day worms are particularly virulent. The effects are exacerbated by either a lack of detection or a high speed of propagation [1]. The threat presented by such malware to the Internet, national security and defence systems is therefore significant.

In the first few years of the twenty-first century, there were a number of notable zero-day worm outbreaks [2][3][4] however, since these events the number of zero-day worm outbreaks has reduced. Understanding this reduction, and assessing whether such worm outbreaks are still viable in a modern setting are essential. This paper presents a discussion of historical worm events to ascertain why they occurred, and then discusses the motivations for malware attacks to assess why worm outbreaks have seen this reduction. The paper then presents a discussion on recent wormable vulnerabilities and operating system security, in order to assess whether zero-day worm outbreaks are still viable on the modern Internet.

The remainder of this paper is presented as follows: Section II presents a lexicon as a definition of terms. Section III presents related work, focusing on similar studies into the assessment of potential threats. Section IV presents a discussion on the motivations for carrying out a malware attack. Sections V and VI present a summary of recent wormable vulnerabilities, and addressing the particular issue of operating system security. Finally, the paper is concluded in Section VII.

II. LEXICON

A lexicon has been presented for the clarification of the following terms, owing to their specific use in this paper.

Zero-Day Worm: In this paper, this is defined as a type of malicious software that propagates automatically without human interaction, using a vulnerability that has not been patched or widely acknowledged at the point of an outbreak. In particular, this paper reports findings on fast, random-scanning worms [5]. This is a similar definition to the taxonomy described by Weaver et al. [6], and other published literature (see [2][3][4]).

Wormable Vulnerability: A vulnerability that has the potential for use in worm propagation, as defined by being network accessible, allowing the execution of arbitrary code and whether a not a vulnerability can be exploited remotely. This is in accordance with the model reported by Nazario et al. [7].

III. RELATED WORK

Research into worms and their outbreaks has been reported in three key areas: the classification of worms and wormable vulnerabilities, potential worm outbreak scenarios and the investigation of previous worm outbreaks. In addition, this paper also considers contemporary malware threats.

A. Classification of Worms and Wormable Vulnerabilities

The taxonomy reported by Weaver et al. [6] presents an overall method of classifying worms. The classification is made under the following categories: target discovery, propagation method, activation, payloads, motivations and attackers. Similar categories are reported by Li et al. [8], which classified worms under a number of schemes: target finding, propagation, transmission and payload. Smith et al. [9] also uses the taxonomy reported by Weaver, however, expands this further to consider evasion and detection methods, which incorporate

 ISBN: 978-1-61208-368-1

different propagation methods and payloads. For the purposes of this paper, we choose to focus on self-carried worms, or worms that do not require other network traffic in order to propagate.

Another factor of classifying worms is the vulnerability they exploit in order to propagate. As reported by Nazario et al. [7], a wormable vulnerability can be summarised in (1), where wormability, W, is a product of the exploit characteristics, E, population characteristics, P, and the time since the disclosure of the vulnerability to account for development of the worm. Nazario et al. also defines the characteristics of a wormable exploit, as shown in (2), where the exploit characteristics, E, are defined by the fractional population of exploit architecture, f_{E_p}, the fractional availability of an exploit for a given vulnerability, f_{E_a}, the number of chances available to attempt an exploit, E_c, the fraction of exploit reliability, f_{E_r}, the Boolean value of whether the exploit is able to be made remotely, R, if the impact of the vulnerability is execution of code, I_e and if the impact of the vulnerability permits network access, I_n.

$$W = E * P * L \tag{1}$$

$$E = f_{E_p}(f_{E_a} + 0.067)(\frac{E_c - 1}{E_c} + f_{E_r})RI_eI_n \tag{2}$$

Using the key factors reported by Nazario et al., and those reported by Weaver at al., Li et al., and Smith et al., a wormable vulnerability can be summarised in the Boolean equation (3), where a wormable vulnerability, V_w, is determined by not requiring human interaction, H, is network reachable, N_r, provides remote code execution, R, and provides network access, N_a once exploited.

$$V_w = \bar{H} \bullet N_r \bullet R \bullet N_a \tag{3}$$

In addition to the reported work that provides a classification, there are also a number of resources that focus on providing details for known vulnerabilities. One such source is the Common Vulnerabilities and Exposures (CVE) system [10], which provide details for a range of vulnerabilities. The CVE system notes the access vector, for instance if the vulnerability is network reachable or requires human interaction, and the impact if the vulnerability were to be exploited, for instance providing remote code execution or network access. These details provide information in order to assess whether a vulnerability is wormable.

B. Potential Worm Outbreak Scenarios

Potential worm outbreak scenarios often focus on new technologies or methods that a worm may use in order to spread faster. As far as the authors are aware, the first notable instance of this was the work reported by Weaver in 2001 [11], which described a Warhol worm - where using a combination of a list of known vulnerable hosts, known as a hitlist, and by dividing up how each worm scans for new susceptible hosts, known as permutation scanning, the worm increases in virulence. Such methods were seen in the Witty outbreak of 2003 [4], and the second version of Code Red, Code Red II, in 2001, respectively.

Work reported by Staniford et al. [12], presents results on the impact of very fast, what is termed as Flash worms, on a contemporary Internet as of 2004. Using simulation, Staniford estimates that an optimised Flash worm could spread within seconds. Similar fast outbreaks are further corroborated in work reported by Tidy et al. [13], as well as reporting work on other potential scenarios in [5], where a worm uses an intentionally slow phase before switching to a fast, random-scanning method in order to increase the number of infected hosts prior to its fast phase; resulting in an impact similar to having a hitlist.

The work in potential worm outbreaks assume that a wormable vulnerability exists, however, there is limited work in investigating contemporary vulnerabilities in order to determine if they are wormable, and the possible worm outbreaks that could occur.

C. Previous Worm Outbreaks

There have been a number of large-scale zero-day worm outbreaks, most notable of which are the Morris Worm outbreak of 1988 [14], the Code Red outbreak of 2001 [2], the Slammer outbreak of 2003 [3] and Witty outbreak of 2004 [4]. Table I summarises these worms, detailing the platform/service that had the wormable vulnerability, the port/s used for propagation, and the exploit method. This shows that these notable events all used a buffer overflow in order to infect susceptible hosts, propagated using different ports and exploited vulnerabilities on a number of different platforms.

Another reported characteristic of these worm outbreaks centre around their payload. Both the Morris and Slammer worms contained no destructive or directly malicious content as part of its payload. Similarly, the Code Red worm only began to undertake a denial of service attack after it had completed a propagation phase. As reported by Shannon and Moore [4], the Witty worm was the first to carry a destructive payload, overwriting randomly chosen sections of the infected hosts hard drive with the phrase "`(^.^) insert witty message here (^.^)`".

Owing to the lack of malicious payload in the Slammer worm, the intentional pause in propagation in the Code Red worm and as the Morris worm was described by its author to be designed to gauge the size of the ARPANET, it can be argued that the motivation to release these worms was one of discovery. Similarly, as the Witty worm was the first of these

TABLE I
SUMMARY OF NOTABLE WORM CHARACTERISTICS

Name	Vulnerable Platform/Service	Port/s	Exploit Method
Morris	DECX Sun 3, sendmail finger	25,79	Buffer overflow
Code Red	Microsoft IIS web service	80	Buffer overflow
Slammer	Microsoft SQL Server 2000	1434	Buffer overflow
Witty	Internet Security Systems firewall	Random	Buffer overflow

 ISBN: 978-1-61208-368-1

to carry a destructive payload, it could have been released to assess whether a destructive payload was feasible.

D. Contemporary Malware Threats

Since the large outbreaks at the beginning of the 21st century, the number of large-scale worm outbreaks has decreased significantly. Panda Security [15] reports that worms only constituted approximately 6% of all malware infections in the first quarter of 2013, it is also reported that trojans constitute the majority of the malware infections with 80% of all malware infections being of this type. One of the largest of these is the Zeus trojan [16], which is designed in order to commit fraud by gaining access to banking details on infected hosts and sending these details to the attacker. This is defined by Wilson [17] as cybercrime, or criminal activity that is "enabled by, or that targets computers".

A return to worm-like characteristics can be seen in the Stuxnet [18] outbreak, which targeted industrial control systems in order to cause damage. It is suggested that Stuxnet is an example of cyberwarfare [19], where the intent was to cause damage to the targeted industrial systems. This is a distinct difference in the cybercriminal activity, as instead of criminal gain the motivation of released Stuxnet was one of causing damage.

IV. Motivations for Malware Attacks

One of the main factors in understanding malware outbreaks is the motivation of the attacker. A difference in motivation can influence the choice of malware that an attacker will choose, given that different malware is more effective at certain tasks than others. In the case of worm outbreaks, this is demonstrated by the reduction in events, owing to a change in the motivation of attackers. Figure 1 illustrates this change, plotting the trend of worm prevalence against time, along with three categories of attacker motivation: experimentation or discovery, cybercrime and cyberwarfare.

Up to the first few years of the 21st century, the use of malware was comparatively in its infancy, and the notable worm outbreaks during this period can be argued to have been for experimental purposes, with the main motivation of the attacker to see if they are feasible; or in the case of the Morris worm to measure the size of the ARPANET. From around 2004 onward, the use of malware for cybercrime has increased. Such criminal activity, as shown by the prevalence of trojans like Zeus [16], has focused on gaining access to confidential data or disrupting services, such as a Distributed Denial-of-Service attack (DDoS) [20], through the use of controlling a large number of machines through a botnet created using a trojan.

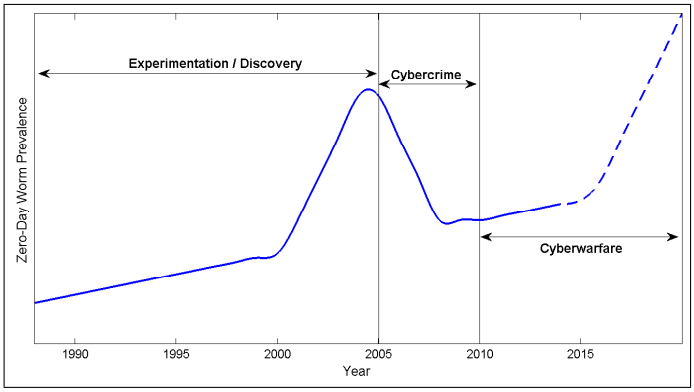

Fig. 1. Trend of Zero-Day Worm Prevalence

Although worms can be used in order to create botnets and carry out DDoS attacks, other methods have been chosen by attackers. Part of the reasoning for this, is that a large-scale, fast random-scanning worm outbreak is easily detectable, and it is often the intent of an attacker to avoid detection for as long as possible. Additionally, as has been shown by the Slammer outbreak [3], there is the possibility that a particularly fast worm can impede the network traffic, that in the case of a botnet, may disrupt the ability of an attacker to issue commands to or receive information from infected hosts.

As it has been shown by the worm-like Stuxnet outbreak [18], if it is the intention of an attacker to cause damage then the use of worms becomes a more attractive option. Although these have been isolated to targeted attacks to date, if it is the intention of an attack to disrupt communication or target the network infrastructure, such as in a cyberwarfare scenario, then the use of worms becomes a much more viable option. Additionally, if the motivation of an attack is to cause disruption of the Internet, then worms also present a viable option for attackers, even in the absence of a payload that causes damage.

Given the further shift of motivation from cybercrime to cyberwarfare, this also depends on the existence of wormable vulnerabilities, in order to exploit and carry worm attacks in the future.

V. Recent Wormable Vulnerabilities

Equation 3 presents a method of assessing whether or not a vulnerability is wormable. This Section presents five case studies of contemporary wormable vulnerabilities, along with their CVE code [10] for reference.

Microsoft Remote Desktop Protocol (RDP) - 13/03/2012 - CVE-2012-0002

The Microsoft RDP is a method for users to remotely access Windows-based hosts across a network. This vulnerability was present in a number of Windows versions, including XP, Vista, 7, Server 2003 and Server 2008. This allows an attacker to send a crafted packet on port 3389 to the host running RDP, and then potentially gain remote code execution. Having gained access to execute remote code, the attacker could then use this to send copies of the malicious packet

This wormable vulnerability is of particular note owing to the potentially large number of susceptible hosts to such an attack. W3Counter [21] reports that these recent editions of Windows constituted of approximately 3 billion Internet-connected hosts in 2012. As RDP is disabled by default, this requires being enabled manually. One estimate for the number

 ISBN: 978-1-61208-368-1

of RDP enabled hosts is one in every 10,000 [22], or 300,000 hosts; resulting in a similar proportion of vulnerable hosts to the Code Red outbreak in 2001. As has been reported in two of the authors previous work [5], such a large proportion of susceptible hosts could result in a particularly virulent worm outbreak.

BigAnt Message Server - 09/01/2013 - CVE-2012-6275

The BigAnt instant messaging (IM) software is an instant messaging solution targeted towards business use. By using a buffer overflow present in the message server portion of the software, an attacker is able to send a crafted packet and execute remove code on the targeted machine. As the software links with Microsoft Active Directory, this can include ascertaining user account details, potentially having a wider impact than just the host running the message server. This can also lead to network access, allowing copies of the malicious packet to be sent to other hosts running the message server software.

Although lacking the install base of the Microsoft RDP vulnerability, this is of particular note owing to its use in a corporate setting, as well as potentially allowing access to further details that could lead to further issues. This vulnerability, as far as the authors are aware, also has yet to be patched and details of how to exploit this vulnerability are publicly available.

VMWare vCenter - 25/04/2013 - VMSA-2013-0006.1

VMWare vCenter is a management platform for virtualised hosts. A number of CVEs reported under the VMWare security advisory VMSA-2013-006.1 [23] detail how an attacker may leverage Microsoft Active Directory integration in order to gain authentication on Windows-based servers running vCenter (CVE-2013-3107), and then use this authentication in order to execute remote code using another vulnerability (CVE-2013-3079). This access provides administrative privileges to the host system, enabling the attacker to then send copies of the malicious packet/s used to other susceptible hosts.

As one of the largest vendors for virtualisation software, a vulnerability in VMWare software presents a scenario where a substantial number of hosts may be susceptible to an attack. Furthermore, access to the virtualisation environment may further allow access to the virtualised hosts that are currently running on it. This vulnerability has since been patched by VMware, however, it demonstrates that virtualisation can present a vulnerability for future worm outbreaks.

ASUS RT-AC66U Router - 26/07/2013 - CVE-2013-4659

The ASUS RT-AC66U router is a router produced for the consumer and small office market. Using a vulnerability in the Broadcom ACSD service allows an attacker to send a crafted packet on port 5916 causing a buffer overflow. This allows administrative access on the target device, providing remote code execution and the ability for the router to send copies of the malicious packet to other susceptible hosts. As far as the authors are aware, no known patch is available for this vulnerability and proof of concepts are currently available.

This vulnerability demonstrates that not only do server and desktop hosts require consideration when considering potential worm outbreaks, but also that of routing infrastructure. In addition to gaining access to further propagate itself, administrative access to the router may also allow for further attacks, including man-in-the-middle or denial of service attacks against hosts connecting to the Internet through this router.

systemd 208 and prior - 20/09/2013 - CVE-2013-4391

Designed specifically for Linux-based operating systems, systemd is a system management service, or daemon, that forms part of the Linux startup process. By using a crafted packet, a buffer overflow can be cause resulting in allowing remote code execution. In addition with another vulnerability, CVE-2013-4394 [10], administrative access can be gained, therefore allowing network access to send copies of the malicious packet/s to other susceptible hosts.

This vulnerability demonstrates that other operating systems, aside from Windows, can also be subject to a wormable vulnerability. It also demonstrates that software required by an operating system for basic functionality, as opposed to additional functionality in the case of the Microsoft RDP vulnerability, can also be vulnerable.

A. Host Discovery

As highlighted in the work reported by Shannon et al. [4] and Staniford et al. [12], the use of a hitlist is one key method of increasing the virulence of a worm outbreak. Given that a number of unpatched vulnerabilities have been highlighted, it is of note that there now exist a number of services that catalogue information provided through the use of meta-data. One such service, Shodan [24], is freely available and allows the collated download of search results at a small price. Such a service could be used in order to collate information prior to a worm outbreak, in order to create a hitlist.

B. Susceptible Population

A key factor in determining the virulence of a worm is the number of susceptible hosts that a worm can infect. As has been demonstrated in some of the authors previous work [5][13], and the measure of exploitability by Nazario et al. [7], the larger the proportion of susceptible hosts on a network both virulence and exploitability increase. In the case studies presented, those vulnerabilities that would provide the greatest number of susceptible hosts, are vulnerabilities in operating systems. Therefore, it is pertinent to further investigate operating system security.

VI. Operating System Security

A. Operating System Memory Security

The main method for exploited vulnerable hosts, allowing for remote code execution, has been the use of buffer overflow exploits (as demonstrated in table I). This has prompted the development of a number of techniques in order to prevent

 ISBN: 978-1-61208-368-1

the writing of arbitrary data in the memory addresses that are being used by a program; and therefore providing remote code execution. The prevention techniques that are widely adopted in modern day operating systems are Address Space Layout Randomisation (ASLR), Data Execution Prevention (DEP), using No eXecution (NX) and canaries.

1) Address Space Layout Randomisation: ASLR is a countermeasure mechanism [25] adopted by operating systems to randomize the positions of executable code and data in memory at each run of a program. Randomising the base address of important memory structures, such as the stack and heap, makes the virtual address needed to perform a control-flow hijacking attack unknown. However, some techniques [26][27] have been reported that can bypass the randomness of ASLR mechanism.

a) Non-ASLR Memory: A non-ASLR module that runs on ASLR enabled operating system can be used to circumvent the ASLR protection mechanism. This can be a shared library in Microsoft Windows compiled without ASLR support for compatibility reasons. When an application that is non-ASLR is executed, the application tends to load its executables at runtime at a fixed memory address, thus allowing critical memory sections to be overwritten, or changing memory location. Additionally, using return-oriented programming techniques the contained data can be abused in order to leak additional memory addresses.

b) Information Disclosure: An information disclosure vulnerability can be used to leak memory locations of elements known to be at fixed addresses. For example, an out-of-bounds memory access vulnerability can be used to read a function pointer, and then send the value back to the remote server. Consequently, the server will control the size parameter of the function and accurately trigger an out-of-bounds read. As a result, the address of the public function is leaked. Based on this address, the memory layout of a corresponding executable file can be inferred.

c) Heap Spraying: Heap spraying is a technique used to allocate a substantially large amount of memory and fill it with a concatenation of multiple copies of a block of data. This is intended to create heap blocks using scripting languages so that a reliable location can be attained, then execute shellcode without looking for an offset in the memory address. This can greatly increase the probability that a chosen address will point to the beginning of the block even in the presence of randomisation.

2) Data Execution Prevention: Execution prevention [27][28] is another important countermeasure used to prevent arbitrary code execution even when an attacker has gained control over the processor's instruction pointer. This technique marks memory regions of executable application or service as writable or executable, but not both at a time. Popularly known as DEP on Microsoft Windows systems, it utilizes a hardware feature of the processor known as the NX bit. This marks writable memory regions, including the stack, as non-executable. Thus, when an address from this memory region is loaded as the instruction pointer, the processor will notice the non-executable flag and then raises a kernel level exception. The kernel will then send a segmentation fault signal to the program and thus terminate the program. Techniques used to circumvent DEP include return into libc, Return-Oriented Programming (ROP) and stack pivoting.

a) Return into libc: This technique [25] bypasses DEP by using the code of the running program or its shared libraries for malicious purposes instead of its intended use. This is achieved since the code is used by the running program itself, then the memory space utilised by the program is marked as executable. For example, in the Windows operating system an attack that uses WinExec and its functions (normally found in ntdll.dll) bypasses DEP as these are stored in an executable part of the memory. Thus malicious code can be copied to the executable memory space giving the attacker control of applications and services as described in [29].

b) Return-oriented Programming: This technique [25] allows an attacker to take control of the processor's instruction pointer and the stack area where return addresses are stored. Small pieces of code called gadgets are chained together to execute a chosen functionality instead of executing the intended functions. These gadgets are simple instructions followed by a return statement. For example, the statement in Figure 2 moves the content of the stack `esp` to `ecx` and then loads the next address from the top of the stack into the processor's instruction pointer through the return statement. This technique can successfully bypass DEP using WinExec as reported in [30].

c) Stack Pivoting: This technique [25] is an improvement of return-oriented programming by utilizing a special ROP gadget in order to make return-oriented programming possible through arbitrary overwrites. Having taken control of the processor's instruction pointer, an attacker will use the pointer to jump to a gadget that modifies the stack pointer to make it point to a controlled location. This can be accomplished directly through an arithmetic operation or by gadgets containing the `popq` instruction. It is intended that the controlled stack area will contain the ROP shellcode that will be executed subsequently.

3) Canaries: This is a compiler technique [31] that protects the stack by inserting a guard, a randomly chosen integer, at the start of the program between the protected region of the stack and the local buffers, i.e., a canary value is placed after the return address. Therefore, overwriting the return address will change the canary value, which is normally checked before a function uses the return address. The function will compare the value on the stack and the original value of the canary, if these values are different, then a message is generated in the system logs and the program will be terminated.

```
mov esp, ecx
ret
```

Fig. 2. Example Return-Oriented Programming Gadget

 ISBN: 978-1-61208-368-1

B. The Windows XP Opportunity

It has been estimated that Windows XP still constitutes 26% of all operating systems installed on desktop hosts [32]. As of the 8th April 2014, the extended support for Windows XP was discontinued. This meant that from this date there were no longer any security patches or support for this version of the operating system being made available for free. Although what is termed "critical patches" will be made available to paying customers. Additionally, after the 14th July 2015, the built-in anti-malware tools, Security Essentials and the Malicious Software Removal Toolkit, will no longer be supported.

Given this lack of support, if vulnerabilities are found in this version of the Windows operating system, it increases the likelihood that these systems will be susceptible to a future worm outbreak. This presents a particular issue, for instance, Slammer was able to cause disruption with less than 1% of the hosts at the time being susceptible to its infection vector [3], therefore it is reasonable that should a Windows XP vulnerability be exploited by a Slammer-like attack it could cause significant network disruption.

VII. Conclusion

Since the turn of the 21st century, zero-day worms have constituted a considerable threat to the Internet. Since 2005 there has been a reduction in the number of worm outbreaks, which can be attributed to a shift in the motivation of attackers from a period of experimentation and discovery to that of criminal activity. As such activity is better suited to the use of other types of malware, such as trojans, this reduction is reasonable. With the advent and increase in prevalence of cyberwarfare, worms once against become a weapon of choice for attackers, owing to their fast propagation and ability to cause considerable damage.

This paper explored the contemporary availability of wormable vulnerabilities and discusses the increased proportion of susceptible hosts made available by exploiting operating system vulnerabilities, highlighting the common techniques used in order bypass the most common techniques for preventing the exploitation methods used by zero-day worms. Furthermore, it highlights the opportunity that has arisen for attackers with the end of extended support, and future end of anti-malware support, for the Windows XP operating system.

References

[1] B. Ediger, "Simulating Network Worms - NWS Network Worm Simulator," http://www.stratigery.com/nws/, Sep. 2003, retrieved: 28th July 2014.

[2] C. C. Zou, W. Gong, and D. Towsley, "Code red worm propagation modeling and analysis," in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 138–147.

[3] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "The spread of the sapphire/slammer worm," *IEEE Security and Privacy*, vol. 1, no. 4, pp. 33–39, 2003, retrieved: July, 2014.

[4] C. Shannon and D. Moore, "The spread of the witty worm," *Security & Privacy, IEEE*, vol. 2, no. 4, pp. 46–50, 2004.

[5] L. Tidy, S. Woodhead, and J. Wetherall, "A large-scale zero-day worm simulator for cyber-epidemiological analysis," vol. 3, no. 1. Universal Association of Computer and Electronics Engineers, 2013, pp. 69–73.

[6] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A taxonomy of computer worms," in *Proceedings of the 2003 ACM workshop on Rapid malcode*. ACM, 2003, pp. 11–18.

[7] J. Nazario, T. Ptacek, and D. Song, "Wormability: A description for vulnerabilities," *Arbor Networks (October 2004)*, 2004, retrieved: July, 2014.

[8] P. Li, M. Salour, and X. Su, "A survey of internet worm detection and containment," *Communications Surveys & Tutorials, IEEE*, vol. 10, no. 1, pp. 20–35, 2008.

[9] C. Smith, A. Matrawy, S. Chow, and B. Abdelaziz, "Computer worms: Architectures, evasion strategies, and detection mechanisms," *Journal of Information Assurance and Security*, vol. 4, pp. 69–83, 2008.

[10] M. Corporation. (2014, April) CVE - common vulnerabilities and exposures. Online. Retrieved: July, 2014. [Online]. Available: https://cve.mitre.org/

[11] N. Weaver, "Warhol Worms: The potential for very fast internet plagues," http://www.iwar.org.uk/comsec/resources/worms/warhol-worm.htm, 15 Aug. 2001, retrieved: July, 2014.

[12] S. Staniford, D. Moore, V. Paxson, and N. Weaver, "The top speed of flash worms," in *Proceedings of the 2004 ACM workshop on Rapid malcode*. ACM, 2004, pp. 33–42.

[13] L. Tidy, S. Woodhead, and J. Wetherall, "Simulation of zero-day worm epedimiology in the dynamic heterogeneous internet," *Journal of Defense Modeling and Simulation*, 2013, in Press.

[14] E. H. Spafford, "The internet worm program: An analysis," *ACM SIGCOMM Computer Communication Review*, vol. 19, no. 1, pp. 17–57, 1989.

[15] Panda Security. (2013, May) Pandalabs q1 report: Trojans account for 80malware infections, set new record. Online. Panda Security. Retrieved 28 July 2014. [Online]. Available: http://press.pandasecurity.com/news/pandalabs-q1-report-trojans-account-for-80-of-malware-infections-set-new-record/

[16] K. Stevens and D. Jackson, "Zeus banking trojan report," *Atlanta, DELL Secureworks. http://www. secureworks. com/research/threats/zeus*, 2010, retrieved: July, 2014.

[17] C. Wilson, "Botnets, cybercrime, and cyberterrorism: Vulnerabilities and policy issues for congress." DTIC Document, 2008.

[18] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," Tech. Rep., 2011.

[19] S. Cherry, "How stuxnet is rewriting the cyberterrorism playbook," *IEEE Spectrum. http://spectrum. ieee. org/podcast/telecom/security/how-stuxnet-is-rewriting-the-cyberterrorism-playbook*, 2012.

[20] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[21] Awio Web Services LLC. (2012, November) W3counter - global web stats. Retrieved: July, 2014. [Online]. Available: http://www.w3counter.com/globalstats.php

[22] B. Krebs. (2012, October) Service sells access to fortune 500 firms. Online. Retrieved: July, 2014. [Online]. Available: https://krebsonsecurity.com/2012/10/service-sells-access-to-fortune-500-firms/

[23] VMWare Inc. (2013, October) Vmsa-2013-0006.1 vmware security updates for vcenter server. Online. VMWare Inc. Retrieved: July, 2014. [Online]. Available: https://www.vmware.com/security/advisories/VMSA-2013-0006

[24] D. Goldman, "Shodan: The scariest search engine on the internet," *Webseite, Stand*, pp. 01–21, 2014.

[25] R. Hund, C. Willems, and T. Holz, "Practical timing side channel attacks against kernel space aslr," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 191–205.

[26] T. Wang, K. Lu, L. Lu, S. Chung, and W. Lee, "Jekyll on ios: when benign apps become evil," in *Proceedings of the 22nd USENIX conference on Security*. USENIX Association, 2013, pp. 559–572.

[27] S. Röttger, "Malicious code execution prevention through function pointer protection," 2013.

[28] A. Cugliari, L. Part, M. Graziano, and W. Part, "Smashing the stack in 2010," *no. July*, pp. 1–73, 2010.

[29] N. Stojanovski, M. Gusev, D. Gligoroski, and S. Knapskog, "Bypassing data execution prevention on microsoftwindows xp sp2," in *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*. IEEE, 2007, pp. 1222–1226.

[30] V. Katoch. Whitepaper on bypassing aslr/dep. Online. Secfence Technologies. Retrieved: July, 2014. [Online]. Available: http://www.exploit-db.com/wp-content/themes/exploit/docs/17914.pdf

 ISBN: 978-1-61208-368-1

[31] H. Marco-Gisbert and I. Ripoll, “Preventing brute force attacks against stack canary protection on networking servers,” in *Network Computing and Applications (NCA), 2013 12th IEEE International Symposium on*. IEEE, 2013, pp. 243–250.

[32] Net Applications. (2014, April) Desktop operating system market share. Online. Net Applications. Retrieved: July. 2014. [Online]. Available: http://www.netmarketshare.com/