01 - Types of Viruses and Worm Overview  
  
In this lesson we will talk about viruses and internet worms. Let's first define what a virus is, and then define what a worm is. A virus is effectively an infection of an existing program that results in the modification of the original program's behavior. A worm is code that propagates and replicates itself across the network. A worm is usually spread by exploiting flaws in existing programs or open services whereas viruses typically require user action to spread. For example, opening an attachment on an email or running an executable file that a friend gave you on an USB key. Worms propagate automatically. We will focus most of our attention on internet worms. But before we dive into the details of internet worms, let's first talk about the different types of viruses. A parasitic virus typically infects an existing executable file. A memory- resident virus infects running programs. A boot-sector virus spreads whenever the system is booted. A polymorphic virus encrypts part of the virus program using a randomly generated key. So one of the key differences between viruses and worms is that viruses typically spread with manual user intervention. Worms typically spread automatically by scanning for vulnerabilities and infecting vulnerable hosts when those vulnerabilities are discovered. A worm might use any of these techniques to infect a particular host before spreading further. In the rest of the lesson we will first talk about a brief history of internet worms, including the first Internet worm, called the Morris worm, and other famous Internet worms from the early days of Internet worms in the early 2000's, including Code Red and other well-known Internet worms of the time. We'll then talk about how to model the spread of a worm, in terms of scanning and infection rates, using analogies from epidemiology. Finally, we'll talk about design techniques for designing super fast-spreading worms. And we'll look at an example of a super fast-spreading worm.

02 - Worm and Virus Quiz  
  
So let's have a quiz to review our knowledge of the difference between worms and viruses. So what's the main difference between a worm and a virus? Is it that worms do not have destructive payloads, whereas viruses do? Is it that viruses only infect Windows machines, whereas worms can infect any kind of machine? Is it that viruses can spread more rapidly than worms? Or is that worms can spread automatically without human intervention, whereas viruses require human intervention to spread.

03 - Worm and Virus Quiz Answer  
  
The main difference between worms and viruses is that, worms can spread automatically by scanning for vulnerable hosts and spreading. Whereas viruses typically require user intervention to spread. Such as clicking on an executable file in an email attachment or installing a particular program from a USB stick.

04 - Internet Worm Lifecycle  
  
A worm's spread on the internet has the following life cycle. First, the infected machine might scan other machines on the internet to discover vulnerable hosts and subsequently infect the vulnerable machines that it discovers via remote exploit. Let's take a look at a couple of well known early worms and how they spread, as well as how one might design a super fast spreading worm.

05 - First Worm Morris Worm  
  
The first worm was designed by Robert Morris, Jr. in 1988. The worm itself had no malicious payload but, it ended up bogging down the machines that it infected by spawning new processes uncontrollably and exhausting resources. And at the time it was released, it affect ten percent of all Internet hosts. It spread, through three different propagation vectors. The worm tried to crack passwords ,using a small dictionary and a publicly readable password file and also targeted hosts. That were already listed in a trusted host file, on the machine that was already infected. This ability to perform remote execution was one way, that the worm was allowed to spread. The second way that it spread ,was in a buffer overflow vulnerability, In the finger demon. This was a standard buffer overflow exploid. And ,if you don't know about buffer overflows, I would urge you to take a computer security coarse. But essentially, this is a very common attack that makes remote exploits possible, effectively resulting in the ability to run arbitrary code. At the root level privilege. The third way that worm spread, was via the debug command in send mail, which is a mail sending service. In early send mail versions, it was possible to execute a command on a remote machine by sending an SMTP message. The worm used this, capability to spread automatically. A key theme that we'll see In the design of other worms, is this use of multiple vectors. Now any particular worm, may end up using, a different set of vectors depending on the remote vulnerabilities that it's trying to exploit. But the idea that any worm should be able to exploit multiple weaknesses in a system gives it more ways to spread. And often also speeds up the propagation of the worm. This worm design also followed the following general approach, which we see showing up over and over again in worm designs. First, the worm needs to scan other hosts to find potentially vulnerable hosts. In the second step, it needs to spread. By infecting, other vulnerable hosts. And in the third step, it needs to remain undiscoverable and undiscovered so that it can continue to operate and spread without being removed from systems.

06 - Worm Lifecycle Quiz  
  
So to review, what are the three steps in a worms life cycle? Please check three of the following options, infecting a vulnerable host, patching the hosts vulnerability after infection, scanning for other vulnerable hosts to infect, or remaining undetectable.

07 - Worm Lifecycle Quiz Answer  
  
An Internet worm first scans for vulnerable hosts, then infects them, and finally, typically takes steps to remain undetectable. A worm does not necessarily need to patch the host vulnerability, although some Internet worms have been known to do so, to prevent other worms from subsequently infecting the machine and interfering with the original worm infection. For example, if a worm was intending to spread to construct a botnet that launched a particular attack or was being used by a botmaster for a particular attack, then whoever had commandeered the machine, probably wouldn't want other attackers to come in behind him and also infect the machine and interfere with the planned attacks.

08 - Worm Outbreaks in Detail  
  
The summer of 2001 essentially saw a new era in internet security with three major worm outbreaks. These three major worms were Code Red One, version two, Code Red Two, and Nimda. Let's take a quick look at each of these worms. Code Red 1 was released on July 13th, 2001, and was the first modern worm. It exploited a buffer overflow in Microsoft's IIS server. From the 1st through the 20th of each month, it would spread By finding new targets using a random scan of IP address space, it would spawn 99 new threads, which generated IP addresses at random and then looked for vulnerable instances of IIS. Now version 2 Of code red one, was actually released six days later and fixed that random scanning bug. So that each instance of the worm scanned a different part of IP address space. After the scanning bug was fixed, the worm was able to compromise 350000 vulnerable hosts in a matter of only 14 hours. By most estimates, that was the complete set of hosts running the vulnerable version of IAS. On the entire internet. The payload of this worm was to mount a denial of service attack on whitehouse.gov. But a bug in the coding, caused the worm to die, on the 20th of each month. If the victims clock was wrong however, the worm would actually resurrect itself. On the 1st. Fortunately in this case, the payload which launched the denial of service attack on whitehouse.gov actually was launched at a particular IP address, not at the domain name. So the operators of the website needed only to move the web server to another IP address to defend against the denial of service attack. A better worm design would have been much more catastrophic. Code Red 2 exploited the same vulnerability but had a completely different payload, it was released on August 4th, 2001, and was called Code Red2 mainly because of a comment in the code. The worm actually only spread on Windows 2000, it actually crashed on Windows NT. The scan actually preferred nearby addresses It would choose addresses from the same /8 with probability one half from the same /16 with probability three eighths, and randomly from the entire internet with the remaining one eighth probability. The reason for preferring nearby IP addresses is that if there was one vulnerable host on the network. There was likely to be more, because the same administrator that failed to patch the compromised machine might have other machines on the same network that were also vulnerable. This notion of preferential scanning can speed up infections in some cases by increasing the probability that scanning will find another vulnerable host. The payload of this worm was an IIS backdoor, and the worm was completely dead, by design, by October 1, 2001. Nimda released on September 18, 2001 and was interesting mostly because it spread using multiple propagation vectors. It was effectively multi-model. So in addition to using the same IIS vulnerability as Code Red 1 and Code Red 2, there were some additional vectors that it used. It could spread by bulk email as an attachment. It copied itself across open network shares. It installed an exploit code on webpages on the corresponding web server running on the machine, so that any browser that visited the webpage for that server would become infected itself and it would scan for the Code Red 2 backdoors that that worm had installed. The interesting thing about the multi-modal nature of the Nimda worm is that signature based defences don't necessarily help, because of the many ways that it could spread, for example, by email or via a website exploit, Nimda actually needs firewalls. Most of the firewalls pass the email carrying Nimda completely untouched, using brand new infection with an unknown signature, and those scanners couldn't detect it. This was the first instance of a worm that exploited what we would call a zero day attack which is when a worm first appears in the wild. And the signature of the worm is not extracted until minutes, or hours later. Zero day attacks are particularly viralent because the worm can spread extremely quickly, before any type of signature-based antivirus has a chance to catch up and prevent the infections in the first place.

09 - Modeling Fast-Spreading Worms  
  
Here is a plot showing infection rate of the Code One Version Two Worm which ultimately infected 350,000 vulnerable hosts. Note the shape of this curve. The worm is effectively dormant or spreading extremely slowly for quite a period of time. And then there's a inflection point at which point the infection rate becomes exponential At some point then, infections slow and the infection rate ultimate plateaus, presumably after all of the infected hosts have been found. We can actually model the spread of these worms using the random constant spread model. If Kis the initial compromised rate, N is the number of vulnerable hosts, and a is the fraction of hosts already compromised, we can now express the number of hosts infected at a particular time increment in terms of the machines already infected, and the rate at which uninfected machines become compromised. So if Nda. Is the number of newly infected machines in DT, we can express that in terms of the number of machines already infected. Which is, N times a. So these are the host already capable of doing more scanning. And now K times 1 minus a is the rate at which uninfected machines become compromised, in a particular time interval dt. If we solve for a the fraction of hosts compromised, which is effectively the y-axis of this graph and we get the following. You get an exponential curve that is exponential where the growth rate depends only on K, or the initial compromise rate. This is very interesting because it tells us that if we want to design a very fast spreading worm, then we should design a worm such that the initial compromise rate is as high as possible. So how do we increase K. Or how do we increase that initial compromise rate?

10 - Increasing Compromise Rate  
  
One possibility for increasing the initial compromise rate, or designing a very fast spreading worm, is to create a hit list, or a list of vulnerable hosts ahead of time. That curve we just saw shows that the time to infect the first 10,000 hosts dominates infection time. So if we start by performing stealthy scans or some reconnaissance to construct a list of vulnerable hosts before we start spreading, then we can get rid of that initial flat part of the curve where the worm is effectively dormant. The second approach is to use something called permutation scanning Where every compromised host has a shared permutation of an IP address list to scan for vulnerabilities. Now if this list is randomly permuted and a particular host starts scanning from its own IP address in the list And works down, then different effected hosts will start scanning from different parts of this list ensuring that compromised hosts don't duplicate each other's work. One worm that exploited these techniques to spread particularly quickly was the Slammer worm, which spread in January of 2003. Exploiting a buffer overflow in Microsoft's SQL sever. In addition to using fast scanning techniques, the entire slammer code fit in a single small UDP packet. The UDP packet contain the worm binary, followed by an overflow pointer back to itself. It would say classic buffer overflow combined with random scanning. Once the control is passed to the worm code, it randomly generated IP addresses and attempted to send a copy of itself to Port 1434 on other hosts. One brilliant aspect of the slammer worm, is that because it was spread by via single UDP packet, it was connectionless, meaning that it could spread. And was no longer limited by the latency of network round trip time, but only by the bandwidth of the network, the worm caused $1.2 billion dollars in damage, and temporarily knocked out many elements of critical infrastructue, including Bank of America's ATM network. An entire cell phone network in South Korea, and five route DNS servers, as well as Continental Airlines' ticket processing software. The worm actually did not have a malicious payload, but the bandwidth exhaustion on the network caused resource exhaustion on all the infected machines. Here's a picture of the hosts around the world that Slammer infected. This damage was inflicted in just thirty minutes, due to the very lightweight nature in which Slammer spread.

11 - Slammer Worm Quiz  
  
So as a quick quiz, what allowed the Slammer worm to spread so quickly? Was it that TCPs reliable transport ensured a clean copy of the worm payload would spread to different vulnerable hosts? Was it that the worm spread by UDP, thereby enabling itself to spread with limited network overhead? Was it that it could infect many different types of operating systems including Linux and Mac OS? Or, was it that it could fit in a single packet?

12 - Slammer Worm Quiz Answer  
  
Slammer was able to spread quickly, because it spread via connectionless transport, or UDP, and because it could fit in a single packet.