1. Introduction

The ability of attackers to rapidly gain control of vast numbers of Internet hosts poses an immense risk to the overall security of the Internet. Once subverted, these hosts can not only be used to launch massive denial of service floods, but also to steal or corrupt great quantities of sensitive information, and confuse and disrupt use of the network in more subtle ways. An analysis presented of the magnitude of the threat that can be created using worms.

2. An Analysis of Code Red I worm

This worm used the vulnerability of a buffer overflow in Microsoft’s IIS Web server. It spread by launching 99 threads which generated random IP addresses, and then tried to compromise those IP addresses using the same vulnerability. A hundredth thread defaced the web server in some cases. But there was a bug, the random number generator was initialized with a fixed seed, so that all copies of the worm in a particular thread, on all hosts, generated and attempted to compromise exactly the same sequence of IP addresses. CRv1 had a linear spread and never compromised many machines.

Later, Code Red I v2 emerged, had the same codebase as CRv1 with the only difference that it fixed the bug with the random
number generation, had an end to web site defacements, and had a DDOS payload targeting the IP address of www.whitehouse.gov.

3.1 Localized scanning—Code Red II

It did use the same vulnerability as CR I. The payload, installed a root backdoor allowing unrestricted remote access to the infected host and on Windows NT it caused a system crash. It used a localized scanning strategy, where it was differentially likely to attempt to infect addresses close to it. Specifically, with probability 3/8 it chose a random IP address from within the class B address space (/16 network) of the infected machine. With probability 1/2 it chose randomly from its own class A (/8 network). Finally, with probability 1/8 it would choose a random address from the whole Internet. This strategy allows a worm to quickly infect vulnerable hosts across the internet. It also can spread very rapidly within an internal network once it manages to pass through the external firewall.

3.2 Multi-vector worms—Nimda

Nimda spread extensively behind firewalls. It spreads by infecting Web servers, bulk emailing of itself, copying itself across open network shares, adding exploit code to Web pages on compromised servers in order to infect clients, scanning for the backdoors left behind by other worms. Onset was quite rapid, rising in just half an hour from no probing to 100 probes/sec. Although, the payload is unknown.

Picture below shows the ecosystem of Worms that Microsoft endures in relation with time.
4. “Better” worms—theory

There are several techniques which, although not yet employed, could further significantly increase the virulence of a worm such as (i) hit-list scanning, (ii) permutation scanning, (iii) topologically aware worms, and (iv) Internet scale hit-lists. The goal is very rapid infection

4.1 Hit-list Scanning

One of the biggest problems a worm faces in achieving a very rapid rate of infection is “getting off the ground.” A worm should produce a list of most Internet connected Web sites to begin its spread. Before the worm is released, the worm author should collect a list of say 10,000 to 50,000 potentially vulnerable machines, ideally ones with good network connections. The worm scans the list and begins to spread. When it infects a machine, it divides the hit-list in half, communicating half to the recipient worm, keeping the other half. The worm will quickly go through the hit-list and establish itself on all vulnerable machines in only a few seconds. Hit list can be generated using Stealthy scans, Distributed scanning, DNS searches, Spiders, Public surveys, P2P advertisement.
4.2 Permutation Scanning

It solves the inefficiency of random scanning by assuming that a worm can detect that a particular target is already infected. In a permutation scan, all worms share a common pseudo random permutation of the IP address space. This permutation can be generated using a 32-bit block cipher and a preselected key: encrypt an index to get the corresponding address in the permutation, and decrypt an address to get its index. Any machines infected during the hit-list phase start scanning just after their point in the permutation, working their way through, looking for vulnerable machines. Whenever the worm sees an already infected machine, it chooses a new, random start point and proceeds from there. Self-coordination keeps the infection rate high and guarantees an eventual comprehensive scan. After any particular copy of the worm sees several infected machines without discovering new vulnerable targets, the worm assumes that effectively complete infection has occurred and stops the scanning process.

4.3 Simulation of a Warhol Worm

It is a combination of hit-list and permutation scanning techniques. It is capable of attacking most vulnerable targets in well under an hour, possibly less than 15 minutes. Hit-list scanning greatly improves the initial spread, while permutation scanning keeps the worm’s infection rate high. We can observe Warmhole Worm (Warhol) performance in relation to Code Red like worm (Conventional) and a fast scan worm (Fast Scanning) in the picture below.
4.4 Topological Scanning

It uses information contained on the victim machine in order to select new targets (e.g. email-address books). It boosts initial speed before they switch to permutation scanning. Peer to peer applications are highly attractive targets for worm authors because a worm could get a list of peers from a victim and use those peers as the basis of its attack.

4.5 Flash Worm

It’s a compact worm that begins with a list including all likely vulnerable addresses, and that has initial knowledge of some vulnerable sites with high-bandwidth links. It appears to be able to infect almost all vulnerable servers on the Internet in less than thirty seconds.

5 Stealth worms—contagion
Worms that allow a patient attacker to slowly but surreptitiously compromise a vast number of systems. These worms could prove much more difficult to detect and counter. The infection spreads from clients to servers and along to other clients, much as a contagious disease spreads based on the incidental traffic patterns of its hosts. That makes a worm stealthy to mechanisms that automatically detect the spread of worms and shut them down.

6. Updates and Control

We examine now, how an attacker can control and modify a worm after its dissemination.

6.1 Distributed Control

In a distributed-control worm, each worm has a list of other known, running copies of the worm and an ability to create encrypted communication channels to spread information. Any new command issued to the worms has a unique identifier and is cryptographically signed using an author’s key. Once a worm has a copy of the command, the command is first verified by examining the cryptographic signature, spread to every other known instance of the worm, and then executed. This allows any command to be initially sent to an arbitrary worm instance, where it is then quickly spread to all running copies. The key to such a network is the degree of connectivity maintained.

6.2 Programatic Updates

Many operating systems already support convenient dynamic code loading, which could be readily employed by a worm’s author. Another possibility has the bulk of the worm written in a flexible language combined with a small interpreter. By making the worm’s commands be general modules, a huge increase in flexibility would be achieved.
7. Envisioning a Cyber “Center for Disease Control”

Given all these threats that are possible to be escalate, we see a vital need of “Cyber-Center for Disease Control” (CDC). It could serve six roles: • Identifying outbreaks. • Rapidly analyzing pathogens. • Fighting infections. • Anticipating new vectors. • Proactively devising detectors for new vectors. • Resisting future threats.

8. Conclusion

We have examined the spread of several recent worms that infected hundreds of thousands of hosts within hours. We showed that some of these worms remain endemic on the Internet. We explained that better engineered worms could spread in minutes or even tens of seconds rather than hours, and could be controlled, modified, and maintained indefinitely, posing an ongoing threat of use in attack on a variety of sites and infrastructures. Thus, worms represent an extremely serious threat to the safety of the Internet. We finished with a discussion of the urgent need for stronger societal institutions and technical measures to control worms, and sketched what these might look like.