CS 526: Information Security - Fall 2017

# Following the Morris Worm

Phil Van Every

December 3, 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…" [Seeley(1989)] [Spafford(1989a)] [Lee(2013)] [Spafford(1989c)] [Spafford(1989b)] [Eichin and Rochlis(1989)]. To honor this tradition, this 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

[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)]

Research Lab, and the University of Chicago, amongst others. By this point many system 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 sometims 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. Mailing lists, including Dr. Spaffords Phage Mailing List [4] were put together and mainted to coordinate anti-worm efforts. Researchers captured the worm and analyzed its behavior. They devoloped worm vaccines with impressive speed. 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 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 form 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 c.p.u. allocation priority in the eyes of the UNIX scheduler, which may slowly reduce the amount of c.p.u. 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 is used to encode disk files it uses. The worm disables core files to protect against accidental core dumps in the event of a mistake, or even forced core dumps by a prodding system administrator. Unnecessary

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

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. This means that trying a blank password, or common things *admin* often allow access to an account.
- 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.
- 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 is able to execute remote shells on neighboring machines with ease.

- Finally, 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.

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 describe 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, 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 know 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 programs stack frame so that the remote c.p.u.

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

- detecting itself / coinflip for self termination and immortality bug
- no checks for good return values
- pass struct instead of pointer to struct
- joke about bug fixes in worm
- other stuff pointed out by Spaf

## Vestigial Organs

- UDP to ernie over TCP connection, never sent
- accidental leftover files in /tmp (of compilation too slow?)
- unused functions and dead code
- unused file structure entries
- unused features- could have been much worse... ***(transition into next section)

# Convalescence

The phase after illness caused by an infectious disease is called *convalescence*. This section addresses the recovery and response of the computing community in the wake of the worm.

## Awareness and Perception

- of the research community
  - virus vs. worm terminology (taxonomy)
  - need for better passwords
  - hashing algorithm
  - this was too easy and it could have been much worse
- of the general public
  - Internet becomes widely know technology
  - cybercrime becomes more well known (at the same time... what a coincidence)
  - Genius computer whiz?
    - researchers: NO
    - public: Yes
  - Is Morris a criminal?
    - yes, fire and brimstone (some researchers)
    - yes, but don't punish too harshly (Cornell Commission and others)
    - no, just a mistaken experiment
    - no, he's a hero for demonstrating security flaws (transition to next section: blaming the victim)
    - the real question - should accidents/experiments be considered crimes?
  - Who is at fault: administrators vs. vendors... beginning of "blame the victim" mentality

## Immune System Growth

- computer security as a legitimate field
  - tons of publications
  - questions about security design principles (like least privilege)
  - questions about ethics and law ***(transition to next section)
- prosecution under CFAA
  - Morris' claim that it was unintentional... "knowingly" vs "intentionally" .. accident
  - Morris' claim that he didn't gain ungranted access because he had access to Cornell computers
  - Precedent: the internet falls under CFAA
  - Question: what if worm hadn't infected government computer?
- crisis response protocol
  - formation of CERT at CMU
  - phage mailing list

## Evolution

The Morris Worm and subsequent decompiled and analyzed code provided ancestral concrete examples of several attack vectors, like…:

- email as a spread mechanism
- scanning as a spread mechanism
- attacking weak passwords
- concept of use to spread other attacks

### Descendants

- Code Red
- Slammer
- Probably others... Nimbda, Stuxnet? etc... (I found mention but no substantive, concrete comparison)

### Evolved Awareness and Perception

- long term influence on perception
  - new vulnerabilities and attacks on same old vectors for same old reasons ***(transition)
  - with new worms, people still saying the same thing in the aftermath
  - there isn't much question now: cybercriminals are criminals, not admirable geniuses or heros
  - there isn't much question now: we blame the victim... both of them. Admins and vendors point fingers at eachother, following the Morris Worm trend
- 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.