Understanding Worms, Their Behaviour and Containing Them

Farhan Syed, farhans@cse.wustl.edu (A project report written under the guidance of Prof. Raj Jain)

Download

Abstract

Worms have emerged as one of the most potent threat to Network Security in the recent years. In this paper, we will present a detailed introduction about worms. This paper is intended for Network Security researchers who need a brief, yet comprehensive and technical introduction, when they start their research on worms. The paper covers a detailed introduction to worms and discusses some of the most potent and dangerous worms known today in brief.

Table of Contents

---

1. Introduction

The paper discusses worms, one of the most potent threats to Network security. Worms have the unique ability to mimic the approach taken by biological viruses. They can infect a host and then choose a medium to propagate to a neighboring host. Generally, the intent of the worm is assumed to be malicious. There are some worms, which do not have malicious intent. They are referred to as anti-worms. The paper is divided into six sections. The first section defines the worm using various criterions. Section two tabulates all the well known worms so far. The next section discusses some of the most potent worms. In section four, we will discuss the aspects of worm design. The next section is dedicated to discussing the methods to detect worms. The last section talks about containment and elimination of worms. Throughout the paper we will refer to the computer as host or PC and the person or victim as user.

---

2. Definition

Worms are one of the most ill defined concepts in Network Security. There is still no universal consensus on the definition of the worm. Usually worms and viruses display similar characteristics and their intention is also similar. To define worms, we will use the following points and then define worm based on these points.

2.1. Where and how does it start?

Worms can start on a host (Computer) in various fashions. It may be an attachment to a mail and when the attachment is opened, will execute the code written in the worm. This is called "invocation by human intervention". It may also start without any human intervention. For example, rebooting the system.

2.2. What it affects?

It affects the host. In contrast to computer viruses, it can affect anything on the host. It may corrupt the files on the host. It may affect communication of the host with other systems. It may disable the anti-virus software on the host, which will enable it to cause more damage. Computer Viruses in the other hand are very specific to files. Worms have a broader scope of attack than viruses.

2.3. How does it spread?

Worms are self replicating codes. This is the most distinct feature of a worm. Once they infect a host, they will try to find a nearby host which they can access, and copy themselves to that host. There it will perform the same actions that it performed on the original host.

2.4. What is it intended to do?

The intention of the worm depends on what the authors of the worm designed it for. Usually, the worms are intended to cause DoS attacks (mischief) or collect personal information from the host. It may scan the host and send all the confidential information on the host to the authors. It may create a back door on the host, allowing the author to remotely control the host. It may simply delete all the files on the host.

Based on the points mentioned above, we can define the worms as,

"A worm is a computer program, which can self-replicate and propagate over the network, with or without human intervention, and has malicious intent."

---

3. Brief History of worms

Now that we have defined worms, we will take a brief look at the worms that we have encountered until today. A very brief description of each worm is provided in table. The next section will discuss some of these worms in detail.

Table 1 : History of Wroms (Source [Wiki09][Darrell03][Eisenberg89][Chen04][Symantec99][Arbaugh00][Cliff02][Chen03][Cynthia04])

Worm(Author)	**Release/Discovered Date**	**Characteristics**	**Damage**
Creeper(Bob Thomas)	Early 1970's	Infected DEC PDP-10 computers running the TENEX OS. It replicated copies of itself to remote systems via ARPANET and displayed a message "I'm the creeper, catch me if you can!"	No damage. Was an experimental program.
Morris(Robert Tappan Morris)	2-Nov-88	Infected DEC VAX and SUN machines connected to the internet, running BSD UNIX OS. It targeted the buffer overflow flaw of operating systems.	Over 10 million USD
Happy99(Spanska)	Mid Jan 1999	Infected Windows OS. When executed, modified Winsock and attached itself to all the mails sent by the user.	No physical damage
Melissa(David L. Smith)	Mid March 1999	Was a MACRO in a word file that had password to 80 pornographic websites. When the MACRO was executed, it picked up the first 50 entries in the address book of the host and mailed a copy of itself. It clogged the mail servers.	Estimated over 400 million USD
ExploreZip(Author not known)	6-Jun-99	Propagated as a zipped attachment in Microsoft Outlook and registered itself to Windows NT Registry. Re-executed itself upon system reboot and mailed itself to all the people in the Outlook's address book. Also deleted Microsoft Documents and C and C++ source files	Not known.

		on the host.	
ILOVEYOU([alleged]Irene and Onel de Guzman, Reomel Lamores)	4-May-00	Propagated as a .VBS attachment in Outlook Mails and mailed itself to all the users on the users' mailing list. Changed the extensions of many files to .VBS and over wrote some others. Also was known to steal passwords and credit card information.	5.5 to 10 billion USD
SandMind(Author not known)	8-May-01	Attacked Sun Solaris and Microsoft IIS servers. Defaced US Government and anti-China Websites.	Not known.
Sircam(Author not known)	Jul-01	Propagated as an attachment in the mail and when file was opened, installed itself on to the host. It then scanned the drive for .xls and .doc files and randomly selected a file to email the people on the users' contacts list. It also scanned the shared network drives and copied itself to the shared drives. It them used Remote Procedure Calls to trigger the worm on the remote machine.	No physical damage
Code Red(Author not known)	13-Jul-01	Affected IIS servers and defaced websites that it hacked. Scanned all the hosts in the vicinity of the hosts and propagated itself. It did not check if the next host had the IIS Server or not. Used buffer overflow to execute binary code.	1.2 billion USD
Code Red II(Author not known)	4-Aug-01	Similar to Code Red, but infected the machines on the same subnet as the host	Over 2 billion USD
Nimda(Author not known)	18-Sep-01	Had 4 different propagation vectors. Compromised websites, LAN, Emails and executables. Caused worldwide DoS attacks.	8.75 billion USD
Klez(Author not known)	26-Oct-01	Exploited Microsoft IE's Trident Engine. Propagated as an attachment in mails and depended upon either buggy HTML engines or user action to execute. Once executed, would pick up a file randomly and mail it to the addresses on the users' mailing list.	Not known.
Slammer(Author not known)	25-Jan-03	Exploited buffer overflow in Microsoft SQL Server. Slowed internet traffic worldwide.	Over 1 billion USD
Blaster(Jeffrey Lee Parson of B variant)	12-Aug-03	Used Syn Flood attacks on windowsupdate.com causing DDoS	Over 500 million USD
Sobig(Author not known)	19-Aug-03	Scaned 20 IP address in the vicinity of the host and sent unsolicited mails via UDP port 8998. Caused major clogging	Over 5.5 billion USD

		in the mail servers.	
Sober(Author not known)	24-Oct-03	Affected Windows Operating System. Initiated as an email from FBI mentioning that user has been caught downloading pirated software. The user was asked to fill out a questionnaire. When the user opened the attachment, the worm installed itself into many windows directories. It then disabled firewalls and antivirus on the host and disabled access to assistance websites. It used Users' contacts list to send identical mails to everyone in the contact list. It was also suspected of stealing personal information from the host machine.	Not known.
Mydoom(Author not known)	26-Jan-04	Attacked Windows OS. Propagated as a "Sending Fail" mail and asked user to resend the mail by clicking on the attachment. Once the user did that, it installed a copy of the worm on the host and sent a copy of itself to the email address. Once installed, sends mail to different contacts in the users address book and also copied itself to shared folders of Peer-to-Peer networks. It also opened a backdoor on the compromised PC to allow access to the hacker at anytime.	Over 22 billion USD
Witty(Author not known)	19-Mar-04	Disabled the antivirus and firewalls made by Internet Security Systems on the host. Propagated via UDP in batches of 20,000. Generated traffic of 9 giga bytes per second in some cases. Spread at the maximum speed of data communication the host can offer.	Not known.
Sasser(Sven Jaschan)	30-Apr-04	The worm was reverse engineered from a Windows patch that was suppose to fix the LSASS component that represents buffer overrun. This vulnerability was supposed to allow remote execution of code on the host without the knowledge of the user. The worm attacked all the Systems that had not installed this update from windows.	Over 14 billion USD
Santy(Author not known)	20-Dec-04	Santy was the first Web based worm that exploited vulnerability in the PHP scripting language. The PHP scripting language had a feature to provide a file on remote PC to be appended to the	Not known.

		URL. When the website opens, the file would get executed. The worm used this vulnerability to propagate.	
Nyxem/Blackworm(Author not known)	3-Feb-06	It was programmed to trigger on 3rd of every month, 30 minutes after startup. It was designed to replace all document files on the host with DATAERROR.txt.	No known damage.
Stration(Author not known)	Sept, 2006	Propagated as a mail from mail server asking to install a security update. This itself was the worm. It opened connections to the servers already compromised by the hackers and used those servers to propagate faster. It also used the information from the contact list of the user on the host which it used to propagate via emails.	Not known.
Storm(Author not known)	17-Jan-07	Propagated as news attachment in Europe and US. The user was asked to open the attachment to see the news. The virus then compromised the host and placed the host in to a botnet. The worm created a new network much similar to Peer-to-Peer networks. It used this network for propagation.	Not known.
Koobface(Author not known)	31-Jul-08	Propagated as a message to the people on facebook. Once the user opens the message he/she is redirected to a website that is affiliated to facebook and asks the user to download the update on Adobe Flash. The downloaded file was the worm. Once installed, it directs the user to all malicious websites.	Not known.
Conficker(Author not known)	Oct-08	Had a specially crafted Remote Procedure Call that forced a buffer overflow and executed a shell code on the host. It then installs a HTTP server on the host and downloads the worm in the form of DLL and attaches it to the windows processes. It also tries to hack shared drives and if the drive is password protected, it uses brute force to hack the password. This generates large amount of network traffic and also causes user account lockouts.	1.2 million USD

Now that we have a general idea of what some of the worms are and what they did, we will take a few of these worms and discuss them a bit more in detail.

4. Understanding Popular Worms

We will now discuss some of the worms that used new techniques. We will discuss only those worms that were non derivative and showed a new technique to attack the system. This section is important to understand how worm writers approach the system to attack. It will also give us an idea about how we can build security measures to avoid them.

4.1 Creeper Worm:

Released in early 1970's and written by Bob Thomas, it was an experimental program to demonstrate the power of programming. Most of the worms written at the time were a result of fascination for self replicating programs by the programmers. There was not malicious intent and the worms did not hide. They were sent in clear. The Creeper worm was written to infect DEC PDP-10 computers running the TENEX operating system. The program used the ARPANET to propagate from node to node and display a message "I'm the creeper, catch me if you can!" A program, Reaper, was written to counter Creeper.

4.2 Morris Worm[Eisenberg89]:

Released in 1988 and authored by Robert Tappen Morris, was the first known worm that had malicious intent. According to the author, the worm was not suppose to cause any damage and was intended to gauge the size of the internet. It however, did cause DoS attacks. The worm exploited the vulnerabilities of Unix sendmail, rsh/rexec and weak passwords. The worm initiated a process on the host and found new hosts to propagate the code. Once it found a new host it would copy itself to the new host and start an additional process there. The worm has a condition to check if the worm is already running on the host. But Morris has programmed in such a way that the worm propagated to the new host even if the answer was "Yes". Every new instance of the worm on the host caused an additional process to be launched. And each new process slowed the system down until the system was unusable. The Morris worm is also considered as the Great Worm as it was first of its kind and it demonstrated the amount of impact such programs can have if they are not secured. It also changed the perception of system Downtime and Internet Security forever.

4.3 Melissa Worm[Chen04]:

This was a worm that caused wide spread damage to the internet and for the first time huge losses to everyone around the planet. It caused over 400 million USD in damages across the globe and shutdown many organizations. It was written as a MACRO on Microsoft Word Document and this helped its widespread propagation. It was released in Mid March 1999 and was authored by David L. Smith. The worm was very simple in its concept, but demonstrated a new technique to propagate. Many of the worms that were written in the years to come, were derived from this concept in one way or another. The worm was present in the MACRO of a MS-WORD document and propagated as a document that supposedly contained passwords for 80 pornographic sites. If the user opened this document, and many of them did, it would execute the MACRO. Once the MACRO was executed, it would pick up the first 50 contacts from the users address book and mail a copy of itself to all the addresses. Since the worm was essentially an email worm and it mailed 50 address every time it infected a new host, many mail servers were clogged with the mails. This caused a wide spread DoS attack. Most of the techniques used by this worm laid the foundation or methodology for many variants and newer worms. Papa and Syndicate are two such variants.

4.4 ExploreZip[Symantec99]:

This worm took the concept of Melissa worm one step further. Melissa worm was not designed to reside on the system. ExploreZip was. The worm propagated via email, just like Melissa, and was present in an attachment called ZIPPED_FILES.exe. Once the user opened the attachment, the worm would seem like a self extracting zip archive and then error out. Behind the scenes it would install itself on to the system and register itself in the Windows Registry. The worm would then stay dormant and do nothing. When the user reboots the system, the worm would get activated and mail a copy of itself to all the people in the address book of the user on the host. It would also delete all the C and C++ source files from the hard drive. There is no record of the amount of damage done by this worm. Since all the computers are not started at the same time, it is unlikely that this worm could have caused any DoS attack. It was not instantaneous like Melissa.

4.5 ILOVEYOU[Darrell03]:

This was the first worm to take the cost of damage to billions of USD. An estimated damage caused by this worm was between 5 and 10 billion USD. The worm was written in VB Script and propagated as an attachment in the email with a message "ILOVEYOU". When users opened this attachment, it would register itself onto the Windows Registry. This would activate the worm after every restart of the system. It would then, search all the drives connected to the host for all files with extensions *.JPG, *.JPEG, *.VBS, *.VBE, *.JS, *.JSE, *.CSS, *.WSH, *.SCT, *.DOC *.HTA, *.MP3, *.MP2 and rename them to .VBS. It also had a component called WIN-BUGSFIX.EXE" or "Microsoftv25.exe". This was a password stealing program. The worm propagated across the network by using the addresses present in the address book of the user. Since the worm activated immediately and also on restart of the PC, the amount of email it generated crippled many mail servers and also individual PCs. The worm was allegedly authored by Irene, Onel de Guzman and Reomel Lamores from Philipines.

4.6 Code Red[Cliff02]:

This worm took the approach to attacking in a completely different direction. Instead of relying on mails address in the user's contact list, it performed network scanning and used the IP addresses connected to the host as a vector for propagation. It attacked the IIS servers and defaced many websites. It used the vulnerability of buffer overflows on IIS servers to execute binary code on the hosts. The initial worm did not check if the new host has windows or was running IIS. It also did not check if the IP address it was trying to access exists. The later versions of this worm were more inclined towards the local subnet rather than accessing some random IP. The total cost of damage was about 1.2 billion USD. It demonstrated a new technique or worm propagation.

4.7 Nimda[Chen03]:

This was the next generation worm in its own league. It had 4 different propagation vectors. It could propagate via Websites, LAN, Emails and as executables. In emails it was disguised as a BASE-64(Binary) file readme.exe in the MIME Section. It would pick up the address retrieved from the user's MAPI Service. In the browser mode of propagation, the worm would rename many of the system files to .htm, .html and .asp. These pages would get executed and download the worm onto the machine, thus infecting the host. In the LAN Mode, it would copy itself on to all the writable shared directories that it could find. If the remote user opened these shared drives and if the "auto preview" option was enabled, the worm would infect the remote computer. It would them repeat the same process on the remote PC. The estimated cost of damage of this worm was about 8.75 billion USD.

4.8 Mydoom[Cynthia04]:

This was the most notorious worms of all times with the highest damage of 22 billion USD. It propagated as a "Sending Failed" mail from the mail server and asked the user to click on the attachment to resent the mail. If the user opened the attachment, it would show that it's resending the mail and in parallel, installed the worm. The worm would then send a copy of itself to all the address in the address book and also copy itself to Peer-to-Peer shared drives. The worm also opened a back door for the hacker to get back anytime.

4.9 Sasser:

This worm was unique in the manner in which it was developed. The worm was reverse engineered from one of the patches provided for Microsoft Windows. The worm would exploit the vulnerability the patch was suppose to address and was targeted at systems that had not installed the update yet. It did not portray any new technological advance from the way the worm behaved. But the design of the worm was a step further in worm innovation. It targeted the LSASS component that represents Buffer Overflow and executed binary code on the hosts. Since buffer overflow causes erratic behavior or shutdown of the system, many organizations across the globe went down almost instantaneously. It caused a damage of over 14 billion USD and was authored by Sven Jaschan.

Now that we have discussed many of the worms of the past and understood what they employed to cause maximum damage, the next step is to understand the aspects that need to be considered to designing a worm. The next section will discuss some of the aspects of worm design.

---

5. Aspects of designing a worm:

In this section we will discuss various areas that a worm will need to address in its design, so that it can cause maximum damage. The success of a worm relies on how well these areas are implemented. If any one of these areas is not designed properly, the worm is rendered useless.

5.1 Finding Vulnerabilities in a system[Arbaugh00][Manual05]:

The most important step that every worm writer must follow is identifying the weakness of the system. When we talk about vulnerability of a system, we intend to ask, "What can do the worms job in the system with minimal actions?" If a worm needs to run millions of line of code to do an operation, the effectiveness of the worm is reduced. More lines of code will employ more logic, hence more ways to counter a worm. The ideal vulnerability would be such that one command will render the system useless or do the intended job. If a worm, for example, is written to modify a file that is read only by default, the effect of the worm will be near zero. It will only affect those hosts on which the file is manually changed to writable by the user. On the other hand, if the worm can change the permissions on the file by itself, or use an unsecured system function to do the job, the worm would affect all the hosts that it encounters. Worm designers, by convention, do not write worms on simple glitches in the system. The idea behind writing a worm is to cause maximum damage. So, the worm authors look for holes in the system that would cause maximum damage to the system or render the system useless.

System components like, Remote Procedure Calls (remote execution of a program), Buffer Overflows (where in data is stored in memory location other than the memory allocated by the programmer), Remote Command Execution (running a shell command remotely on a different host) etc, provide the worm authors the ability of using one worm to do as many things as it can. If we had a vulnerability, where a system component could delete all the files on the system, the worm writers could use this to write a worm that would stay dormant until a specific date and trigger in all systems at the same time. This would definitely cause tremendous damage to users across the globe. This however has two drawbacks. If the hacker deletes all the files from the system, he/she cannot use a back door to enter the system as the system would not come up and also cannot steal and personal information from the host. This is another aspect that the worm writes keep in mind. If the worm is intended to just steal information or create a back door, it will ensure that nothing is deleted from the system. If the intention of the worm is to cause DoS attack, then this approach is best suited.

If a user wants to create a back door on the host, which they can use to enter the host and take complete control of the host, the worm writers will try to find the most secure way into the system, quietly install the back door and terminate itself, after it has successfully propagated a copy of itself to another host. These types of worms do not cause havoc at once and there is not known data about the amount of damages they cause. These types of worms are carefully crafted to slip through holes in the firewalls, antivirus software and operating system components. Most of these worms try not to rely on user activity. Depending on the user activity means reduction in the speed of propagation. These worms try to keep the system intact so that the hacker can get the most out of the system.

5.2 Speed of propagation[Chen04][Cynthia04][Cliff02]:

This is fundamentally the top priority for worm writers. Since worms have a very short life span before they are detected and contained, worm writers go to a great extent to ensure that worms propagate from host to host at the maximum speed allowed on the connection. Whatever the worm's intention may be, this is one area where they will all try to show similar design characteristics.

The earliest worms used emails to propagate. This has the slowest rate of propagation. The worms will have to propagate from the host to the mail server and then will be sent to the next host where the user has to open the email to infect the machine. This process takes time of the order of minutes in the best case. Also, if the worm floods the mail server, the worms' speed of propagation is diminished by the damage caused by the worm. Even though this is a big bottle neck for worm writers, it has its own advantages. These types of worms can spread very rapidly over the globe across different mail servers in a very short span of time. This is due to the fact that the user's contacts usually belong to different mail servers and they way they are distributed over the demographics are fairly uniform compared to other approaches. The term "demographics" is used because using mail servers, the worms can propagate to different networks. If the worm has used IP Address scanning, it would affect mostly the internet connected hosts. Given the fact that everyone uses mailing services, most of the damage has been caused by worms in this category.

More speedy worms do not rely on emails, but reply on direct interaction with the network. They follow a root-to-braches approach in propagation. Here, they infect the first host (root) and propagate to its neighbors. They usually rely on faster network services like TCP and UPD. UDP uses single root topology (one host) and TCP uses dual root topology (one host and one server). Worms written to use TCP connects are slower than the ones using UDP. The worms using TCP connections, except the new host to either accept a TCP connection from the infected host to propagate the worm, or expect the new host to establish a connection to the server to download the worm. In this case, the speed of propagation is three times slower than UDP in the best case. Even though the TCP handshake does not consume a lot of time, it is important to remember that the worm propagation is a single call. Also, the TCP connections are dependent on network congestion. On a congested network, they will not propagate faster.

Worms using UDP are the fastest. They put the worm as a payload load on the UDP packet and send it to the next host. UDP packets are connectionless; hence they are three times faster than TCP worms. Moreover, they are not subjected to congestion as the worm writers usually send multiple copies to the same host to ensure delivery. The only time wasted by UDP worms is in scanning the network to find hosts. Since the UDP will need to start at a host and traverse the network node by node, they might take a lot of time to affect everyone around the globe. But, the worm will affect a very high number of hosts from the place it starts. The speed of propagation of UDP worms is limited only by the speed of the network.

5.3 Stealth[Darrell03][Symantec99][Cliff02][Chen03]:

The stealthier the worm is, the longer can it can go undetected. Hence, this is the next factor worm designers consider after Vulnerability and Speed of propagation. And longer times means, it can infect many more hosts. Many of the worms are not designed to withstand the securities built by the Operating System and

Antivirus Software. Worms are designed based on the vulnerability of the system. If the vulnerability is patched, the worm is rendered useless. Most of the worms are detected and countered within hours. Various techniques have been devised by worm writers here. A few of them disable the firewall and the antivirus software.

The best way is to compromise a trusted system process and use that process to do the job. A trusted process is one that meets a certain standard of security. These standards are defined and maintained by United States Department of Defense. A trusted process usually has full considerable access to the system and minimal blockage by firewalls and anti-virus programs. If a worm attaches itself to the trusted process, it will not trigger any alarm and will not need to disable any security. Stealth of the worm lies in going undetected, rather than disabling the security.

5.4 Propagation Vectors[Manual05]:

Propagation vector is the number of different ways a worm can spread. For example, some of the propagation vectors are port scanning and emails. This area of the worm design is dedicated to decide the medium through which the worm will propagate. The speed of propagation of a worm also depends on choosing a right medium of propagation at the right time. For example, if a worm is designed to propagate via Email and UDP, the speed of propagation is multiplied hundred folds. The worm would first send itself as an email to various contacts on the user's address book. It would then quickly spread on the local subnet. If we consider this operation globally, we can see that the distant reach of emails, coupled with speed of UDP will cause the worm to spread across the globe within hours. Sticking to a single propagation vector will reduce the propagation speed of the worm. The more the number of propagation vectors, the faster the worms can spread.

Now that we are familiar with different aspects of worm design, we can use same ideas to understand how the worms can be detected. Detection of a worm is the first and crucial step for containment of a worm. In the next section we will discuss some of the methods to worm detection.

---

6. Detecting Worms:

In this section we will discuss how we can detect worms in the system. This will give us an idea to device mechanisms to counter or contain the worms upon infection.

6.1 Detection by Monitoring "mistrusted processes"[Manual05]:

A mistrusted process is one that does not meet the security standards defined by United States Department of Defense. A mistrusted process has limited access to the system and is constantly monitored by the operating system security and anti-virus software.

The worm will either attach itself to a trusted process or will use another process to execute its task. If the worm uses a mistrusted process to execute its task, we can monitor those processes and detect any anomalies. For example, if the operating system defines a set of rules that every mistrusted process will need to follow, the worm will most likely violate at least one of the rules in order to do its job. In case the worm does not violate any rules, it will not be able to cause any damage as the rules will ensure that no critical functionality of the operating system is hampered via a mistrusted process. If at any point we see that a mistrusted process is violating any rule in the operating system, we can safely say that a worm has been detected. If the mistrusted process is buggy, it may cause false alarms. This may make the system believe that there is worm when there is none.

6.2. Detection by Monitoring "trusted processes"[Manual05]:

The trusted processes are not easy to monitor as they usually have more privileges in the system and perform a variety of tasks. But most of the trusted processes are designed for a specific task. If we see that any trusted process is performing a task it is not suppose to perform, we can say that a worm as been detected. For example, when the trusted processes is about to execute a program, we can check if the trusted process is executing a different program than the one it was suppose to execute. This approach is colloquially referred to as trusted process hijacking. If we detect a trusted process hijack, we can conclude that a worm has been detected.

6.3. Detection by Byte Pattern Monitoring[Newsome05]:

Byte Pattern, in simple terms, can be defined as the binary representation of the worm program. Let is consider that the worm propagates via UDP. All the UDP packets that the worm will send will have the same payload. Only the UDP Header will differ. If we have one packet that was sent by the worm, we can reject all the packets that contain the information as indicated by the sample packet.

While monitoring a byte pattern on the network, if we already know that a specific worm is active on the network and we know the byte pattern of the worm, we can detect packets that contain these worms and selectively quarantine these packets. This would decrease network speed, but it will prevent any known worm from propagating in the network. This is also known as signature based detection or anomaly detection. For example, on a broader scale, if we had employed this technique for ILOVEYOU worm, the mail server could have detected all the mails that contained the worm.

This approach will reduce network speed as every packet will need to be checked. It is most suitable for quick detection and containment of the worm. This check can be removed once the most optimal solution for worm containment is deployed.

6.4. Detection by Monitoring IP address scanning[Cliff05]:

The worm, after infecting a host, will try to scan neighboring IP addresses to find the next targets. The worm writers do not depend on standard commands, as monitoring and restricting the commands might lead to containment of the worm. Instead they try to evaluate the next host by scanning all the IP addresses in the address space of the host. This can be a good place to detect if a host is infected with a worm. Generally, any legitimate program exactly knows where to go on a network. Worms, on the other hand, need to find targets. If we monitor the number of IP address scanned by the host, and if it exceeds a certain threshold, then we can safely say that a worm has been detected.

Another method is to monitor the number of times a process tries to access an IP address that does not exist. This would happen if the worm designers randomly send packets to many IP addresses from a host. On the same lines, we can also monitor if the same byte pattern is being sent to multiple IP addresses from the host. This would be true for all the worms that spread via scanning IP addresses.

6.5. Detection by deploying Guardian Nodes[Lidong05]:

On every subnet, if we have one host, which employs the highest level of security possible and also installs a honey pot or any other monitoring process, we can trap all the packets that are intended to intrude the system. The probability that a worm will attack the guardian node is same as the probability with which the worm would attack a non guardian node. Since we use the guardian node as a decoy, we need not have the same amount of security on all the hosts on the subnet. If a worm gets detected by the guardian node, it would alert all the hosts on the subnet about the infection and also will provide a containment plan for the worm. Since the hosts are now expecting this attack, they can either reject such packets individually or, choose to update their antivirus software with the worm information. The same information can be used to trigger the antivirus software to eliminate the worm if it already exists on the host. This approach would initially load the network, but it will contain the worm within the subnet.

Now that we have seen how we can detect worms using various techniques, the next step is to determine how these programs can be contained once they are detected.

---

7. Containing and Destroying Worms:

Detecting the worm is fairly is simpler compared to containing the worm. If the designers of the worm choose to embed the worm into a system process, we will need to get the actual source code to remove the worm from the system process. We will need to destroy the existing file and replace it with a new one. We can also try to find the byte pattern of the worm within the system file and try to remove it. But this may lead to some instability in the system process. The best way to contain a worm is to contain the worm and block its access.

7.1. Quarantine and Monitor[Manual05]:

In this approach, we aggressively quarantine any process that shows erratic behavior. After isolating the process, we monitor it for a period of time corresponding to the erratic behavior shown by the process. If the process does not show any behavior during the time it's monitored, it is released. If it shows the same behavior again and again, it is quarantined and labeled as a worm. This is a very good approach when we are dealing with a very dangerous worm. But it will also mean that many healthy processes will also get quarantined even if they show a tiny erratic behavior due to some bugs. It costs a lot of money and resources to contain the worm. This method will be far cheaper than if the worm actually caused the damage. This method can be installed on guardian nodes to generate alert messages.

7.2. Setting minimal permissions for specific processes.

If a host can determine whether a process is infected by a worm or not, it can selectively filter that process and prevent it from performing any execution on the heap and stack pages on the host. Other permissions, like establishing a connection, ability to scan the network of the host, accessing and writing to the file system, ability to influence any other process that communicates outside the host like an email program etc can be restricted to the infected process. The restriction imposed by the operation system or antivirus program will depend on what erratic behavior the worm shows. In this case only one host falls prey to the worm. It's not the worm that will disable the system; it will be the worm containment program that will disable the system to a certain extent. This is better than infecting all the hosts and disabling them.

7.3. Installing the latest update from antivirus software and Operation System vendor[Darrell03][Arbaugh00][Cliff02][Cynthia04][Niels06]:

Once the system understands that it's under threat, the antivirus organizations and OS vendors are usually quick to respond to the attack. They provide a patch to counter the worm within a very short period of time. This is the best way to counter the worm. All the other techniques used are very generic as they need to detect any kind of worm in any form. But the patches provided to fix a worm are very specific to the worm and are aimed at either containing the worm or destroying it. This reduces the number of erratic processes in the system and hence far less discomfort and losses to the user. All users need to regularly update their antivirus software and their Operating Systems. Since worms are designed to attack known vulnerability of the system, taking precaution usually ensures that the worm is contained in the initial stages itself. Eliminating the vulnerability of the system is the most potent way of protecting the system against a worm attack. The initial methods of containment are like "first aid" in case of a worm infection. They are intended to provide quick protection from the worm damage at the cost of system disability and user inconvenience.

So far we have discussed the definition of worms, took a brief look at some popular worms, understood the designing aspects of the worm and discussed some methods to detect and contain the worms; we can now summarize the worms in brief.

---

8. Summary:

We have defined worms and took a brief look at some of the most popular worms known to date. We have discussed some of these worms in detail and understood their approach to attack the system. Next we discussed the various aspects that need to be considered in worm designing. Various methods of worm detection were discussed. The different methods of worm containment were also covered. All approaches to contain a worm do cause certain amount of harm to the system. This damage is well defined and is far less severe than the damage caused by the worm.
We have seen that worms are getting very innovative in the way they attack systems and also the increasing extent of damage they cause. We also saw that they best way to counter the worm is not very quick or light. It involves very heavy weight programs to detect a worm and counter them. The best way to counter a worm is to destroy it from the system, although, this is not always possible.

---

9. References

- [Wiki09] "Timeline of computer viruses and worms". Wikipedia. 2009. http://en.wikipedia.org/wiki/Timeline_of_notable_computer_viruses_and_worms
-
- [Darrell03] Darrell M. Kienzle and Matthew C. Elder "Recent Worms : A Survey and Trends" ACM workshop on Rapid malcode. 2003. http://portal.acm.org/citation.cfm?id=948189
- [Eisenberg89] T. Eisenberg et al. "The Cornell commission: on Morris and the worm" Communications of the ACM. 1989. http://allan.friedmans.org/papers/P2Psecurity.pdf
- [Chen04] Chen, T.M. Robert, J.-M. "Worm epidemics in high-speed networks" IEEE. 2004. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1306386
- [Symantec99] "Worm.ExploreZip" Symantec. 1999. http://www.symantec.com/security_response/writeup.jsp?docid=2000-121514-1418-99&tabid=1
- [Arbaugh00] Arbaugh, W.A. Fithen, W.L. McHugh, J. "Windows of vulnerability: a case study analysis." IEEE. 2000. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=889093

- [Cliff02] by Cliff Changchun Zou, Weibo Gong and Don Towsley. "Code red worm propagation modeling and analysis." 9th ACM conference on Computer and communications security. 2002. http://portal.acm.org/citation.cfm?id=586130
- [Chen03] Chen, Z. Gao, L. Kwiat, K "Modeling the spread of active worms." INFOCOM. 2003. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1209211
- [Cynthia04] Cynthia Wong et. al. "A study of mass-mailing worms." 2004 ACM workshop on Rapid malcode. 2004. http://portal.acm.org/citation.cfm?id=1029620
- [Niels06] Niels Provos, Joe McClain, Ke Wang. "Search Worms" Helsinki University of Technology. 2006. http://portal.acm.org/citation.cfm?id=1179542.1179544&coll=GUIDE&dl=&type=series&idx=SERIES320&part=series&WantType=Proceedings&title=CCS
- [Manual05] Manual Costa et. al. "Vigilante: end-to-end containment of internet worms" ACM SIGOPS Operating Systems Review. 2005. http://portal.acm.org/citation.cfm?id=1095809.1095824
- [Newsome05] Newsome, J. Karp, B. Song, D "Polygraph: automatically generating signatures for polymorphic worms" 2005 IEEE Symposium on Security and Privacy. 2005. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1425070
- [Cliff05] Cliff C. Zou et.al. "The monitoring and early detection of internet worms." IEEE/ACM Transactions on Networking (TON). 2005. http://portal.acm.org/citation.cfm?id=1103543.1103546
- [Lidong05] Lidong Zhou et. al. "A First Look at Peer-to-Peer Worms: Threats and Defenses" Springer Berlin / Heidelberg. 2005.

---

10. List of Acronyms

- ARPANET - Advanced Research Projects Agency NETwork
- BSD UNIX OS - Berkeley Software Distribution UNIX Operating System
- DDoS - Distributed Denial of Service
- DEC PDP-10 - Digital Equipment Corporation Programmed Data Processor model 10
- DEC VAX - Virtual Address eXtension Virtual Address eXtension
- DLL - Dynamic Link Library
- DoS - Denial of Service
- DOS - Disk Operating System
- FBI - Federal Bureau of Investigation
- HTML - HyperText Markup Language
- HTTP - HyperText Transfer Protocol
- IIS - Internet Information Services
- LAN - Local Area Network
- LSASS - Local Security Authority Subsystem Service
- MAPI - Messaging Application Programming Interface

- SQL Server - Structured Query Language Server
- UDP - User Datagram Protocol
- USD - United States Dollar

Last Modified: April 2009
This and other papers on latest advances in network security are available on line at http://www.cse.wustl.edu/~jain/cse571-09/index.html
SHARE Back to Raj Jain's Home Page