CS 526: Information Security - Fall 2017

Following the Morris Worm

Phil Van Every

December 8, 2017

Abstract

On November 2, 1988, a Cornell graduate student named Robert Tappan Morris unleashed one of the first ever computer worms on the fledgling internet. It quickly spread to thousands of connected computers, causing crashes, performance degradation, and panic until it was contained and eradicated. This unprecedented crisis elicited both immediate and long term responses spanning multiple disciplines. The research community responded with the formation of new security emergency diagnosis and response protocols and organizations, including the CERT at CMU. Law enforcement responded by making Robert Morris the first felon convicted under the Computer Fraud and Abuse Act of 1986. The public at large began to appreciate the potential impact of internet security, or a lack thereof.

This paper explores the Morris Worm and its overall impact. It details the events surrounding the Morris Worm crisis and the inner workings of the worm itself. It goes on to trace the worm's influence on cyber security legislation, cyber security research, and other cyber attacks over the past several decades. Finally, it draws conclusions about the quality of the worm's overall impact. A jarring and devastating nuisance, the Morris Worm ultimately spread awareness of the gravity of computer security to both legitimate and malevolent users.

Contents

Incubation

The Internet's massive size and popularity today make it hard to imagine its humble beginning. What we now know as the Internet began as a research project sponsored by the Defense Advanced Research Project Agency (DARPA) called the Arpanet. On the day of its birth in December, 1969, the Arpanet consisted of only four connected nodes: UCLA, SRI, UCSB, and the University of Utah. Lines of communication between these nodes were "two million times slower than today's fastest networks" [Strawn(2014)].

The Arpanet opened the door for a flurry of new research projects. New ideas grew into numerous applications like file sharing, remote logon, and email. Soon, more nodes and networks were added. By the mid 1980s, the growth and success of the Arpanet prompted the National Science Foundation to build a network to connect research universities to some of its newly built supercomputer centers. This new network came to be known as the NSFnet. It made internetworking ubiquitous amongst universities and researchers, eventually connecting "more than 2,000 universities and colleges and a number of high tech companies" [Strawn(2014)]. Ultimately, the 1990s would see the NSFnet becoming commercialized into the Internet that we know today, but not without some growing pains. This paper focuses on one of the earliest and most vehement growing pains the developing internet[1] would face, known initially as the "Internet Worm" and later as the "Morris Worm".

An important detail in the internet's conception and early life is the attitude of its creators. This internet was developed by researchers focused on optimizing its efficiency, not its security. Some security measures were built in as an afterthought, but security was not woven into the core fabric of the internet or its applications. As the internet grew, researchers who used and designed it were generally assumed to have good intentions. According to [Lee(2013)], "...the Internet was like a small town where people thought little of leaving their doors unlocked. Internet security was seen as a mostly theoretical problem, and software vendors treated security flaws as a low priority." The same article

[1]The above summary of the birth of the Internet used the names "Arpanet", "NSFnet", and "Internet" for various stages in a growth of a widely connected network of computers. For simplicity, the rest of this paper will collectively refer to all of these stages as the "internet".

quotes the sentiments of Dr. Eugene Spafford [2]:

> The majority of people had some tie to computation for their jobs. I wouldn't say that we trusted each other, but there was more a community sense of caring for the stability and appropriate use of the computing systems... There was no such thing as a firewall back then. You didn't have people who were vandals or anarchists or criminals as much.

Thus, there was no perceived need for strong security. This very lack of security would ultimately provide a temperate environment for the incubation of a virulent infection: the Morris Worm.

The remainder of this paper tunnels through the worm hole in history that the Morris Worm has left behind. First, it covers *what* happened during the worm's outbreak and eradication. Second, it recounts the discoveries that were made by researchers about *how* the worm works upon its dissection. Third, it discusses the *immediate impact* that the worm had on the research community and the general public as they convalesced from the infection. Fourth, it traces the worm's *long term influence* on the evolution of future worms and shifting cyber-security perspectives. Finally, the paper provides a post mortem synthesis on the quality and depth of the worm's overall impact on the history of computer security.

[2] Dr. Spafford is a computer security researcher and professor at Purdue University. He is also executive director of the Purdue's Center for Education and Research in Information Assurance and Security and an internationally recognized expert in computer security.

Outbreak

This section describes events that took place between November 3^{rd} and Novermber 8^{th} of 1988. It outlines the Morris Worm's release, spread, and eventual eradication at a high level. Many works covering the Morris Worm begin with some variation of the phrase "On the evening of November 2^{nd}, 1988...". To honor this tradition, the next section does the same.

Black Thursday

On the evening of Wednesday November 2^{nd}, 1988, Robert Tappan Morris, a first year graduate student at Cornell University, released a worm onto the internet. The worm was released at around 6 pm at MIT. Spreading rapidly, by 11 pm it had infected machines at the University of Pittsburgh, RAND Corporation in Santa Monica, UC Berkely, the University of Maryland, the University of Utah, Stanford, the University of Minnesota, and the University of North Carolina [Seeley(1989), Spafford(1989c)].

Perhaps the worm spread faster than even Morris had anticipated. Around 11 pm and again at around 11:30 pm, he contacted Andrew Sudduth and Paul Graham [3] to tell them that he had released the worm and steps that could be taken to stop it. Furthermore, Morris requested that Andrew Sudduth alert the research community of the presence of the worm and how it might be stopped, which Sudduth did anonymously via email to a widely used internet research mailing list, called TCP-IP shortly thereafter. Unfortunately, by that time system administrators had already noticed the worm and had shut off internet gateways in an effort to quarantine the infection; thereby blocking Sudduth's email for several days [Lee(2013)][Eisenberg et al.(1989)].

The worm continued to spread throughout the night. By early Thursday morning, the infection had spread to the University of Arizona, Princeton University, Lawrence Livermore Labs, UCLA, Purdue University, Georgia Tech, Dartmouth, the Army Ballistics Research Lab, and the University of Chicago, amongst others. By this point many system

[3] Andrew Sudduth was a friend of Morris'. He and Paul Graham were both members of the technical staff at Harvard University's Aiken Computational Laboratory [Lee(2013)]

administrators were aware of the spreading infection, and Peter Yee of UC Berkeley and NASA Ames Research Center had posted a message to the TCP-IP mailing, stating "We are under attack"[Seeley(1989)] [Spafford(1989c)]. Eventually, Thursday November 11^{th} would come to be known as "Black Thursday".

At its peak, the infection is estimated to have spread to around 6,000 machines. That is 10% of the approximately 60,000 computers connected to the internet at that time[Eichin and Rochlis(1989)][Marsan(2008)]. The worm targeted systems running 4.2 or 4.3 BSD UNIX and SunOS. The worm caused many of its infected hosts to crash. Hosts that didn't crash were riddled with processes that appeared to be shells. Their process tables, open file tables, and sometimes swap space were exhausted. Latency soared as legitimate user processes competed for cpu time with worm processes. Logs showed odd activity that wasn't actually being caused by the users that they were recording. Finally, strange files were appearing in the */usr/tmp* directory [Seeley(1989)][Spafford(1989a)].

Fortunately, the research community had not only noticed the worm by early Thursday morning, they had already begun fighting back.

Inoculation

[Spafford(1989c)] points out:

> It is particularly interesting to note how quickly and how widely the Worm spread. It is also significant to note how quickly it was identified and stopped by an ad hoc col- lection of "Worm hunters" using the same network to communicate their results.

By 5 am on Thursday, "less than 12 hours after the program was first discovered on the network, the Computer Systems Research Group at Berkeley had developed an interim set of steps to halt its spread" [Spafford(1989a)]. While Black Thursday saw many a researcher and system administrator struggle with slow and crashing machines and trickling network connectivity, it also saw many swift and staggering successes for them. They devoloped worm vaccines with impressive speed. Mailing lists, including Dr.

Spaffords Phage Mailing List [4] were put together and maintained to coordinate anti-worm efforts. Researchers captured the worm and analyzed its behavior. By the end of the day, software patches had been posted to eliminate the vulnerabilities exploited by the worm, namely in the *sendmail* and *finger* applications. [Spafford(1989a)] [Seeley(1989)].

"By about 9 p.m. Thursday, another simple, effective method of stopping the invading program, without altering system utilities, was discovered at Purdue and also widely published"[Spafford(1989c)]. This involved simply creating a file called *sh* in the /usr/tmp directory.

Autonamously teaming up, researchers had discovered how to defeat the Morris Worm in just one day and infection was on the decline. Over the next several days, the Morris worm was eradicated. [Spafford(1989c)] relates that "By Tuesday, November 8, most machines had connected back to the Internet and traffic patterns had returned to near normal."

By early Friday, November 4^{th}, researchers had disassembled the worm's code [Spafford(1989a)] [Seeley(1989)]. Several surprises were in store for those performing the autopsy.

[4]The Phage Mailing List would become the main conduit of communication and source of information about the worm both during the worm crisis and in subsequent weeks as Morris was prosecuted [Spafford(1989a)][Spafford(1989c)][Lee(2013)].

Autopsy

In the wake of the Morris Worm outbreak, several publications presented in depth analyses of the worms anatomy based on disassembled worm code. This section explores the worm's anatomy as presented by these dissections at relatively shallow depth. Readers interested in more detail are encoured to read the following publications on which the entirety of this section is based: [Seeley(1989)] [Spafford(1989a)] [Spafford(1989b)] [Eichin and Rochlis(1989)]

The Morris Worm is written in C. Its head consists of a 99 line "bootstrap" or "vector" program [5] , used to download the main body of the worm and get it up and running on a new machine. About 3200 lines of C code comprise the worm's body. It contains code to help the worm preserve itself, discover and infect new machines, crack user passwords, and spread. It also contains unused code and storage space, as well as numerous software bugs and mistakes.

Self Preservation

The worm comes equipped with several mechanisms disguise itself and even fend off potential attacks from system administrators. Upon infection, a new worm reads any necessary files into memory and deletes them so that it leaves no trace on disk before zeroing out its argument list. It then changes its name to *sh* and masquerades as a harmless shell process. To continue to camouflage itself throughout its lifecycle, the worm occassionally uses the UNIX *fork* system call to change its process I.D. This also renews the long running worm's cpu allocation priority in the eyes of the UNIX scheduler, which may slowly reduce the amount of cpu time granted to long running processes. All of these tactics allow the worm to blend in and run unnoticed, but hiding is just one of its defense mechanisms.

The worm also employs measures to obscure itself in case of its discovery. A basic form of encryption based on the logical exclusive or operation encodes its disk files. The worm disables core files to protect against accidental core dumps in the event of a mistake, or

[5] also called a "grappling hook", or the "l1" program because it is found in the file *l1.c* [Spafford(1989a)]

even forced core dumps by a prodding system administrator. Unnecessary symbols are removed from the worm program symbol table to make analysis more difficult in the event that a worm is captured. These hurdles serve to passively obstruct worm analysis, but the worm takes its own security a step further.

The Morris Worm implements its own active authentication. When the bootstrap program downloads the worm body to a newly infected host, the client program authenticates the worm server via the exchange of a "magic number", a random number that both parties agree upon prior to communication. This prevents anyone from battling the worm with imposter worms. Equipped with tools to hide on a host system, change its appearance, and actively authenticate other worms, a worm program is reasonably capable of fending for itself. With multiple methods of self preservation at its disposal, the worm can focus on its other work.

Password Cracking

Once it has infiltrated a system, the worm opens the password file containing hashed account passwords and begins to try to crack them, guessing different passwords, hashing them, and comparing the results to the entries in the password file. The worm generates guesses for user passwords in the following ways:

- Users and administrators sometimes carelessly fail to change default passwords, or even to enter a password for a new account. The worm is therefore able to infiltrate some accounts with a blank password or other default passwords like *admin*.

- To remember passwords more easily, users often choose simple, easy to guess passwords. The worm tries very simple passwords, like the user name or the user name backwards, in the hopes that it can easily guess a user's password. The worm also carries with it a miniature dictionary of known commonly used passwords to try.

- If none of the previous methods work, the worm begins to try all of the words in the UNIX online dictionary, both capital and lowercase.

- For convenience, users often use the same password on multiple machines. If this is the case, once the worm has a user's password on one machine, it tries to login as the user with the same password on neighboring machines.

Notice that none of these methods attempt to attack the hashing algorithm used to encode the passwords in the password file. Rather, the worm attacks a known tendency of users to choose weak passwords.

Reproduction

Because of the security flaws they exploit, the Morris Worm's spreading mechanisms are particularly noteworthy. It first tries to capitalize on weak user passwords to create a remote shell with the *rsh* and *rexec* commands. If this doesn't work, it uses a buffer overrun in the *finger* application to create a remote shell on a new machine. Finally, if the two previous methods are ineffective, the worm sneaks in through a trap door in the *sendmail* application. These exploits are described in more detail below.

rsh/rexec

These programs are intended to allow users to spawn shell sessions on remote machines. After infiltrating a user account as described in the previous section the worm can use the *rsh* and *rexec* programs to log into a remote machine and infect it. The worm seeks out new host machines to infect by looking for connected host accounts in system tables and user files on an infected machine, or even scanning the network for other machines.

finger

The finger application gives directory information about users. In 1988 it used the C library *gets* function to read an incoming request into a buffer. Even in 1988, this function was known to have a classic buffer overflow vulnerability, but programmers often prioritize ease of use and convention over security when writing code. The Morris worm exploited this buffer overflow vulnerability, writing VAX machine code to create a remote shell into to the buffer and then overwriting the program's stack frame so that the remote cpu

would execute this code when the request finished processing. This allowed the worm to spread to any VAX machine running the Berkeley version of the finger server.

SENDMAIL

The sendmail application provides SMTP email service. The Morris Worm exploits a trap door that is intended to be used for debugging. Entering a debug command allows one to send commands instead of email to a remote host. The worm uses this little known feature in an unintended way, sending an anonymous debug email to a remote host with a copy of the bootstrap program and commands to compile and run it. This way, the infecting worm doesn't need to open a shell on the remote host to download and run the bootstrap program.

Maladaptive Traits

As witnessed on Black Thursday, the previously described innards of the worm are keenly adept in their respective purposes, suggesting that the worm program was fairly well engineered. However, upon further inspection, researchers found numerous instances of poor programming practice and several critical bugs that ultimately led to its exposure and subsequent demise.

To avoid infecting the same machine multiple times, the worm intermittently checks for other worms on the same machine. If another worm is found, the two perform a virtual coinflip and the loser agrees to self terminate. However, 1 in 7 times, the loser of the coinflip becomes immortal instead of self terminating, perhaps in order to prevent imposter worms planted by network administrators from attacking the real worms[Spafford(1989b)]. Rather than preventing redundant infection, however, this behavior results in numerous immortal worms clogging up the same machine. This particular bug is what led to the very noticeable latency and crashing symptoms that affected many machines on Black Thursday and made the worm rather noticeable.

[Spafford(1989b)], [Seeley(1989)], and [Eichin and Rochlis(1989)] all point many more mistakes, several of which are listed below:

- The worm's code doesn't bother to check return conditions for system calls. This means that it could try to access memory that was unsuccessfuly allocated, communicate over network connections that failed to connect, etc.
- The worm tries to send a UDP packet the a remote user account, *ernie.berkeley.edu* over a TCP socket, which, of course, doesn't work.
- In some places, the code passes incorrect arguments to functions, like a *struct* instead of a pointer to the *struct*.
- Several bugs exits in the worm's reproduction exploitations that make it less effective than it could have been.
- If the worm code doesn't compile fast enough on a newly infected system, error exiting doesn't erase all files, leaving traces of the worm leftover on disk.

Perhaps even more telling than the numerous mistakes in the buggy worm code are its unused features.

Vestigial Organs

Researchers dissecting the worm found several unused attributes. The worm contains redundant code, dead code[6], and meaningless code. For instance, it tries to remove the non-existent file */tmp/.dumb* and it redundantly seeds a random number generator multiple times. Some of its functions have local variables that are never used and it contains code that is only called in conditions that can never be met.

Dissecters also found conspicuous and curious omissions. The worm doesn't bother to do anything special with the *root* account if it is cracked, even though this account gives the worm complete power over the host machine. It doesn't delete system files, modify other programs, attempt to *deliberately* disable a system, install back doors or trojan horses, attempt to record or transmit passwords or other data, or engage in any other malicious behaviors that one might expect.

[6]code that is never used

A particularly interesting omission lies in the unused spaces in its file table. The worm maintains a data structure that can hold information about and transer up to twenty files, of which it only uses three. These other slots could have housed files that *do* do the pernicious things mentioned previously. According to [Lee(2013)],

> An early version of the worm recovered from an automatic backup of Morris's Cornell files included extensive comments describing Morris's vision for the project. Those comments suggest that Morris had even more ambitious goals than he eventually achieved. Morris didn't just want to create a worm that would silently replicate itself across the Internet. He hoped to build what we would now call a botnet: a network of thousands of computers coordinating with one another and available to carry out further instructions at the direction of their master.
>
> The worm, he wrote in comments on an early version of the worm, will need to "decide what to break into next" and will need "methods of breaking into other systems." He also wanted "some way for ME to send out commands, protected by an encoded password."

Perhaps it was Morris' intention to transfer the code containing these capabilities in the empty slots in the file table. [Lee(2013)] speculates that the datagram the worm failed to send to the *ernie.berkeley.edu* may have been a beginning phase of the development of this type of command center. It is clear that had Morris built in this or any other malicious functionality, the worm could have been much more dangerous.

Convalescence

Researchers and organizations spent many man hours fighting the worm. Time was lost rebooting machines and jobs that depended on infected machines were stalled for days. Email communication also took a hit in cases where networks were shutdown to quarantine the spread of the infection[Seeley(1989)]. For several days, a large portion of the internet and its connected computers were unuseable.

This opportunity cost the Morris Worm crisis caused is rather shallow, however, when compared to the reaction that the worm illicited. The phase in which one recovers after illness caused by an infectious disease is called *convalescence*. This section addresses the recovery of the computing community and the general public through their reponses to the worm crisis.

Awareness and Perception

The Morris Worm crisis elevated awareness of computing, the internet, and security. Prior to the crisis, much of the general public didn't even know about the internet, and therefore could not fathom the impact that a cyber attack could have. The computing community, aware of some of the very vulnerabilities that the worm exploited, knew that someone one *could* set off a catastrophic cyber attack, but didn't suspect that anyone *would*. The computer literate and computer agnostic were both faced with questions regarding the ethical nature of what Robert Morris had done.

Public

As stories about the worm crisis began to hit the news, people were suddently confronted with new terminology. "Computer Virus", and "Internet", amongst others, became new common vocabulary words for people who hadn't heard them before. The Morris Worm had thrust the concepts of large communications networks linking computers and attacks against them into the spotlight. People began digesting the new ideas, at times rather comically. [Lee(2013)] quotes, from an interview with Dr. Spafford:

> "I got one call from a newspaper in Southern Indiana," Spafford says. "The

> reporter asked me, in all earnestness, 'Do our readers need to worry about catching this virus?"'

[Marsan(2008)] quotes, from an interview with Steve Bellovin, a Computer Science Professor at Columbia University:

> "The Morris worm was the first time most people ever heard the word 'Internet,"' Bellovin says. "For most people, it was a novelty, a strange and wondrous world ... and one rogue operator could take it down. Nobody had ever heard of the Internet unless you were a computer scientist."

The news coverage over the next several months would reinforce a growing public awareness of two new ideas: that a huge communications network of connected computers exists and it can be used to perpertrate massive and destructive cyber attacks. It is curious to ponder that the existence of the internet and its potential for misuse both dawned on the public at the same time.

Researchers

Researchers were faced with some realizations of their own. While aware of the possibility of security breaches, computer scientists didn't appreciate the magnitude of the damage that could be caused or the likelihood that someone would actually launch a massive attack. [Seeley(1989)] points out that the vulnerabilities exploited by the Morris worm were already known to the computing community. In fact, Morris himself had warned people about them. [Spafford(1989b)] relates that "... at a recent meeting, Professor Rick Rashid of Carnegie-Mellon University was heard to claim that Robert T . Morris, the alleged author of the Worm, had revealed the fingerd bug to system administrative staff at CMU well over a year ago." Aware of many existing security vulnerabilities, system administrators still considered their risk for a cyber attack to be miniscule. The Morris Worm showed them otherwise.

Even amongst the computer literate, many people didn't know how to classify the attack without first developing a heightened awareness of cyber attack taxonomy. Of

the flurry of publications generated in response to the crisis, many of them started by addressing this issue. [Spafford(1989c)] points out:

> There seems to be considerable variation in the names applied to the program described here. Many people have used the term worm instead of virus based on its behavior. Members of the press have used the term virus, possibly because their experience to date has been only with that form of security problem. This usage has been reinforced by quotes from computer managers and programmers also unfamiliar with the difference.

[Eichin and Rochlis(1989)] obstinately asserted that the internet attack was caused by a "virus" [7] . The rest of the research community classified it as a "worm", stating that viruses instert their code into other programs and are passively executed when the host program runs while worms are standalone programs and use system resources to actively spread themselves [Seeley(1989), Spafford(1989b), Spafford(1989c)]. The Morris Worm incited this new appreciation for semantic nuance in cyber attack classification.

Upon inspection of the Morris Worm, researchers realized that the crisis could have been much worse. The worm could have deliberately performed malicious actions on infected machines, like deleting files and trafficking passwords. It could have been used as a mechanism to spread viruses. It could have installed back doors or trojan horses. Finally, it could have done all of this more covertly, making it much harder to discover, diagnose, and destroy. [Spafford(1989a)] points out that

> Entire businesses are now dependent, wisely or not, on the undisturbed functioning of computers. Many people's careers, property, and lives may be placed in jeopardy by acts of computer sabotage and mischief.

In light of all of this, the computing community realized that the internet at large needed a more robust immune system.

[7] In particular, [Eichin and Rochlis(1989)] thought the Morris Worm should be classified as a "lysogenetic" virus based on its behavior with respect to its host.

Morris: Criminal or Genius?

In the wake of the Morris Worm's destructive campaign, an important question faced the general public and the research community: How should Morris' actions be regarded? People began to debate about whether Morris was a criminal. There was also discussion about whether he was a genius, or "computer whiz". Faced with the first major cyber attack of this nature, researchers and the public developed a broad spectrum of contrasting opinions about Morris and his behavior.

An immediate question that generated diverse opinions concerned the ethical nature of Morris' actions, and, by extension, of cyber attack perpetrators in general. While some felt he was a vicious criminal that should be punished as harshly as possible, others argued that he was a hero for demonstrating security flaws. [Spafford(1989a)] observes that

> One oft- expressed opinion, especially by those individuals who believe the worm release was an accident or an unfortunate experiment, is that the author should not be punished. Some have gone so far as to say that the author should be rewarded... The other extreme school of thought holds that the author should be severely punished, including a term in a federal penitentiary.

Ultimately, a middle and more grounded opinion prevailed. The Cornell Commision that investigated Morris' actions as a Cornell student determined that

> Sentiment among the computer science professional community appears to favor strong disciplinary measures for perpetrators of acts of this kind. Such disciplinary measures, however, should not be so stern as to damage permanently the perpetrator's career.

This sentiment is echoed in [Spafford(1989a)]:

> ...it would not serve us well to overreact to this particular incident. However, neither should we dismiss it as something of no consequence... Furthermore, we should be wary of setting dangerous precedent for this kind of behavior. Excusing acts of computer vandalism simply because the authors claim there

> was no intent to cause damage will do little to discourage repeat offenses, and may, in fact, encourage new incidents.

Another question of interest brought to light by the worm crisis concerned the intellectual prowess associated with cyber attacks. While the public often envisions computer hackers as geniuses who flex an advanced intellect and knowledge of computer science to perpetrate cunning attacks, the academic community was quick to point out the fallacy in such thinking. [Spafford(1989a)] asserts thats "mastery of computer science and computer engineering now involves a great deal more than can be shown by using intimate knowledge of the flaws in a particular operating system." The Cornell Commission report points out that "Although the worm was technically sophisticated, its creation required dedication and perseverance rather than technical brilliance." Where the underinformed and computer illiterate saw technological mastery, educated and experienced members of the computing community saw cheap hacks that lacked any substantive creativity or insight.

A final question of interest that came up involved accountability for the incident. While it seemed obvious to some that Morris, the sole perpetrator of the attack, was responsible for the damage that his worm had caused, others attempted to redirect the blame. It is an unfortunate tendency in our society to blame perpetuation of many types of crime on the victim, either subtly or overtly. Computer crime is no different in this regard. In the wake of the Morris Worm incident, much of the public attempted to place blame on network administrators and computer users who were using vulnerable versions of the software that the worm exploited. Fortunately, once again, the research community, viewing computer security from an experienced, educated, and rational platform was quick to point out fallacy in misplacing blame. For example, [Spafford(1989a)] observes that

> The claim that the victims of the worm were somehow responsible for the invasion of their machines is also curious. The individuals making this claim seem to be stating that there is some moral or legal obligation for computer users to track and install every conceivable security fix and mechanism available... To attempt to blame these individuals for the success of the worm is equivalent to blaming an arson victim for the fire because she didn't build her

house of fireproof metal.

Whether or not the victims of cyber attacks should bear some of the blame, it was clear to everyone that they should be implementing stronger security measures.

Antibodies

Antibodies are produced to fight a biological infection. To fight previously unencountered infections, an immune system must often develop new antibodies. The same can be said cyber attacks and their countermeasures. As a result of the worm incident, Robert Morris was the first felon convicted under the Computer Fraud and Abuse Act (CFAA) of 1986. In the wake of the crisis, computer security gained recognition as a legimate and important research area. In response to the demonstrated need for computer crisis response infrastructure, the Computer Emergency Response Team (CERT) was established at CMU. Thus, the Morris Worm spurred the unprecedented use of some existing countermeasures and the development of important new ones.

The Computer Fraud and Abuse Act

Morris was tried and convicted as a felon under the CFAA. He plead guilty and publicly apologized for the incident. It was clear that he was the worm's author. It was also apparent that he hadn't intended to do the major damage that he did, and as a result he didn't serve any prison time. Instead, he was fined $10,000 and sentenced to three years probration and 400 hours of community service[Lee(2013)]. This seems to reflect the sentiment expressed by academics above: that he should be punished, but not too severely. This helped to establish the precedent that even accidental cyber crime would be taken seriously in the eyes of the law. It also set the precedent that the intenet fell within the purview of the CFAA[Adams(1996)].

CERT

The Morris Worm was aptly contained by a group of researchers and system administrators that autonamously came together. However, the computing community recognized the need

for a more formal approach to computer system crisis response. An National Computer Security Center (NCSC) post-mortem workshop issued a recommendation that "that a formal crisis center be established to deal with future incidents and to provide a formal point of contact for individuals wishing to report problems" [Spafford(1989a)]. ARPA installed the Computer Emergent Response Team Coordination Center as part of the Software Engineering Institute at Carnegie Mellon University as the first official internet crisis management center in direct response to the Morris Worm[Fithen and Fraser(1994)].

Computer Security Research

The worm crisis invigorated the field of Computer Security. This is cleary evidence by the numerous publications it generated. Papers like [Eichin and Rochlis(1989)] began using the incident as a platform to discuss underlying security principles and best practices, including arguments for stronger passwords, personnel secruity, and a software vulnerability patching. Computer Security suddently gained elevated legitimacy as a research field. [Marsan(2008)] quotes, from an interview with Dr. Spafford:

> "I had been told by my advisors there was no future in applied computer security research," Spafford says. "When this happened, suddenly a whole lot of people realized that the development of systems had leapfrogged the controlled, mainframe environment and a different kind of security model needed to be observed."

[Marsan(2008)] goes on to quote from an interview with Eric Allman, author of the *sendmail* application:

> "The Morris worm opened up computer security as a legitimate topic," Allman says. "Before that, there were a few people who worked on computer security and they were mostly cryptographers, but the concept of computer security as a field of study was legitimized a lot by that."

It wasn't just legitimate researchers that benifited from a study of the first major internet attack, however. Future generations of hackers would also have much to learn from the Morris Worm.

Evolution

In the field of biology, evolution is defined as change in a gene pool over generational time. Amongst other things, evolutionary biologists study this gradual change, tracing inhertited genetic traits from ancestors on to their descendants. Through their exhibition of similar traits, several other worm attacks that have occured in the past several decades appear to be descendants of the Morris Worm. Furthermore, as the cyber attacks evolved, so too did awareness and perception of cyber security. This section explores this subtle evolution of worm attacks and cyber security awareness.

Descendants

The Morris Worm provides concrete, ancestral examples of several commonly used attack vectors in computer security, most notably:

- using email as a spreading mechanism,
- identifying targets for infection by reading connection lists maintained on an infected machine,
- discovering new targets for infection by randomly scanning a network,
- exploiting widely known buffer overflow and trapdoor vulnerabilities
- and attacking weak passwords.

Several subsequent worms have found new ways to capitalize on some of these same exploits.

Code Red

In the summer of 2001, the Code Red Worm infected about 500,000 servers. It spreads by exploiting a known buffer overflow vulnerability in Microsoft Internet Information Services (IIS) servers. When the first infection was discovered, a patch for the IIS vulnerability had already existed for a month[Fisher(2001)]. [Fisher(2002)] calls the worm “a direct

descendant of Morris' creation". After infecting a new host, the worm scans the internet for other hosts running the same IIS server to infect.

Much like the Morris Worm, the Code Red contains several telling bugs, including a bug in the random IP address generator that it uses to scan the internet. In fact, [Berghel(2001)] points out that

> An interesting byproduct of the bug in the original Code Red was that instead of creating random paths to each infected server, Code Red infected each new server via the same path as its predecessor, thereby leaving a log of infected servers behind with each new infection. A failure of epidemic proportions in hiding one's tracks (should the perpetrator be identified, perhaps we should give him or her an honorary "F" in software design).

Unlike the Morris Worm, the Code Red Worm does more than just reproduce. It also contains code to display mischevious messages in web browsers on infected hosts, and even to launch a distributed denial of service attack (DDOS) on the White House website on July 19^{th} 2001. Fortunately, researchers found that code in the worm designed to flood the White House website with traffic used the site's actual IP address, rather than the URL "Whitehouse.gov". This meant that the planned attack could be easily thwarted by simply changing the IP address[Berghel(2001)].

The attack incited a familiar discussion about who is to blame for security vulnerabilites. [Berghel(2001)] relates:

> Many feel that the responsibility belongs to the administrators who failed to heed repeated warnings to patch their systems... Others believe the real culprits are software vendors that release unsecure products and government agencies that are slow to react to major security threats.

Notice that blame for the attack seems to have been cast upon almost everyone except the actual creator of the Code Red Worm.

Slammer

The Slammer Worm was released on January 25^{th}, 2003. Within 10 minutes of its release it had infected 75,000 machines and within 30 minutes it disrupted 1/5 data packets traversing the internet. It caused flight cancellations and disabled ATMs. As with previous worms, Slammer exploited a buffer overflow vulnerability in Microsoft SQL Server 2000. As with previous worms, the vulnerability that Slammer exploited was well known. A patch had even been released 6 months prior to the worm's release, but many system administrators hadn't applied it. It scanned random IP addresses on the web to discover new victim machines to infect [Tynan(2003)][Panko(2003)].

The finger of blame was again pointed at almost everyone but its creator. [Schultz(2003)] notes that "Microsoft will undoubtedly continue to hold users at fault for not installing appropriate patches, but the real problem is the bug-filled software that Microsoft and other vendors have continually produced over the years."

Blaster

The Blaster Worm was first discovered on August 11^{th}, 2003. Within a week it spread to over 100,000 Microsoft Windows systems. The Blaster worm randomly scans the internet for new victims. When a connection is found, it exploits a known trap door vulnerability in a Microsoft server for which there is a patch. It contains code to launch a DDOS attack on Microsoft's Windowsupdate.com website[Bailey et al.(2005)Bailey, Cooke, Jahanian, and Watson].

As with previous worms, it contains bugs and other poor coding practices that reveal fundamental programming deficiencies of its creator. [Bailey et al.(2005)Bailey, Cooke, Jahanian, and Wats points out that its repeated and redundant re-seeding of the random number generator that it uses "indicates that the worm author significantly lacks understanding of random number generators."

Conficker

The Conficker Worm began to spread in 2008 and infected millions of Windows computers. It does do something new, however. After infecting a host, it opens a communication

channel and awaits further instructions, effectively turning the host machine into a "zombie". [Lee(2013)] suggests that its authors have "finally achieved something like Morris's original, unrealized vision of using a worm to create a vast network of computers operating under the control of the malware's author." A network of these zombie computers is known as a "botnet" and "unscrupulous individuals use them to send spam e-mail messages, overwhelm Web sites with traffic, or perform other nefarious tasks"[Lee(2013)].

Evolved Awareness and Perception

In the decades after the Morris Worm crisis, many aspects of computer security have evolved and many haven't. As demonstrated in previous sections, new cyber attacks written by sloppy programmers continue to exploit old vulnerabilities that simply haven't been patched yet. The victim shaming mentality born after the Morris Worm continues to prevail as vendors continue to blame administrators for not applying patches and failing to update software, administrators blame users for using weak passwords, and users blame vendors for releasing vulnerable software, all seeming to forget about author of the cyber attack. When attention is turned on the perpetrators of cyber attacks, however, they are less often regarded as brilliant and more often regarded as despicable frauds out to steal financial information. This is likely because computers are for more ubiqitous than they were in 1988, so the general public is more intimately familiar the consequences of pesky cyber attacks.

The CFAA is still being expanded to deal with new types of cyber crime [Adams(1996)]. The CERT also continues to grow and respond to computer security crises of all shapes and sizes.

- Long term influence in formal cyber security and cyber crime
  - CFAA still being expanded to deal with criminals like Morris
  - CERT still handling real crises, like the newer ones mentioned above
  - CIRT is a commonly studied and applied topic
  - Cyber Security is a more popular and relevant field than ever

Post Mortem

- what this paper covered
- synthesize/speculate
  - what else could have happened?
  - if not for this worm, would something else have taken its place? probably, and it could have been meaner
  - was the overall impact positive, negative, or neutral?... neutral to positive
  - are Morris and other's like him brilliant? Not even in his day, and definitely not today... kiddies
  - How have worm driven attacks evolved? (not by much honestly, they just spread faster)
  - How has public perception of attacks shifted? - as far as blame, not much. as far as idolozing, very much
  - How has prosecution of cybercrime changed over time? - still a cat and mouse came between technology and legislation

References

[Adams(1996)] Jo-Ann M. Adams. Controlling cyberspace: applying the computer fraud and abuse act to the internet. *Santa Clara Computer & High Tech. LJ*, 12:403, 1996.

[Bailey et al.(2005)Bailey, Cooke, Jahanian, and Watson] M. Bailey, E. Cooke, F. Jahanian, and D. Watson. The Blaster Worm: Then and Now. *IEEE Security Privacy*, 3 (4):26–31, July 2005. ISSN 1540-7993. doi: 10.1109/MSP.2005.106.

[Berghel(2001)] Hal Berghel. The code red worm. *Communications of the ACM*, 44(12): 15–19, 2001.

[Coffin(2003)] B. Coffin. Slammer worm exploits risk management lapses. *Risk Reporter*, 2003.

[Eichin and Rochlis(1989)] Mark W. Eichin and Jon A. Rochlis. With microscope and tweezers: An analysis of the internet virus of november 1988. In *Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on*, pages 326–343. IEEE, 1989.

[Eisenberg et al.(1989)] Eisenberg et al. The Cornell commission: on Morris and the worm. *Communications of the ACM*, 32(6):706–709, 1989.

[Fisher(2001)] Dennis Fisher. Code Red worm exposes security flaws. *eWeek*, 18(29):1, July 2001. ISSN 15306283.

[Fisher(2002)] Dennis Fisher. Living with worms, viruses, and daily security alerts. *eWeek*, 19(6):20, February 2002. ISSN 15306283.

[Fithen and Fraser(1994)] Katherine Fithen and Barbara Fraser. CERT incident response and the Internet. *Communications of the ACM*, 37(8):108–113, 1994.

[Gardner(1989)] Phillip E. Gardner. The internet worm: What was said and when. *Computers & Security*, 8(4):305–316, June 1989. ISSN 0167-4048. doi: 10.1016/0167-4048(89)90092-8.

[Lee(2013)] T. Lee. How a grad student trying to build the first botnet brought the Internet to its knees - The Washington Post. *The Washington Post*, 2013.

[Marsan(2008)] Carolyn Duffy Marsan. Morris worm turns 20: Look what it's done. *Network World*, October 2008.

[Moore(2001)] Cathleen Moore. CM targets Web services. *InfoWorld*, 23(32):18, August 2001. ISSN 01996649.

[Moore et al.(2003)Moore, Paxson, Savage, Shannon, Staniford, and Weaver] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer worm. *IEEE Security Privacy*, 1(4):33–39, July 2003. ISSN 1540-7993. doi: 10.1109/MSECP.2003.1219056.

[Panko(2003)] Raymond R. Panko. Slammer: The first blitz worm. *Communications of the Association for Information Systems*, 11(1):12, 2003.

[Schultz(2003)] Eugene Schultz. Security views. *Computers & Security*, 22(3):176–187, April 2003. ISSN 0167-4048. doi: 10.1016/S0167-4048(03)00302-X.

[Seeley(1989)] Donn Seeley. A Tour of the Worm. In *Proceedings of 1989 Winter USENIX Conference, Usenix Association, San Diego, CA, February*, 1989.

[Spafford(1989a)] Eugene H. Spafford. Crisis and aftermath. *Communications of the ACM*, 32(6):678–687, 1989a.

[Spafford(1989b)] Eugene H. Spafford. The Internet worm program: An analysis. *ACM SIGCOMM Computer Communication Review*, 19(1):17–57, 1989b.

[Spafford(1989c)] Eugene H. Spafford. The internet worm incident. In *European Software Engineering Conference*, pages 446–468. Springer, 1989c.

[Strawn(2014)] G. Strawn. Masterminds of the Arpanet. *IT Professional*, 16(3):66–68, May 2014. ISSN 1520-9202. doi: 10.1109/MITP.2014.32.

[Tynan(2003)] Daniel Tynan. Dawn of the Superworm. *PCWorld*, 21(5):26, May 2003. ISSN 07378939.