GIAC Security Essentials, version 1.4b, option 1

Internet Worms: Walking on Unstable Ground

By Jon Maurer

Abstract

Each day, worms are becoming a more common occurrence on the Internet. As the incidents increase, we must be thinking proactively in order to lessen the negative effects these worms have on the Internet community. It is important to remember that the livelihood of many businesses is based on an Internet presence.

The monetary losses incurred by businesses relating to these worms are hard to measure. Some estimate losses for each occurrence to be around \$1 billion. [1] The true value of damages may never be known. Many companies prefer not to publicly report losses since they do not want to diminish customer confidence in their services.

So far, we have been lucky. There has not been a worm that caused widespread permanent damage to computers. Everything we have seen so far has been related to some sort of denial of service. In order to prepare for the future, we need to be thinking ahead to beat worm writers before they release the next worm onto the Internet. The worms we have seen so far have been fairly sloppy. Most propagate slowly, giving system administrators a chance to catch up on their security practices prior to any major damage taking place.

Brief History of Worms

One definition of an Internet Worm is: "A program or algorithm that replicates itself over a computer network and usually performs malicious actions, such as using up the computer's resources and possibly shutting the system down." [2] The main difference between a worm and a virus is that the worm purposely replicates until the host runs out of space for it to go any further, while the typical virus infects one host and moves on to the next. This characteristic of worms is what causes them to be particularly damaging as they spread across the Internet and on to private networks.

Worms can be traced back to 1982 at the Xerox Palo Alto Research Center for network experiments. [3] Observations showed that it was difficult to control how many instances of the worm would exist at any one time. This same fact is what allows worms to spread so quickly in the wild.

The first malicious worm was launched in 1988 by Robert T. Morris from MIT. Morris realized his worm was spreading faster than he anticipated and tried to post removal instructions. These instructions were not viewed widely due to many administrators removing themselves from the Internet. It is thought this worm caused \$98 million worth of damage across the Internet. Morris was

[1] Lemos, p.1. "Counting the cost of Slammer."
[2] Webopedia, p.1. "Definition of Worm."
[3] Kobes, p.1. "Cyber Attacks."

sentenced to three years of probation, four hundred hours of community service, $10,050 in fines plus the cost of his supervision." [4]

Present

Starting in the late 1990's, we began to see an increase in both the frequency and damages caused by worms. On the Microsoft platform, these include Melissa in 1999, Code Red and Nimda in 2001, Slammer and Blaster in 2003. On various Linux platforms, the most famous include Ramen in 2001 and Simile in 2002. We have even seen a few occurrences of worms that can propagate across multiple platforms. Examples include Explorezip and W32.Winux.

Vectors

We have seen these worms attack via numerous vectors. Some of the most common vectors include open services, modems, peer to peer applications, email, web browser, instant messenger, newsgroups, solid media, and macros. Almost every application written for use on a computer has had some sort of vulnerability discovered. The problem occurs when these services are facing the Internet. This allows them to be exploited any time these vulnerable services are found. Most of these worms are based on newly found vulnerabilities called "zero-day exploits". [5] This means that the exploit is brand-new and has not been disclosed. Software companies have not had time to create software patches for zero-day exploits.

Another vector available for worms is email. Email worms typically attempt to exploit vulnerabilities in a specific email client such as Microsoft Outlook. These worms will typically attempt to send an infected file or script to all the contacts in the address book. They also often use social engineering by appearing to reply to a legitimate email in the Inbox. This tricks users into opening the email since it appears to be from someone known to the user.

One of the original ways to get a worm or computer virus was through floppy disks. The boot record of a disk could be infected and as the disk was read on a computer, that machine would also get infected. Today, we see the increased use of other solid media such as cdroms and flash media similar to the types used in digital cameras and USB keys. This physical transmission of data is bound to be exploited in the future.

Attacks on services are a very common vector by which worms propagate. This is typically because common services run on known ports so that clients can connect without needing proprietary information about the service. It would be very difficult to access a company website if it was set to something other than the standard port 80. This feature common to services running on every platform

[4] Boettger, p.1. "The Morris Worm"
[5] Bradley, "Zero-Day Exploits"

causes them to become easy targets for worms. We have seen attacks on Internet Information Services (Microsoft's IIS on port 80), Remote procedure call (RPC on port 135), Microsoft's SQL Server (database server on port 1434), and File transport protocol (FTP on port 21). Any time an exploit is found, it is quite easy to use the well known port number to search for and attack vulnerable servers.

Newsgroups and peer to peer applications are an easy way to share files. This also makes these technologies likely to become an infection vector for worms. The Morris worm mentioned above was first distributed from a newsgroup. Today, peer to peer applications are being used by millions of users and should be thought of as being un trusted. For this reason, it is a good idea to be sure common peer to peer application ports are blocked at the firewall.

We have yet to see worms attack via vectors such as PDAs, cell phones, and wireless technologies such as 802.11 and Bluetooth. Devices such as PDAs and cell phones are just starting to become complex enough to make easy targets. As the code they run becomes more complicated, it will be only a matter of time before vulnerabilities are found and exploited.

Wireless technologies are becoming an increasingly common way for attacks to get into a network. The 802.11b wireless protocol standard has already been proven to be inherently insecure. [6] The implementation of encryption algorithms used for the devices is flawed and can be broken in a matter of minutes. If exploited, this could be just another way for a worm to propagate across computer networks. Bluetooth is not very widespread at this time, but if it gains acceptance as 802.11b did, exploits will surely be found in the protocol. Imagine a worm that could spread from a computer, to a keyboard, to a wrist watch, to a telephone, and then to another computer. As devices become more "user friendly", we must take the time to thoroughly evaluate the security behind them.

The Future of Malicious Worms

In the future, we should expect to see worms that cause much more damage. Worms will be written by people with advanced programming knowledge. They may have a political or even military agenda. The payloads of these worms will most likely be malicious in nature, propagate via multiple vectors, and have specific targets.

Imagine the damage that could be caused by a worm that has infected several hundred thousand hosts. What if this worm had a malicious payload that was timed to set off a distributed denial of service (DDOS) against the top level domain name service (DNS) servers? If a DDOS attack could be kept active for a sufficient duration, this could effectively disable name resolution across the majority of the Internet.

[6] NIPC, "Best Practices for Wireless Fidelity (802.11b) Network Vulnerabilities"

Another possibility is for timed data destruction. If a similar worm infecting hundreds of thousands of servers was set to destroy operating system or data files at a particular time, it could also be quite costly.

Worse yet, what if the worm was programmed to infect as many hosts as possible and search for specific data, such as credit card numbers or other sensitive information. It could then take this information and deposit it in a preset specific location or even randomly copy the data. This would have the potential to cause chaos across the Internet as business scrambled to contain the threat. New regulations such as HIPPA would require massive customer notifications to take place at a large monetary cost and at a loss to the reputation of the business. [7]

We should expect future worms to propagate via multiple vectors and across multiple platforms. If the same worm had the ability to intelligently attack hosts based upon real time reconnaissance information, it could be deadly. This super worm could be based upon optimized propagation code tweaked from previous worms. Once the worm finds a good host, it could then use operating system fingerprinting similar to nmap to determine which exploits to attempt. Once the machine is exploited, the worm could then look at the newly found subnet for similar servers that could be quickly and easily exploited. These exploits could be from any of the vectors mentioned previously in this paper.

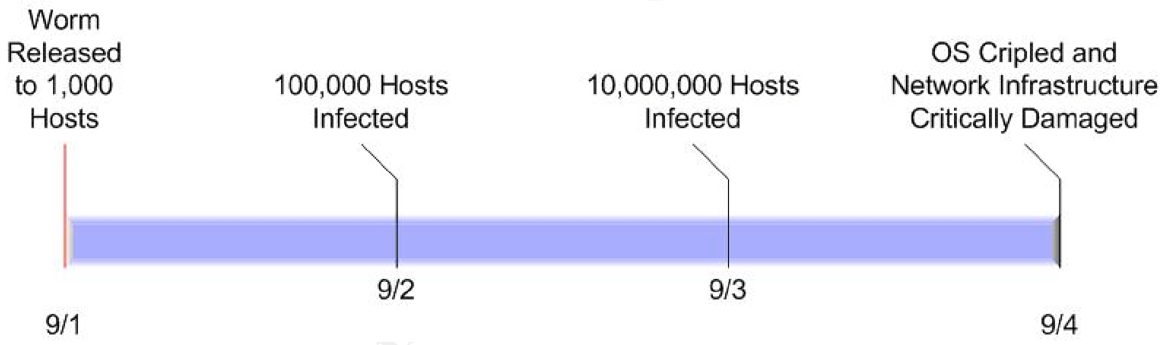
Pseudo code for the above example:

```
GenerateHostIP()      //find a random host IP

//step 1 – attack and replicate
While()    //attack loop - keep going until a pre-determined time
        ReconHost()   //find out what type of host and patch level, services
         LogOStoIP()     //start a list of OS and IP addresses
        while(exploited=0)      //keep trying until exploited
                FindExploit()  //query an existing list of
                Attack() //launch the exploit against the specific host service
        if (ChoseSameSubnet()=1)  //large chance to stay on same subnet
                GetHostSameSubnet()    //find a host on the same subnet
        else
                GenerateHostIP()   //find a random host IP

//Step 2 – damage the host OS
 if(hostOS=microsoft)
   delete kernel       //do something terminal to the OS
if(hostOS=linux)
   rm kernel        //do something terminal to the OS
```

[7] Decru. "Business Drivers for Storage Security."

```
//Step 3 – Damage the Internet infrastructure, causing slow response
 while()
     for (list of cisco hosts found)
       exploitCisco(hostname)       //exploit infrastructure devices
```

If thousands of instances of the worm would run through the attack loop until a pre-determined time and date, each instance would be able to replicate to the maximum number of target machines prior to going to the next step. It would also be concurrently building a list of hosts mapped to IP addresses to be used later. The internal logic would be designed to stay on the current subnet with a random chance to go to another subnet. This would allow the worm to infect as many hosts as possible before moving on. It could also keep a list of IP addresses it has already tried so that they would not be repeated. Once the pre-determined time arrived, the worm would then take steps to damage the operating system of the host machine by either deleting or modifying critical files depending on the host OS. The final step would be to go through the list of Cisco device IP addresses and exploit them in a manner that would prevent them from forwarding traffic anymore.


Worm Released to 1,000 Hosts
100,000 Hosts Infected
10,000,000 Hosts Infected
OS Cripled and Network Infrastructure Critically Damaged
9/1
9/2
9/3
9/4

This diagram illustrates a possible timeline of events which happen quickly enough to prevent new mitigating controls to be put in place to stop the spread of this worm.

Targets

There are many possible targets for the next Internet worm. Possible targets include infrastructure, business, and political targets. By affecting the confidentiality, integrity, or availability of this type of targets, attackers could hope to cause those sites damage. The motivation of the attacker will help determine what type of damage is caused by the worm. If the attacker simply wants to damage the reputation of the target, but not cause permanent damage, a denial of service attack might be probable. This denial of service could also serve as a distraction to provide cover for other, more serious attempts to damage the target.

There are any number of possible infrastructure targets. Many power systems are connected to the Internet with systems that are documented publicly. During the Northeastern United States power outage in 2003, some sources initially thought it might have been caused by the Blaster worm, the two events were proven to be unrelated. [8] We should expect this type of attack, and work to prevent it from happening in the future. Water distribution systems are also a viable target. There are also many Internet infrastructure targets including time servers, domain name servers, core routers, and patch update websites (windowsupdate.com). Political targets we could see more of in the future include military, government, and party websites. Just as the Internet and television have allowed people around the world to communicate, it also allows individuals to influence others with relative impunity. Extradition treaties and cyber crime laws can vary greatly depending on the country in question.

Threat Mitigation

The best way to mitigate the threat of getting infected by a worm is through defense in depth. [9] Through border protection, network segmentation, host security, application security, anti-virus software, and user training, a company can ensure the network is as secure as possible. There are many resources available on the Internet or by attending training classes that can help teach defense in depth.

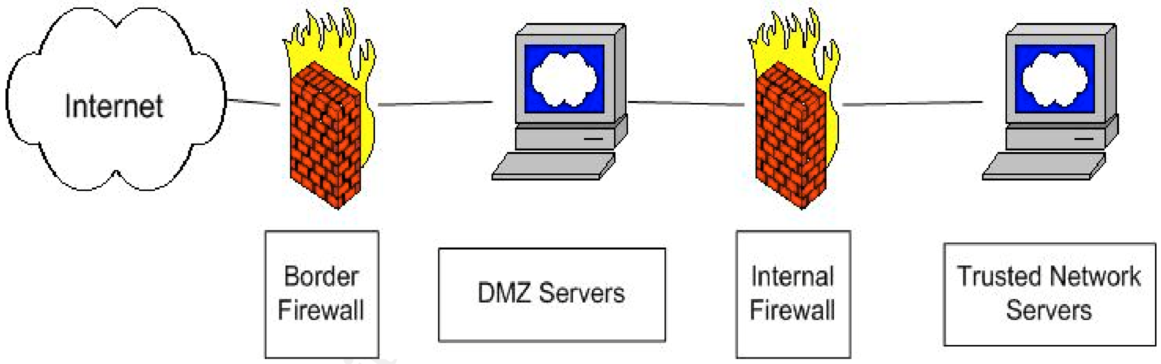
One of the biggest challenges against building a secure architecture is convincing the business that security is not about taking away access and slowing things down, it is about protecting the business from risk. Risk assessment can be used to include the business in the process of providing security. By including the business early and often, it be comes possible to influence decisions during requirements gathering rather than after the fact. This will help ensure business decisions are made while at the same time giving thought to security concerns.

The first step is to keep the worms from entering the network. Border protection is the classic strategy of keeping the bad things out. If the worm can not get to your network, it can not infect it. It is important to think about every possible entry point into the network. The most obvious entry points are Internet connections. A large company will most likely have more than a single entry point from the Internet. Firewalls and intrusion detection devices can help to mitigate risk at the border. Modems are another common entry point into a network. Remote administration of servers and other devices is common, especially if third-party support is used. As laptops become more popular, they become another way to get into the network. If a user takes their laptop home and plugs it into their home network with broadband to connect back to the corporate network (hopefully through a virtual private network or VPN), the laptop could be openly exposed to

[8] Reuters. "Outage Unrelated to Blaster Worm."
[9] McKee, Bob. "What it Takes to Develop Defense in Depth."

the Internet without the benefit of a firewall. This can be mitigated by a firewall appliance or personal software firewall such as Zone Alarm.

Network segmentation includes the use of security zones such as DMZs and physically separated networks. A DMZ refers to the concept of a demilitarized zone. This is typically a network segmented off between the Internet and the trusted internal network. Legitimate traffic from the Internet to web servers in the DMZ would pass through a firewall appliance prior to being able to get to the servers. A second firewall between the web servers and any back end processing servers would protect the trusted network servers from the Internet. This traffic would be tunneled through specific known ports and all others can be blocked. The traffic at each firewall can be analyzed using intrusion detection systems (IDS) and any attack attempts can be investigated. Another important concept with regards to the use of firewalls is egress filtering to prevent unauthorized traffic from leaving your network. Egress filtering is the concept of blocking specific ports, protocols, or hosts from sending data to the outside interface of the firewall. This prevents unknown programs such as worms from sending traffic outside this network, thus helping to prevent their spread across the Internet.



This diagram illustrates a simple DMZ design which segments services that need to be available to the Internet from trusted servers on the private network. This concept helps to mitigate the risk of allowing outside users from the Internet to access resources connected to the private network. The border firewall can be configured to allow traffic from the Internet to only access specific ports and IP addresses in the DMZ and block traffic to everything else. The internal firewall can be set to block all unknown traffic coming from both the Internet and the DMZ servers. Trusted traffic on specific ports would then be allowed from the DMZ servers to servers on the trusted network such as database servers.

Once we have kept the un trusted traffic away from the servers on the network, we need to talk about configuring the servers themselves. Host security is one of the hardest layers of security to keep up to date. This includes operation system patches, removal of unneeded services, and file permissions. Any server, regardless of operating system, should be hardened prior to being attached to the network. This is especially true if the server is accessible from the Internet.

New operating system and application patches are released each day. These need to be quickly tested and applied to servers in order to prevent known vulnerabilities from existing. When the Slammer worm was released, over 35,000 servers were unpatched and got infected. The Slammer worm was released six months after Microsoft had released the patch to MS-SQL server and the Microsoft Desktop Engine (MSDE). [10] System administrators must work to get these patches out in an expedited manner despite the high number of patches that are released. This can prove quite challenging in a large enterprise with a large number of servers and application dependencies.

Any services running by default on a server need to be disabled prior to making the server available as well. Unneeded services tend to be forgotten and therefore not patched. This could cause the server to have vulnerable services available and ready to be exploited, which would lead to compromise of the other applications or data on the server. It is also a good idea to follow the least privilege model when it comes to file permissions. The default file permissions on many operating systems are designed with ease of use in mind rather than true security. This could potentially allow an attacker to have access to a server when it is not needed.

Now that the servers are hardened, we need to discuss the security of the applications running on the servers. Application security is developing code with security in mind. This includes techniques such as coding to prevent buffer overflows as well as by implementing role-based security. If custom code is written, it must be assumed that the user will provide bad, or improper, input. Input validation for type and size will help prevent buffer overflows and other attacks, such as directory transversal attacks. Developers need to be aware of all the possible things a user might do, rather than just testing and doing error trapping on the "happy path". If software development companies would apply these application coding techniques, it would become more difficult for worm writers to find vulnerable code to exploit. One has to wonder if there is any correlation between the push for faster time-to-market and the increased existence of worms.

Since it is a given that there will be bugs in software, it is time to help stop known worms from spreading. Anti-virus software can help prevent files from getting infected on particular systems. Virus definition files need to be kept up-to-date on a regular basis. This can be a challenge in a large organization. This is important since anti-virus software can mitigate the risk posed by worms.

The last step is to be sure the people who use the systems understand what they must do to protect the network. It can not be left up to system administrators and security analysts to protect the network from worms. End user training, in the form of information security awareness is probably the most effective tool to mitigate threats to information systems. Users need to understand the risks

[10] Segal, Harry J. "Secure Your Servers Against Attack."

involved with opening email attachments, connecting laptops to home networks, and having weak passwords. Users are the customers of a network. Humans make mistakes and training is the mitigation for this risk. Proper password policy can help keep out attackers. This can be enforced by programmatic means and also by user training.

Summary

As we have seen recently, worms in the wild have been getting progressively more sophisticated and dangerous. Each of these worms has prompted behavior to make systems more secure. Administrators implement controls that increase their level of defense in depth. Firewall rules are updated; important servers are segmented from the network; host security is increased by current patches being applied and unneeded services being removed; more thought is put into application security; anti-virus definitions are updated; and users are informed about what they can do to prevent the network from being attacked. In this scenario, the network actually becomes more secure each time a new worm is found. This will also force worm writers to think of new and more devious ways to exploit systems. An interesting way to expand on this topic would be to conduct research to determine if there would ever be a point where the “immune system” of the Internet became so resistant to worms that the only worm able to replicate would be catastrophic event, causing so much damage it ended its own lifespan.

Worms cost businesses nationwide billions of dollars each year. We can assume the frequency and damages caused by worms will get worse before it gets better. As discussed earlier in this paper, there are many possible worst-case scenarios possible when it comes to worms and servers connected to the Internet.

Worm writers will put more time and planning into creating super worms. If everyone connected to the Internet works to secure their networks, worms can be slowed dramatically. By practicing defense in depth, we can hope to reduce the threat of future super worms.

List of References

Boettger, Larry. “The Morris Worm: how if Affected Computer Security and Lessons Learnd by it.” 24 December 2000.
URL: http://www.wbglinks.net/pages/reads/misc/morrisworm.html

Bradley, Tony. “Zero-Day Exploits.” 2003.
URL: http://netsecurity.about.com/library/weekly/aa031903a.htm

DECRU. “Business Requirements for Storage Security.” 2003.
URL: http://www.decru.com/docs/businessdrivers/

Kobes, Jason C. “Cyber Attacks”. 2001. URL:
http://www.public.iastate.edu/~jkobes/Attacks/WormsHistory.htm

Lemos, Robert. “Counting the cost of Slammer.” 31 January 2003.
URL: http://news.com.com/2100-1001-982955.html

Litterio, Francis. “The Internet Worm of 1988.” Unknown Date.
URL: http://world.std.com/~franl/worm.html

National Infrastructure Protection Center (NIPC). “Best Practices for Wireless Fidelity (802.11b) Network Vulnerabilities”. 17 May 2002.
URL: http://www.nipc.gov/publications/nipcpub/bestpract.html

McKee, Bob. “What it Takes to Develop Defense in Depth.” 2003. URL:
http://www.computerworld.com/securitytopics/security/story/0,10801,78170,00.html

Reuters. “Outage Unrelated to Blaster Worm.” 15 August 2003.
URL: http://zdnet.com.com/2100-1105_2-5064251.html

Staniford, Stuart. Paxon, Vern. Weaver, Nicholas. “How to 0wn the Internet in Your Spare Time.” 2002
URL: http://www.cs.berkeley.edu/~nweaver/cdc.web/

Segal, Harry J. “Secure Your Servers Against Attack.” August 2003. URL:
http://www.fawcette.com/dotnetmag/2003_08/magazine/columns/security/default_pf.asp

Webopedia. “Definition of Worm.” 03 June 2003.
http://www.webopedia.com/TERM/W/worm.html