Interested in learning more about security?

SANS Institute InfoSec Reading Room

Worms as Attack Vectors: Theory, Threats, and Defenses

Self-replicating, self-propagating, malicious programs (worms) are described in the context of being likely attack mechanisms for a variety of illicit or illegal activities. A brief discussion of what constitutes a typical worm is given, along with a brief history of worms, reasons they may be released, and who might gain from their use. A proposal for future worms is presented. Finally, current and future (proposed) defenses are presented and discussed in light of potential new threats.

Worms as Attack Vectors: Theory, Threats, and Defenses

A Practical Assignment, submitted in partial requirement for GSEC certification (GIAC Security Essentials Certification [GSEC], Version 1.4b, Option 1)

Matthew Todd, Ph.D.

January 31, 2003

Abstract

Self-replicating, self-propagating, malicious programs (worms) are described in the context of being likely attack mechanisms for a variety of illicit or illegal activities. A brief discussion of what constitutes a typical worm is given, along with a brief history of worms, reasons they may be released, and who might gain from their use. A proposal for future worms is presented. Finally, current and future (proposed) defenses are presented and discussed in light of potential new threats.

Contents

1 Introduction

Self-replicating, malicious software has been a topic of interest ever since Ken Thompson's discussion of placing self-replicating code within a C compiler in 1984.[1] The first known incidence of a self-replicating, self-propagating program occurred in 1988 with the Morris worm, which had, at the time, a devastating effect on the early Internet.[2,3] Since that time, numerous worms have been observed in the wild. At present, the concept of the worm (in terms of the Internet) is so well known that Merriam Webster defines the term as "a usually small self-contained computer program that invades computers on a network and usually performs a malicious action."[4]

Worms, in their most basic form, present a fascinating computational challenge. Much like biological viruses, worms propagate by infecting a host and causing the host to create images of the worm to infect other hosts. Similarly, worms take advantage of weaknesses in the hosts' defenses to propagate. In many cases, simply propagating has detrimental side effects – much like the fatal effects of viruses that cause AIDS or Ebola, worms can damage or shut down hosts or networks simply by how they propagate (examples include the Morris worm[5] and the very recent MS-SQL Server Worm[6]). Unfortunately, worms can go beyond simple replication, and can contain payloads designed with specific, malicious intent such as back doors, DDoS (distributed denial of service) tools, drive erasers, or even espionage tools.[7]

Scores of tools have been developed to fight worms and their close cousins, computer viruses. Unfortunately, as observed by George Smith, "while every warm-blooded living thing has an immune system for fighting invaders, …silicon immunology – despite outbursts of unwarranted ebullience – remains only awkwardly workable."[8]

Recently, SANS asked a number of security experts to opine on the trends for security in 2003.[9] Their predictions portend a potentially bleak future on the Internet. Among other experts, Schneier predicts that the next big security trend will be "Real crime. On the Internet… Just as Willie Sutton robbed banks because 'that's where the money is,' modern criminals will attack computer networks."[10] Murray predicts that "small improvements in software quality will be overwhelmed by increases in software… We will continue to try and patch and fix our way to security; we will continue to fail."[11] Spafford predicts that "we will see destructive political cyber attacks."[12] All indications seem to point towards an increase in malicious activity on the Internet, made possible by the continuing presence of numerous vulnerable systems and processes. Worms are poised to take advantage of these vulnerabilities to provide opportunities for such malicious activity.

2 Worms

Worms present an effective, automated means towards malicious activity on the Internet. They can be used to identify and take advantage of attack vectors into networks and hosts, streamlining the access necessary to perpetuate crime or other illicit activity. In this section, we discuss the essential components of a worm, give a brief history of significant worms, and discuss the potential sources (and reasons) for worms.

It is important to briefly describe the relationship between worms and viruses. Commonly, they are differentiated by their means of propagation: worms are self-propagating, whereas viruses require some form of human interaction. There are several noted cases, however, where a malicious piece of software may be worm-like in some situations, and virus-like in others (see examples of "mail worms," below). In the context of this paper, any piece of software capable of self-replication in some form is discussed in the context of being a worm (regardless of its title).

2.1 Components of Worms

Two excellent sources describe the essential components of worms. The first, "I don't think I really love you, or writing internet worms for fun and profit," by Zalewski, describes some general characteristics of worms, and takes the reader step-by-step through the creation of a worm.[13] The second, "The Future of Internet Worms," by Nazario, Anderson, Wash, and Connelly, gives another view of the essential components of a worm.[14] Nazario, et al., also defines some useful terms that will be used below, including a "worm network" ("a network of systems which have been compromised by a particular worm") and a "node" (any single host that has been infected by the worm).[15] Discussed in these two papers, a worm consists of the (often overlapping) components described (briefly) below. (The interested reader is encouraged to read Zalewski and Nazario, et al., for more complete discussions.) The first four components are requirements for a worm to function; the subsequent components represent more sophisticated components which make worms more virulent, more difficult to eradicate, or even more dangerous.

Not listed among the components is the purpose of a worm (perhaps more a meta-component than an actual component). All worms have some reason for being. This is discussed under payloads as well as under Why Worms?, below.

2.1.1 Autonomy

At its core (and by definition), a worm is an autonomous entity. Once released, it is designed to function and propagate without human interaction.

2.1.2 Replicability

A worm is designed to replicate itself on other hosts (also by definition). It may not necessarily replicate itself in an identical form (see polymorphism, below). Replication may occur via simplistic means (e.g., copying files from a source to a

destination) or via more elegant means (see Thomson's self-replicating code discussion[16]).

2.1.3 Reconnaissance Capabilities

A worm must be able to identify potential targets for replication.[17] To do so, a worm typically has an exploit database, consisting of known vulnerabilities for known software or hardware systems. The worm commonly uses network components within its host node to send packets to a potential victim to determine the presence of known vulnerabilities.

Not all worms have specific reconnaissance capabilities, per se. The Melissa macro virus acts much like a worm to some systems, where email browsers were configured to automatically open attachments (one might argue that some human interaction was necessary at some point).[18] However, as a worm, Melissa does not actively search for new nodes by looking for known vulnerabilities. It simply sends copies of itself to the first 50 entries in all available Microsoft Outlook MAPI address books, regardless of the type of destination. Thus, the Melissa-as-worm spreads without itself knowing of the presence of a known vulnerability at the target. Other so-called "mail worms" include the Love Letter worm,[19] VBS/OnTheFly (Anna Kournikova),[20] and SirCam.[21]

2.1.4 Attack Capabilities

Once a potential victim node is identified, a worm must have the means to take advantage of the known vulnerability to infect the host. This often consists of two steps. First, the worm makes use of a vulnerability to gain access to the host. Some examples of entry paths include buffer overflow vulnerabilities in software, as in the case of Code Red[22] or the recent SQL 2000 worm,[23] web server vulnerabilities such as directory traversal, as in the case of the sadmind/IIS Worm,[24] and email systems (see above).

Once the worm has gained access, it causes code to be executed to establish itself at the host and proceed with further attacks. Often, this involves the use of root kits to elevate privilege (see, for example, a detailed analysis of the Code Red II worm at eEye[25]), but it may simply use a well-established tool like an email browser (as in the case of Melissa[26] or Love Letter[27]). (It should be pointed out that email worms such as those mentioned generally do more than simply propagate via email – see payloads, below.)

These first four components must exist (with the caveats mentioned) for a worm to function. The following capabilities, known to exist in present-day worms, increase the sophistication of a worm, and subsequently its effectiveness.

2.1.5 Multiple Attack Capabilities

Rather than one possible attack vector, a sophisticated worm makes use of knowledge about a variety of vulnerabilities to attack multiple different kinds of systems. For example, the Morris worm made use of known vulnerabilities both in Sendmail on Unix systems and finger on VAX systems.[28] Nimda spread via

email, file shares, IIS vulnerabilities, and even back doors left "open" by Code Red II and sadmind/IIS.[29]

Similarly, a sophisticated worm can exist on multiple different architectures: for example, the sadmind/IIS worm exists on both Windows and Solaris servers[30] (although sadmind/IIS does not propagate from infected IIS servers).

2.1.6 Defense

A well-crafted worm has some means to avoid detection. Many worms use root kits to avoid detection, making use of existing tools to mask worm processes as critical system processes (as in the case of the ExploreZip Trojan Horse[31]), or even substituting new versions of critical system tools (as was the case in SirCam[32]). Some worms attempt to foil anti-virus/anti-worm tools: the Magistr worm (variant b) attempts to disable ZoneAlarm (unsuccessfully),[33] while W32/Goner attempts to disable and delete many anti-virus tools from the infected host.[34] Similarly, a properly crafted worm will act to protect the worm network (see communication, below), by making the act of detecting parents or children (the infector of a node or those nodes infected by a node, respectively) difficult. For example, the Tribe FloodNet 2000 (TFN2K) tool, introduced in December, 1999, makes use of "features to confuse attempts to locate other nodes in a TFN2K network by sending 'decoy' packets."[35] (Such a tool might be used as the payload of a worm; see payloads, below.)

2.1.7 Command Interface

Sophisticated worms provide command interfaces for subsequent action. For example, the Apache/mod_ssl Worm uses a UDP communications channel to "share information on other infected systems as well as attack instructions" (as part of a DDoS network).[36]

2.1.8 Polymorphism

Similar to defense, above, well-designed worms avoid capture and termination by being polymorphic – that is, assuming a variety of forms. Magistr contains a polymorphic engine to change the structure of its code to help avoid both detection and debugging.[37]

2.2 Virulence vs. Payloads

The damage caused by a worm can be measured by two key components: the virulence and the payload. Worms can be harmful without being virulent (not many examples exist, but the potential is there) or without having a payload (as in the case of the recent SQL worm[38]).

2.2.1 Virulence

Virulence is a measure of how rapidly a worm spreads. Especially virulent worms either use a variety of vulnerabilities to spread (increasing the variety, and therefore the number, of potential victims), or compromise especially popular (or common) software and systems with unpatched vulnerabilities. Nimda is an

example of the former, spreading via email, network shares, web sites, and CodeRed II back doors.[39] Email worms such as Melissa,[40] KAK,[41] and VBS/OnTheFly (Anna Kournikova)[42] were especially virulent because they used a popular email tool (Microsoft Outlook). The latter also preyed upon users' interest in a popular tennis star to increase the likelihood of being executed.

Virulence by itself typically affects resources such as network bandwidth, router CPU/memory, or email server availability. In extreme cases, extremely virulent worms have caused or forced the shutdown of critical systems (e.g., email gateways or core routers).

2.2.2 Payloads

The payload is the portion of the worm not necessarily used for propagation. It can be a directly malicious action, such as deleting or forwarding files (as in the case of Magistr[43] and SirCam[44]) or "tagging" of a site by altering web server pages (as in the case of Ramen[45]), installation of a back door for future action (as in the case of CodeRed II[46]), or installation of a keystroke logger (as in the case of BadTrans[47]).

Not all worms have payloads, instead existing only to propagate.

2.3 A Brief History of Worms

The following is a by no means exhaustive history of worms, including some proof-of-concept worms not actually seen in the wild.

1988 The first known instance of a worm seen in the wild, the **Morris worm** was released on 11/2/88.[48]

1989-98 Relatively little happened during this period, with two notable exceptions. In 1992, **DAME** (Dark Avenger Mutation Engine) was created as the first toolkit for making a virus polymorphic.[49] In 1998, **Back Orifice** was developed as a ready-to-use back door for Windows 95 and 98.[50] It was developed by the Cult of the Dead Cow (cDc, http://www.cultdeadcow.com).

1999 **Melissa**, a Word 97 macro virus that spread via email, was first observed in the wild on 3/26/99.[51] It was extremely virulent, but required that a user open an attachment to execute. **Bubbleboy**, released as a proof-of-concept later that year, was a VBScript virus that only required that a message be previewed to execute.[52] Bubbleboy was never actually observed in the wild. Later, **KAK** used the Bubbleboy concept and was extremely virulent.[53]

2000 The **911** worm (4/4/00) spread via unprotected Win 98 shares, specifically for the purpose of a denial of service attack against the 911 system (thankfully, it was ineffective).[54] The **Love Letter** virus/email worm (5/4) was extremely virulent, as it spread via "electronic mail, Windows file sharing, IRC, USENET news, and possibly via web pages."[55]

2001 2001 was an extremely busy year. **Ramen** (1/17/01) attacked Red Hat Linux installations.[56] **AnnaKournikova/OnTheFly** (2/12) was another email worm that took advantage of the popularity of a public figure – effectively, a "social attack."[57] **Gnuman** (2/26) was the first worm to spread via a peer-to-peer network, in this case the Gnutella network.[58] **sadmind/IIS** (5/8) was the first worm to act on both Windows and Solaris.[59] **Mac.Simpson** (June) was the first Applescript worm, and spread via email.[60] **PeachyPDF** (8/7) was the first PDF worm. It required a full version of Acrobat (effectively limiting its virulence), and spread via email.[61] 2001 also saw some spectacularly virulent, potentially dangerous, and sophisticated worms. **Magistr** (3/13) was polymorphic, and emailed copies of randomly selected (potentially confidential) documents to email addresses found on the host.[62] **SirCam** (7/25) contained its own SMTP engine, and potentially revealed or deleted files on Windows systems.[63] **CodeRed** (7/19) spread via IIS, and attempted to perform a DDoS attack against www.whitehouse.gov.[64] **CodeRed II** (8/4) also spread via IIS, and created a Trojan into the host system.[65] **Nimda** (9/25) spread via a variety of means, making it extremely virulent.[66] **BadTrans** (11/27) was an email worm that logged keystrokes on affected hosts.[67]

2002 **Donut** (1/9/02) was the first known .NET worm.[68] **SQLSpider** (January) infected Microsoft SQL servers and stole NT passwords and network information.[69] **Benjamin** (5/19) was the first worm known to spread via the KaZaa peer-to-peer network.[70] **Scalper** (6/28) was a proof-of-concept worm (not apparently observed in the wild) that attacked Apache running on the FreeBSD operating system. It was designed to create a "flood net," and was capable of flooding via TCP, UDP, DNS, and email.[71]

2003 **SQL worm** (1/25) attacks Microsoft SQL servers. It is purely memory-resident, and can thus evade most anti-virus scanners.[72]

2.4 Why Worms?

Worms began as an interesting idea to create a self-replicating program, and have become an extremely effective means to attack systems. The potential reasons for worms vary from "because it is possible" all the way to warfare and espionage. Some motivations for the creation of worms are outlined below. Understanding the motivations as well as the sources of the worms can help security professionals plan defenses.

A key point is this: worms present efficient means to identify attack vectors into systems, attack the systems themselves, and streamline (later) access to those systems for a variety of purposes. With that in mind, worms may provide one of the most efficient means for system compromise, superceding other, more manual methods. Thus, any potential reason for compromising a computer system can be applied to a worm.

2.4.1 "Just Because"

Some worms simply exist as a proof of concept, as in the case of the Morris worm.[73] There is no particular malicious intent, although the worm may have damaging effects.

2.4.2 Fame

Similar to "just because," some worm authors may create the worm to give themselves some fame or notoriety. Many worms include some sort of signature, such as Melissa's reference to Kwyjibo,[74] or Ramen's reference to the "Ramen crew."[75] Worms also may be used as a means for bragging rights among competing hackers. Ramen, for example, closed the holes used to gain access on compromised systems, keeping other hackers from using the same exploit, and logged compromised systems by sending email to Yahoo! and Hotmail accounts.[76]

2.4.3 Crime

Schneier predicts that the next big trend will be "real crime. On the Internet."[77] In Secrets and Lies, Schneier further describes the prospects for crime on the Internet by relating it to the world of bricks and mortar.[78] The same kinds of crimes one might find committed in the real world can easily be extended to the Internet, where real commerce occurs every day. Robbing a convenience store may net a few hundred dollars, whereas "robbing" an online merchant could net thousands of credit cards. Fraud, blackmail, and thievery – all are possible on a grand scale on the Internet. A worm could be designed specifically to work its way into a network with the sole purpose of committing a crime.

2.4.4 Politics/Religion

The Internet provides a ready means of expression to political and religious groups. In extreme cases, particularly motivated political or religious groups could use a worm to spread a particular message or slogan. As the Internet does not generally pose any barriers between political or geographic areas, it would be much easier to have a vast audience through the use of a worm on the Internet than via any other, more traditional broadcast medium. (Such a worm would probably also generate interest in the press, managing to provide additional exposure for free.)

Note that political or religious groups may also wish to make their presence known by more drastic means; see sabotage, below.

2.4.5 Sabotage

Any number of groups may wish to use worms to sabotage another party. Governments could use a specially targeted worm to disable the computer infrastructure of an enemy, to spread disinformation, or otherwise confound or disrupt the activity of an enemy government. Political or religious groups may similarly attack an enemy, perhaps as an act of terrorism. Corporations may act

against competing organizations to hinder operations. Individuals (e.g., disgruntled employees) may even act to damage a company's assets, infrastructure, or image. Worms are particularly adept at spreading quickly and causing disruptions in networks and systems.

2.4.6 Intelligence Gathering/Espionage

Governments, companies, and other groups wishing to gain access to confidential or secret data often use computer hacking techniques to gain illicit access to databases or other sources of information. Individuals have stolen personal data for purposes of identity fraud. Such actions can be automated through the use of worms (as has been demonstrated in the case of SirCam,[79] Magistr,[80] and BadTrans[81]).

3 Worms of the Future

Worms can be expected to become more virulent, more sophisticated, and more flexible. Below, we discuss some possible avenues for future development of worms, first describing enhancements to existing components as well as some not previously mentioned. The sophistication of worms is only limited by network bandwidth, memory and CPU on a victim node, and the imagination of the authors. Unfortunately, all are steadily increasing.

3.1.1 Reconnaissance Capabilities

Identifying potential targets takes time, especially when a random scanning method is used (linear scans of sequential ports or IP addresses are blocked by most firewalls, and are thus extremely inefficient for the spread of a worm). At the same time, the act of reconnaissance can expose the worm to network listening devices.

Several means of reconnaissance have been proposed to greatly enhance the speed of propagation of a potential worm. The Warhol Worm concept describes how a worm's initial attack can be coordinated by gathering a large number of potentially susceptible hosts prior to attacking (an initial population), and then using a partitioning technique to limit the range of hosts a given node would scan and attack.[82] The Flash Worm elaborated on the Warhol Worm concept by proposing that a potentially complete list of vulnerable hosts could be obtained, and a massive, simultaneous attack could be launched.[83] Curious Yellow was proposed as a worm with an efficient, coordinated attack mechanism, using a distributed hash table design.[84]

Note that mass-email lists can be easily obtained, presenting a substantial initial population for any worm with email-based capabilities.

Particularly intelligent worms could gather intelligence before attempting to infect additional nodes. CodeRed II is one example of this, specifically identifying and attacking a local network:

> Instead of searching only randomly selected addresses, Code Red 2 preferentially probed for machines on the same subnet and nearby subnets. As a result, once a single machine within a corporate firewall was infected, it would quickly probe virtually every machine within the firewall and since it was attacking an on-by-default service, Code Red 2 quickly infested entire corporate networks.[85]

As an extension of this, a worm may watch network connections to and from an infected node, gathering information about local and adjacent networks, before attempting to compromise additional hosts (see, for example, Nazio, et al.[86]). Thus, if such a worm managed to infect a web server resident within a DMZ, it might gain knowledge of a back-end database on a protected network by watching how the web server communicates. It could then potentially use a known vulnerability to attack the database server, masquerading as the web server and potentially eluding discovery.

3.1.2 Multiple Attack Capabilities

Sophisticated worms will have a large cache of known vulnerabilities for both reconnaissance and attack, and will have some means of obtaining new vulnerability signatures to use. Thus, a worm might manage to compromise one system and stealthily wait for a new vulnerability to come available in order to progress further. New vulnerabilities may be passed via some communications mechanism (see below), or may be made available in a specific location for pickup (a "drop box").

Newer attack mechanisms may make use of old ideas, e.g., communications hijacking (as demonstrated by Mitnick[87]). A worm might substitute packets within a stream or otherwise masquerade as an expected source to infect a new host.

3.1.3 Defense

Worms may use a variety of means to defend themselves against detection or removal in addition to those described above. Worms could use a network smokescreen during a coordinated attack, hiding the true nature of an attack under a barrage of apparent "script-kiddie" activity that overwhelms the responses of a network or IT group.

Worms could use encryption to authenticate communications between nodes of a worm network, ensuring that updates came from proper (and not disruptive) sources. Such encryption could also be used to ensure that one node does not reveal other nodes within the network.[88]

Worms could create redundant networks. When crossing a network boundary (e.g., in to a private network, as evidenced by a difference between an apparent, Internet-facing IP and the local IP address of the node), a worm could establish "lieutenant" nodes as redundant points for communications up or down the worm network. Then, child nodes could communicate with any of the lieutenants for updates, and the failure of any one lieutenant would not disrupt the communications channel.

3.1.4 Command Interface and Communication

Given the premise that worms will be used for illicit activities, future worms will likely have sophisticated command interfaces. These interfaces could be used to communicate orders ("Attack at dawn!"), distribute new vulnerability signatures, or distribute new modules (see below). Command interfaces will likely also make use of secure communications mechanisms to avoid compromise. Worms could make use of optimized and redundant communications mechanisms to ensure efficient, fault tolerant communication between nodes.

3.1.5 Polymorphism/Adaptation/Expansion

Future worms will not only be polymorphic to avoid detection, they may also be capable of adapting to a hostile environment (e.g., by mimicking existing applications) and could be designed in a modular fashion so as to easily receive and incorporate new functions from the worm network.

3.1.6 Payload

Payloads may be carried by a worm, or may be distributed at a later time over the worm network. Regardless, payloads will be the key to a successful targeted worm attack. Nodes of a worm network intended for thievery may listen intently to network communications in search of personal information, passwords, or bank codes, or they may actively seek out databases and probe directly for information. This information could be passed to a parent node that may then coordinate a response (e.g., one node may be responsible for intelligence gathering, while another will cause the bank to transfer $1M to an offshore account). Worm units dedicated to sabotage might contain instructions for changing voltages or temperatures on specific devices, harming them physically, or they may act in a more subtle fashion by injecting misinformation into communication streams between an enemy's hosts.

3.1.7 Intelligence

Many of the proposed features of future worms are akin to artificially intelligent systems. Given the ever-increasing capabilities of systems and networks, it is not entirely outside of the realm of possibility to see truly self-adapting worms in the future. Worm nodes may be adaptable and intelligent by themselves, or in concert with other nodes within the worm network.

3.1.8 One Possible Scenario

Consider the following possible scenario: A worm much like the recent SQL worm is launched. It is extremely virulent, causes a substantial amount of network traffic, and causes disruptions to networks, but has no apparent payload. It rapidly affects thousands of vulnerable systems, and security professionals quickly identify the vulnerability. Systems administrators apply the patch, shore up their border controls, reboot their systems, and finally rest, assured that they have fixed the problem. Security news groups discuss the worm as "another example of what could have been a spectacular problem."

Meanwhile, it turns out that the publicly observed worm is really a cover for another, much more targeted, worm that takes advantage of the same vulnerability. Unlike the "big" worm, the much lower-profile worm carries a substantial payload, consisting of sophisticated communications packages and intelligence gathering routines. Also unlike the publicly observed worm, the secret worm does not vanish with a patch and a reboot, and it is sophisticated enough to seek out alternate means of communications should firewalls be altered. It quietly sits and gathers confidential information and reports up the worm network via secure channels.

4 Defense – Present and Future

Worms take advantage of known vulnerabilities to propagate. As new technologies or extensions to existing technology arrive, new vulnerabilities are exposed – it is simply not possible to account for all vulnerabilities in a system prior to its release. At the same time, IT departments lack the skills or resources to stay on top of all known vulnerabilities (as is demonstrated with every new worm attack), and users limit the effectiveness of defenses by making extensive use of web browsers (forcing HTTP access across firewalls), instant messaging systems, and peer-to-peer networks. To mitigate risk, organizations must make use of the concept of Defense in Depth to build up layers of security.[89] It is critical to remember that no defense is foolproof, and that vulnerabilities will always exist.

A sampling of technologies or programs especially associated with worms is presented below, including some proposals for emergent technologies. The defenses are described in terms of being proactive or reactive – obviously, proactive technologies typically provide better defenses than reactive. Where appropriate, sample technologies are provided, but the list is by no means comprehensive, and the samples do not necessarily reflect any preference or endorsement.

4.1 Patches

A diligent program of monitoring vendor patch releases and applying relevant patches in a timely manner is a substantial, proactive defense. Most worms attack vulnerabilities that have been known about and patched for some time. Unfortunately, many IT departments lack the skill or resources to keep abreast of all relevant patches, and important updates are ignored or overlooked.

New patches are made available via a variety of channels, but most commonly via Internet sites. Most vendors supply security patches for their products for free. In some cases, software packages can proactively notify the user of newly available updates; see, for example, Microsoft's "Automatic Updates" tool available in Windows 2000 and XP, as well as the on-line English version at http://v4.windowsupdate.microsoft.com/en/default.asp, or Apple's "Software Update" (on-line at http://www.info.apple.com/support/downloads.html). Most operating system vendors have email distributions for patch notification.

Numerous email distributions are available that discuss newly discovered vulnerabilities even before they are patched. For example, Security Focus maintains the "bugtraq" mailing list (http://www.securityfocus.com/popups/forums/bugtraq/intro.shtml).

4.2 Firewalls and Routers

Packet-filtering firewalls and routers provide proactive defense against network-based attacks by promoting a "least privilege" approach, providing access to only those packets that are specifically approved. Stateful-inspection firewalls (e.g., Check Point, http://www.checkpoint.com) are more sophisticated than simple packet filters in that they examine packets in the context of other packets or other connections, ensuring that packets pass only if they are appropriate for that particular TCP connection, or if their contents match approved characteristics (like a session ID). Unfortunately, attacks based on buffer overflows (for example) can appear to both types of firewalls as "expected" traffic.

Proxying firewalls (e.g., Symantec's Velociraptor, http://www.symantec.com) make use of application proxies to avoid situations like buffer overflows. Unfortunately, such devices don't typically support secure protocols such as HTTPS. Proxying firewalls also do not protect against logical failures (e.g., parameter substitution).

Other proxy solutions exist that can be tailored to an application as a proactive measure. For example, AppShield (from Sanctum, Inc., http://www.sanctuminc.com) provides an application proxy that makes use of specific user-defined rules to enforce application logic.

4.3 Network-based IDS

Network-based intrusion detection systems (IDS) are either proactive or reactive, depending on their configuration. Generally, network based IDS have a large number of false positives, and are therefore not designed to automatically block attacks, but instead notify upon particular conditions. They typically make use of published rules that look for intrusion signatures to detect particular attacks (e.g., SNORT at http://www.snort.org) or look for trends or violations of policies (e.g., StealthWatch at http://www.lancope.com).

4.4 Host-based IDS

4.4.1 Checksum-based Detection

Checksum-based host IDS tools are proactive defenses against worms that make changes to files on disk. Such IDS tools create a database of file signatures (checksums) for critical system and application files, and compare this database against existing files to ensure that they have not changed. Unfortunately, checksum-based tools suffer from two key limitations: (a) they cannot react to memory resident worms (e.g., the recent SQL worm), as such worms do not alter any files on the disk; (b) they are not continuously active, and can only react to an intrusion based on the scanning schedule. Still, checksum-

based tools do not rely on known attack signatures, and they can easily detect inappropriate changes to critical systems files or environment settings. See, for example, Tripwire (http://www.tripwire.com).

4.4.2 Signature-based Detection

Anti-virus tools are the best-known signature-based tools. They make use of published attack signatures to quickly detect, block, and clean up after worms and viruses. Unfortunately, they are generally reactive tools, based on known attacks. In many cases, heuristic engines are available, allowing tools to look for malicious activity that resembles known signatures (aiding in the discovery of polymorphic worms or viruses). See, for example, McAfee VirusScan (http://www.mcaffee.com) or Symantec AntiVirus (http://www.symantec.com).

4.5 Host-based IPS

IPS, or intrusion prevention systems, are relatively new but powerful, proactive defenses. Software tools such as Entercept (http://www.entercept.com) ensure that system and application calls adhere to strict policies of accepted behavior. Attempted activity outside of defined rule sets (e.g., elevation of privilege, modification of systems files, etc.) is blocked outright, and an alert is given. In some ways, IPS tools provide a "trusted operating system" (TOS). A TOS provides a hardened kernel with fine-grained access control (e.g., TrustedBSD at http://www.trustedbsd.org).

The Trusted Computing Platform Alliance (TCPA, http://www.trustedcomputing.org) is an "open alliance… formed to work on creating a new computing platform for the next century that will provide for improved trust in the PC platform."[90] One of the proposed technologies is a hardware co-processor and dedicated memory that are capable of performing secured functions, such as performing a cryptographic hash using a hardware-specific key. This co-processor could mediate operating system activities and only allow actions if the kernel had not been changed since a hash was established (even memory-resident routines could be fingerprinted and monitored). Because the hardware is separate from that running the (main) operating system, it is protected from harm, and can act as an "external monitor."

4.6 Tarpits and Honeynets

Tarpits and Honeynets, while not strictly defenses against worms, can be used in an overall defensive posture. Tarpits (see, for example, http://www.hackbusters.net/LaBrea) are blocks of IP addresses that "pretend" to be vulnerable systems, but which in fact do nothing. They can be used to hinder the progress of a worm by making it attack without the possibility of harm. Honeynets (see http://project.honeynet.org) are groups of systems (servers, routers, and firewalls) that are used to monitor and research real world attacks. In both cases, systems are set up as separate from "production" systems, so that any traffic hitting these servers is, by default, unexpected.

Neither tarpits nor Honeynets can stop the action of a worm; however, when used in conjunction with a centralized monitoring and management tool (see below), they may act as detectors and help to quickly analyze the nature of the attack (by isolating the attack from other sorts of traffic).

4.7 Centralized Monitoring and Management

Centralized monitoring tools take multiple events from a variety of sources, and attempt to determine if there is any significant correlation. For example, a centralized monitor might collect events from two devices on opposite sides of a firewall, each running SNORT, and determine that an attack was occurring based on traffic patterns across the firewall. (See, for example, Symantec's ManTrap, http://www.symantec.com.) Some even make use of a "network honeypot" to detect anomalous traffic (e.g., NetScreen-IDP, http://www.netscreen.com).

Future centralized monitoring and management tools might correlate host-based as well as network-based events to detect anomalous activities. Co-processors on hosts could communicate securely with a central management console to report on activity, and receive instructions to block certain events.

4.8 Social Awareness

Raising individual user awareness of vulnerabilities and risks presents a unique but important challenge. If all users proactively patched their systems, updated their anti-virus tools, and installed desktop firewalls, worms would have a much more difficult time propagating. Unfortunately, the popularity of the Internet combined with consumers' (regrettably naïve) expectations that computers be as simple and secure as a television "out of the box" means that home (and even office) users will often have vulnerable (and more and more powerful) computers available for worms to use as nodes.

5 Trends

In the SANS newsletter mentioned in the introduction, Rob Clyde, VP & Chief Technology Officer, Symantec Corporation, believes that "we will see a rise in more 'professional' types of attackers, targeting specific, crucial online systems and posing great potential dangers."[91] Thankfully, he and other experts predict that there will be a corresponding increase in the sophistication of defenses. For example, Clyde predicts that we will "see the emergence and initial deployment of … new proactive technologies."[92] Tom Noonan, Chairman, President, and CEO, Internet Security Systems, predicts that

> Intrusion detection technology [will advance] into intrusion protection. This technology will combine pattern matching, several layers of protocol analysis, pre-emptive behavioral inspection, anomaly detection and firewall blocking to not only detect online threats, but also to block them altogether.[93]

He also believes that "Individual protection agents will protect the enterprise systems from the entire spectrum of Internet threats…"[94] Gil Shwed, Chairman and Chief Executive Officer, Check Point Software Technologies Ltd., proposes that "Technology to consistently manage and enforce security policies must be deployed both in front of and behind the perimeter to secure all access points," and further states that new correlation technologies are "essential."[95]

Worms will continue to exist, taking advantage of ever emerging vulnerabilities in new and dangerous ways. Thankfully, we can expect that present and emerging defenses, when properly administered, will guard against the threat.

6 Notes

[1] Thompson, Ken. "Reflections on Trusting Trust." September 1995. URL: http://www.acm.org/classics/sep95/ (29 January 2003)

[2] Weaver, Nicholas. "A Brief History of The Worm." 26 November 2001. URL: http://online.securityfocus.com/infocus/1515 (29 January 2003)

[3] "Virus History" URL: http://www.cknow.com/vtutor/vthistory.htm (29 January 2003)

[4] Merriam-Webster OnLine Dictionary. http://www.m-w.com (29 January 2003), definition 6 for "worm"

[5] See notes 2, 3.

[6] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2003-04 MS-SQL Server Worm." 27 January 2003. URL: http://www.cert.org/advisories/CA-2003-04.html (29 January 2003)

[7] See note 2.

[8] Smith, George. "Lessons from the Laboratory." 06 January 2003. URL: http://online.securityfocus.com/columnists/133 (29 January 2003)

[9] SANS NewsBites, Bonus Issue (email sent from NewsBites@sans.org). "Experts Predict the Future of Computer Security." 13 December 2002

[10] See note 9.

[11] See note 9.

[12] See note 9.

[13] Zalewski, Michal. "I don't think I really love you, or writing internet worms for fun and profit." © 1998-2000. URL: http://lcamtuf.coredump.cx/worm.txt (29 January 2003)

[14] Nazario, Jose; Anderson, Jeremy; Wash, Rick; Connelly, Chris. "The Future of Internet Worms." 20 July 2001. URL: http://www.crimelabs.net/docs/worms/worm.pdf (29 January 2003)

[15] See note 14, pg. 3.

[16] See note 1.

[17] See note 14, pg. 5.

[18] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-1999-04 Melissa Macro Virus." 31 March 1999. URL: http://www.cert.org/advisories/CA-1999-04.html (29 January 2003)

[19] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2000-04 Love Letter Worm." 09 May 2000. URL: http://www.cert.org/advisories/CA-2000-04.html (29 January 2003)

[20] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2001-03 VBS/OnTheFly (Anna Kournikova) Malicious Code." 13 February 2001. URL: http://www.cert.org/advisories/CA-2001-03.html (29 January 2003)

[21] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2001-22 W32/Sircam Malicious Code." 23 August 2001. URL: http://www.cert.org/advisories/CA-2001-22.html (29 January 2003)

[22] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL." 197 January 2002. URL: http://www.cert.org/advisories/CA-2001-19.html (29 January 2003)

[23] See note 6.

[24] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2001-11 sadmind/IIS Worm." 10 May 2001. URL: http://www.cert.org/advisories/CA-2001-11.html (29 January 2003)

[25] eEye Digital Security. "CodeRedII Worm Analysis" 04 August 2001. URL: http://www.eeye.com/html/Research/Advisories/AL20010804.html (29 January 2003)

[26] See note 18.

[27] See note 19.

[28] See note 14, pg. 10.

[29] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2001-26 Nimda Worm." 25 September 2001. URL: http://www.cert.org/advisories/CA-2001-26.html (29 January 2003)

[30] See note 24.

[31] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-1999-06 ExploreZip Trojan Horse Program." 14 June 1999. URL: http://www.cert.org/advisories/CA-1999-06.html (29 January 2003)

[32] See note 21.

[33] F-Secure Security Information Center. "F-Secure Computer Virus Information Pages: Magistr." 06 September 2001. URL: http://www.f-secure.com/v-descs/magistr.shtml (29 January 2003)

[34] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Incident Note IN-2001-15." 04 December 2001. URL: http://www.cert.org/incident_notes/IN-2001-15.html (29 January 2003)

[35] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-1999-17 Denial-of-Service Tools." 03 March 2000. URL: http://www.cert.org/advisories/CA-1999-17.html (29 January 2003)

[36] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2002-27 Apache/mod_ssl Worm." 11 October 2002. URL: http://www.cert.org/advisories/CA-2002-27.html (29 January 2003)

[37] See note 33.

[38] See note 6.

[39] See note 29.

[40] See note 18.

[41] Symantec. "Symantec Security Response – Wscript.KakWorm." 24 June 2002. URL: "http://www.symantec.com/avcenter/venc/data/wscript.kakworm.html (29 January 2003)

[42] See note 20.

[43] See note 33.

[44] See note 21.

[45] Symantec. "Symantec Security Response –Linux.Ramen.Worm." 15 April 2002. URL: http://service1.symantec.com/sarc/sarc.nsf/html/Linux.Ramen.Worm.html (29 January 2003)

[46] See note 25.

[47] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Incident Note IN-2001-14 (W32/BadTrans Worm)." 27 November 2001. URL: http://www.cert.org/incident_notes/IN-2001-14.html (29 January 2003)

[48] See notes 2, 3.

[49] F-Secure Security Information Center. "F-Secure Computer Virus Information Pages: Cryptlab." http://www.f-secure.com/v-descs/mte.shtml (29 January 2003)

[50] Symantec. "Information on Back Orifice and NetBus." URL: http://www.symantec.com/avcenter/warn/backorifice.html (29 January 2003)

[51] See note 18.

[52] Symantec. "Symantec Security Response – VBS.BubbleBoy." URL: http://www.symantec.com/avcenter/venc/data/vbs.bubbleboy.html(29 January 2003)

[53] See note 41.

[54] Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Incident Note IN-2000-03 (911 Worm)." 04 April 2000. URL: http://www.cert.org/incident_notes/IN-2000-03.html (29 January 2003)

[55] See note 19.

[56] See note 45.

[57] See note 20.

[58] Symantec. "Symantec Security Response – W32.Gnuman.Worm." 15 April 2002. URL: http://www.symantec.com/avcenter/venc/data/w32.gnuman.worm.html (29 January 2003)

[59] See note 24.

[60] Sophos. "Sophos virus analysis: AplS/Simpsons-A." URL: http://www.sophos.com/virusinfo/analyses/aplssimpsonsa.html (29 January 2003)

[61] Symantec. "Symantec Security Response – VBS.PeachyPDF@mm." 15 April 2002. URL: http://www.symantec.com/avcenter/venc/data/vbs.peachypdf@mm.html (29 January 2003)

[62] See note 33.

[63] See note 21.

[64] See note 22.

[65] See note 25.

[66] See note 29.

[67] See note 47.

[68] Proland Software. "Win32/Donut.A Worm." URL: http://www.pspl.com/virus_info/win32/donut.htm (29 January 2003)

[69] Sophos. "Sophos virus analysis: W32/SQLSpider-A." URL: http://www.sophos.com/virusinfo/analyses/w32sqlspidera.html (29 January 2003)

[70] Symantec. "Symantec Security Response – W32.Benjamin.Worm." 22 November 2002. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.benjamin.worm.html (29 January 2003)

[71] Symantec. "Symantec Security Response – FreeBSD.Scalper.Worm." 01 July 2002. URL: http://www.symantec.com/avcenter/venc/data/freebsd.scalper.worm.html (29 January 2003)

[72] See note 6.

[73] See notes 2, 3.

[74] See note 18.

[75] See note 29.

[76] See note 29.

[77] See note 9.

[78] Schneier, Bruce. Secrets and Lies, Digital Security in a Networked World. New York: John Wiley & Sons, Inc., 2000. 15-16.

[79] See note 21.

[80] See note 33.

[81] See note 47.

[82] Weaver, Nicholas C. "Warhol Worms: The Potential for Very Fast Internet Plagues" 13 February 2002. URL: http://www.cs.berkeley.edu/~nweaver/warhol.html, (29 January 2003)

[83] Staniford, Stuart; Grim, Gary; Jonkman, Roelof. "Flash Worms: Thirty Seconds to Infect the Internet." 16 August 2001. URL: http://www.silicondefense.com/flash (29 January 2003)

[84] Wiley, Brandon. "Curious Yellow: The First Coordinated Worm Design." URL: http://blanu.net/curious_yellow.html (29 January 2003)

[85] See note 2.

[86] See note 14, pg. 14.

[87] Shimomura, Tsutomu. "Tsutomu Shimomura's newsgroup posting with technical details of the attack described by Markoff in NYT." 25 January 1995. URL: http://www.gulker.com/ra/hack/tsattack.html (29 January 2003)

[88] See note 14, pg. 16 , and note 84.

[89] VanMeter, Charlene "Defense In Depth: A Primer." 19 February 2001. URL: http://www.sans.org/rr/start/primer.php (29 January 2003)

[90] Trusted Computing Platform Alliance. URL: http://www.trustedcomputing.org (29 January 2003)

[91] See note 9.

[92] See note 9.

[93] See note 9.

[94] See note 9.

[95] See note 9.

7 References

Thompson, Ken. "Reflections on Trusting Trust." September 1995. URL: http://www.acm.org/classics/sep95/ (29 January 2003)

Weaver, Nicholas. "A Brief History of The Worm." 26 November 2001. URL: http://online.securityfocus.com/infocus/1515 (29 January 2003)

Computer Knowledge. "Virus History" URL: http://www.cknow.com/vtutor/vthistory.htm (29 January 2003)

Merriam-Webster OnLine Dictionary. http://www.m-w.com (29 January 2003), definition 6 for "worm"

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2003-04 MS-SQL Server Worm." 27 January 2003. URL: http://www.cert.org/advisories/CA-2003-04.html (29 January 2003)

Smith, George. "Lessons from the Laboratory." 06 January 2003. URL: http://online.securityfocus.com/columnists/133 (29 January 2003)

SANS NewsBites, Bonus Issue (email sent from NewsBites@sans.org). "Experts Predict the Future of Computer Security." 13 December 2002

Zalewski, Michal. "I don't think I really love you, or writing internet worms for fun and profit." © 1998-2000. URL: http://lcamtuf.coredump.cx/worm.txt (29 January 2003)

Nazario, Jose; Anderson, Jeremy; Wash, Rick; Connelly, Chris. "The Future of Internet Worms." 20 July 2001. URL: http://www.crimelabs.net/docs/worms/worm.pdf (29 January 2003)

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-1999-04 Melissa Macro Virus." 31 March 1999. URL: http://www.cert.org/advisories/CA-1999-04.html (29 January 2003)

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2000-04 Love Letter Worm." 09 May 2000. URL: http://www.cert.org/advisories/CA-2000-04.html (29 January 2003)

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2001-03 VBS/OnTheFly (Anna Kournikova) Malicious Code." 13 February 2001. URL: http://www.cert.org/advisories/CA-2001-03.html (29 January 2003)

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2001-22 W32/Sircam Malicious Code." 23 August 2001. URL: http://www.cert.org/advisories/CA-2001-22.html (29 January 2003)

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL." 197 January 2002. URL: http://www.cert.org/advisories/CA-2001-19.html (29 January 2003)

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2001-11 sadmind/IIS Worm." 10 May 2001. URL: http://www.cert.org/advisories/CA-2001-11.html (29 January 2003)

eEye Digital Security. "CodeRedII Worm Analysis" 04 August 2001. URL: http://www.eeye.com/html/Research/Advisories/AL20010804.html (29 January 2003)

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2001-26 Nimda Worm." 25 September 2001. URL: http://www.cert.org/advisories/CA-2001-26.html (29 January 2003)

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-1999-06 ExploreZip Trojan Horse Program." 14 June 1999. URL: http://www.cert.org/advisories/CA-1999-06.html (29 January 2003)

F-Secure Security Information Center. "F-Secure Computer Virus Information Pages: Magistr." 06 September 2001. URL: http://www.f-secure.com/v-descs/magistr.shtml (29 January 2003)

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Incident Note IN-2001-15." 04 December 2001. URL: http://www.cert.org/incident_notes/IN-2001-15.html (29 January 2003)

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-1999-17 Denial-of-Service Tools." 03 March 2000. URL: http://www.cert.org/advisories/CA-1999-17.html (29 January 2003)

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Advisory CA-2002-27 Apache/mod_ssl Worm." 11 October 2002. URL: http://www.cert.org/advisories/CA-2002-27.html (29 January 2003)

Symantec. "Symantec Security Response – Wscript.KakWorm." 24 June 2002. URL: "http://www.symantec.com/avcenter/venc/data/wscript.kakworm.html (29 January 2003)

Symantec. "Symantec Security Response –Linux.Ramen.Worm." 15 April 2002. URL: http://service1.symantec.com/sarc/sarc.nsf/html/Linux.Ramen.Worm.html (29 January 2003)

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Incident Note IN-2001-14 (W32/BadTrans Worm)." 27 November 2001. URL: http://www.cert.org/incident_notes/IN-2001-14.html (29 January 2003)

F-Secure Security Information Center. "F-Secure Computer Virus Information Pages: Cryptlab." http://www.f-secure.com/v-descs/mte.shtml (29 January 2003)

Symantec. "Information on Back Orifice and NetBus." URL: http://www.symantec.com/avcenter/warn/backorifice.html (29 January 2003)

Symantec. "Symantec Security Response – VBS.BubbleBoy." URL: http://www.symantec.com/avcenter/venc/data/vbs.bubbleboy.html (29 January 2003)

Carnegie Mellon Software Engineering Institute CERT® Coordination Center. "CERT® Incident Note IN-2000-03 (911 Worm)." 04 April 2000. URL: http://www.cert.org/incident_notes/IN-2000-03.html (29 January 2003)

Symantec. "Symantec Security Response – W32.Gnuman.Worm." 15 April 2002. URL: http://www.symantec.com/avcenter/venc/data/w32.gnuman.worm.html (29 January 2003)

Sophos. "Sophos virus analysis: AplS/Simpsons-A." URL: http://www.sophos.com/virusinfo/analyses/aplssimpsonsa.html (29 January 2003)

Symantec. "Symantec Security Response – VBS.PeachyPDF@mm." 15 April 2002. URL: http://www.symantec.com/avcenter/venc/data/vbs.peachypdf@mm.html (29 January 2003)

Proland Software. "Win32/Donut.A Worm." URL: http://www.pspl.com/virus_info/win32/donut.htm (29 January 2003)

Sophos. "Sophos virus analysis: W32/SQLSpider-A." URL: http://www.sophos.com/virusinfo/analyses/w32sqlspidera.html (29 January 2003)

Symantec. "Symantec Security Response – W32.Benjamin.Worm." 22 November 2002. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.benjamin.worm.html (29 January 2003)

Symantec. "Symantec Security Response – FreeBSD.Scalper.Worm." 01 July 2002. URL: http://www.symantec.com/avcenter/venc/data/freebsd.scalper.worm.html (29 January 2003)

Schneier, Bruce. Secrets and Lies, Digital Security in a Networked World. New York: John Wiley & Sons, Inc., 2000. 15-16.

Weaver, Nicholas C. "Warhol Worms: The Potential for Very Fast Internet Plagues" 13 February 2002. URL: http://www.cs.berkeley.edu/~nweaver/warhol.html (29 January 2003)

Staniford, Stuart; Grim, Gary; Jonkman, Roelof. "Flash Worms: Thirty Seconds to Infect the Internet." 16 August 2001. URL: http://www.silicondefense.com/flash (29 January 2003)

Wiley, Brandon. "Curious Yellow: The First Coordinated Worm Design." URL: http://blanu.net/curious_yellow.html (29 January 2003)

Shimomura, Tsutomu. "Tsutomu Shimomura's newsgroup posting with technical details of the attack described by Markoff in NYT." 25 January 1995. URL: http://www.gulker.com/ra/hack/tsattack.html (29 January 2003)

VanMeter, Charlene “Defense In Depth: A Primer.” 19 February 2001. URL: http://www.sans.org/rr/start/primer.php (29 January 2003)

Trusted Computing Platform Alliance. URL: http://www.trustedcomputing.org (29 January 2003)

Last Updated: June 23rd, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Cyber Defence Canberra 2017	Canberra, AU	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MDUS	Jun 26, 2017 - Jul 01, 2017	Live Event
SEC564:Red Team Ops	San Diego, CAUS	Jun 29, 2017 - Jun 30, 2017	Live Event
SANS London July 2017	London, GB	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, JP	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS ICS & Energy-Houston 2017	Houston, TXUS	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, SG	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CAUS	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Munich Summer 2017	Munich, DE	Jul 10, 2017 - Jul 15, 2017	Live Event
SANSFIRE 2017	Washington, DCUS	Jul 22, 2017 - Jul 29, 2017	Live Event
Security Awareness Summit & Training 2017	Nashville, TNUS	Jul 31, 2017 - Aug 09, 2017	Live Event
SANS San Antonio 2017	San Antonio, TXUS	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Hyderabad 2017	Hyderabad, IN	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MAUS	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Prague 2017	Prague, CZ	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UTUS	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NYUS	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Chicago 2017	Chicago, ILUS	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Adelaide 2017	Adelaide, AU	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VAUS	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Paris 2017	OnlineFR	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced