

Early DoS and Worms

Ben Wilde
7 February, 2005
Comp 290 – Network Intrusion
Detection

Outline

- Introduction to worms
- Potential damage that *could* be caused (theoretical)
- Examples of recent worms and DoS attacks
 - Slammer Worm
 - Shaft DoS attack
 - Mstream DoS attack
 - Trin00 DoS attack
- Worm Propagation: past and future

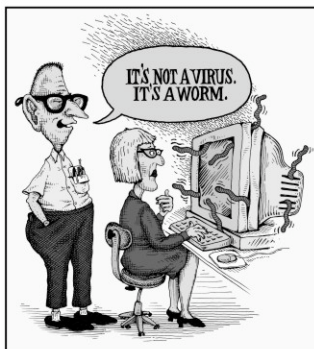
So, what are these “worms”?

- What’s a worm?
- How does it pick who is infected?
- What are their payloads?
- But why would somebody do this?

What is a worm?

- A computer worm is a program that self-propagates across a network exploiting security or policy flaws in widely-used services
 - First gained notice with the Morris worm of '88
- Different from viruses and other DoS attacks in that they self-propagate automatically, without need for user input

I’m sorry... this is terrible.



Who gets infected?

- For a worm to infect a machine, it must first discover that the machine exists
- There are a number of techniques by which a worm can discover new machines to exploit
 - Scanning
 - Target lists or Hit lists
 - Passive monitoring

Worm target selection

- Scanning
 - A worm scans IP addresses, which are selected either sequentially or randomly
- Target Lists
 - Worm infects computers based on a list of IP addresses, either generated by the attacker or extracted from information stored on the computer
 - More on this later...
- Passive Monitoring
 - Worm waits for potential victims to contact it, then spreads to that new computer

Potential Payloads

- The payload (code carried by the worm apart from the propagation routines) is limited only by the imagination of the attacker.
- Some examples fall into the following categories:
 - None/Non-functional
 - Internet Remote Control (control a user's computer)
 - Spam Relays (let spammers avoid known IP's)
 - HTML Proxies (hard to shut down illegal websites)
 - DoS
 - Data Collection (snoop around on a users hard drive)
 - Access for Sale (sell remote control of a zombie army)

Why would somebody do this?



Why ELSE?

- Experimental Curiosity
- Pride and Power
 - The old "show off" factor
- Commercial Advantage
 - Such as DDoS against competitors
- Extortion and Criminal Gain
 - "You wouldn't want your network to crash, would ya? Pay me and it won't..."
- Protest
- Economic Terrorism
- Cyber Warfare
 - Attack a government's computers

Potential Damage

- Now that we have some of the background down, let's look about some calculations about what damage could result...
- Keep in mind... it's just a little piece of software. How much damage could it really cause?
- Over 100 BILLION DOLLARS worth. That's how much.

Potential Damage

- Cost of a worm can be modeled as:

$$D_{system} = D_{rec} + T_{time} \cdot D_{time} + P_{data} \cdot D_{data} + P_{bios} \cdot D_{bios}$$

- The four parameters are:
 - Recovery Costs
 - Productivity loss due to down time
 - Value of data loss times the probability of unrecoverable data loss
 - Replacement value of the computer times the probability of hardware damage

Potential Damage

- Based on estimates of costs, the researchers produced the following table:

Attack & Assumptions	N_{inf}	D_{rec}	T_{time}	D_{time}	P_{data}	D_{data}	P_{bios}	D_{bios}	D_{system}	D_{total}
Baseline	50 M	\$20	16	\$35	0.1	\$2,000	0.1	\$2,600	\$1,040	\$52 B
Increased Downtime	50 M	\$20	32	\$35	0.1	\$2,000	0.1	\$2,600	\$1,600	\$80 B
Increased Data Damage	50 M	\$20	16	\$35	0.2	\$2,000	0.1	\$2,600	\$1,240	\$62 B
Increased BIOS Damage	50 M	\$20	16	\$35	0.1	\$2,000	0.2	\$2,600	\$1,300	\$68 B
Increased All	50 M	\$20	32	\$35	0.2	\$2,000	0.2	\$2,600	\$2,060	\$103 B

See Paxson's "A Worst Case Worm" for more information

Catch your breath...

- Let's back up... that number was based on numerous assumptions, all of which are for the worst case.
 - Assumes attacker has "infinite" resources
 - Like a nation state
 - Assumes all code in the worm is perfect and bug-free (yeah... right.)
 - Assumes that all difficult-to-predict possibilities go in favor of the attacker

Bring on some real worms

- Now, we'll begin looking at some REAL WORMS AND DOS ATTACKS that have appeared in the past five years.



- Again, I apologize. I really am sorry.

Common Bonds between worms

- Before we get into the specifics of any one attack, I'd like to cover the bonds that are shared between the attacks.
 - Specifically how someone else takes control of your machine
 - Command hierarchy
 - Password Protection
 - Attack Protocols

Installation of a Worm

- Let's start by looking at how worms get onto your computer...
- Warning: Lots of text and very few diagrams... I'll try to move fast.

Installation of Worms

- A stolen account is set up as a repository for pre-compiled versions of scanning tools, attack tools, root kits and sniffers, daemon and master programs, lists of vulnerable hosts and previously compromised hosts, etc.
- This would normally be a large system with many users, one with little administrative oversight, and on a high-bandwidth connection for rapid file transfer.

Installation of Worms

- A scan is performed of large ranges of network blocks to identify potential targets.
 - Targets would include systems running various services known to have remotely exploitable buffer overflow security bugs

Installation of Worms

- A list of vulnerable systems is then used to create a script that performs the exploit and sets up a command shell running under the root
 - Slammer stops at this phase and begins to propagate
 - Trinoo and others set up a TCP port and responds to the master confirming the success of the exploit
 - Sometimes, email messages are sent to confirm which systems have been compromised

Installation of Worms

- From this list of compromised systems, subsets with the desired architecture are chosen

Installation of Worms

- A script is then run which takes this list of "owned" systems and produces yet another script to automate the installation process.
 - Each installation is run in the background for maximum multitasking.

Installation of Worms

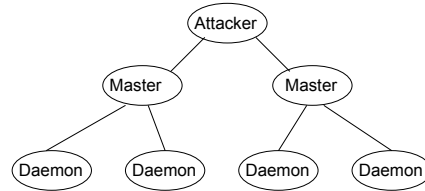
- Optionally, a "root kit" is installed on the system to hide the presence of programs, files, and network connections.
 - A root kit is "a set of tools used after cracking a computer system that hides logins, processes, and logs as well as usually sniff terminals, connections, and the keyboard." (Wikipedia)
 - A sniffer monitors packets on the network that a particular terminal is connected to, potentially giving a remote user confidential information
 - i.e. logins and passwords

Worm Hierarchy

- Now that we've seen how a worm can get loaded onto your computer, we can see how it is generally controlled.
- There are two general attack schemes
 - Interactive attack
 - Oblivious attack

Worm Control Hierarchy

- Interactive, controlled attack
 - Shaft, Trinoo, Mstream



Worm Control Hierarchy

- Commands are issued by an attacker
 - Specifying which hosts to attack, for example
- Responses can be returned by the daemons
 - Statistics about attack
- Because the attack is interactive and TCP-based, we say that it is "latency-limited"
 - You must wait to hear back from the daemon before you issue another command

Worm Control Hierarchy

- "Oblivious" UDP attack
 - Slammer
- One way
 - Does not need to hear back from 'daemons'
- Bandwidth Limited
 - Can go as fast as the network bandwidth will allow
 - Not latency-limited like others
 - Result of using UDP over TCP

Password Protection

- With trinoo, mstream and shaft, commands are issued by one or a few attackers, and can go to (potentially) thousands of zombies
- From the attacker's point of view, they don't want somebody else being able to 'hijack' their zombie army.
- Simple password schemes are used

Attack Protocols

- The main types of attack protocols are UDP, TCP SYN, or ICMP
 - They are increasingly being used in combination



Attack Protocols

- TCP SYN Attack
 - The attacker sends TCP connection requests faster than the victim can process them
 - Victim will respond to a spoofed IP address, then wait for a response (which will never come).
- UDP Flood
 - The attacker sends a bunch of UDP packets to a victim, resulting in a backlog of responses on the victim-side.
- ICMP Flood
 - Many flavors... one is to spoof source IP of the victim, then send out PING (or other) requests, flooding the victim with the responses.

http://www.riverhead.com/re/generic_ddos.html has an extensive list of attacks

Examples of Worms

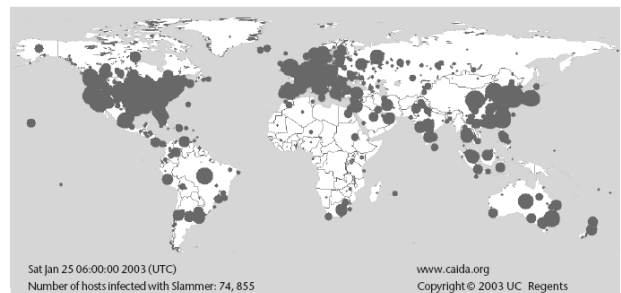
- We'll now discuss four different worms/DoS attacks
 - Slammer, MStream, Shaft and Trinoo
- For each, we'll see the following:
 - Damage that was caused
 - Exploits that were used to gain control
 - Communication between attacker and daemons
 - Examples of commands that were passed back and forth
 - Password Protection
 - Defenses against the worm
 - Weaknesses of the worm
 - Potential modifications to the worm

Slammer

- Also known as Sapphire
- Unleashed 25 January, 2003
- Within 10 minutes, 75,000 hosts were infected
 - Fastest worm seen to date
- Resulted in bandwidth saturation and network failure

Slammer

- This is after 30 minutes



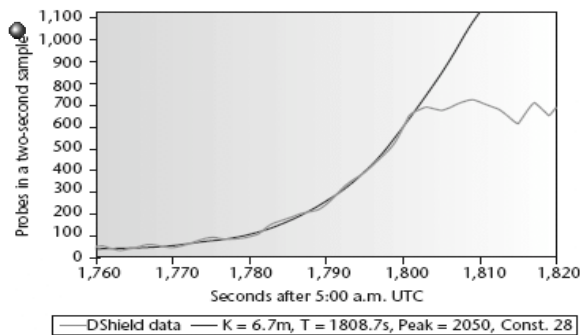
Slammer Exploits

- Exploited buffer-overflow vulnerability in computers on the Internet running Microsoft's SQL Server or Microsoft SQL Server Desktop Engine (MSDE) 2000.
- A single 404 byte UDP packet is sent out to port 1434, which then replicates itself if it succeeds with the exploit
 - The process is then repeated
- Because all of the code could be sent in a single packet and there was no limit imposed by latency, Slammer was the fastest worm seen to date
 - 3 times faster than worms like Code Red

Slammer Propagation

- Slammer used a random number generator to randomly select IP addresses to attempt to infect
 - A potential bug in the randomizer limited the scope of the hosts that were infected
- Slammer experienced exponential growth until networks became saturated and/or shut down.

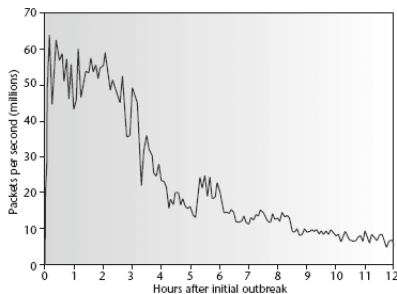
Slammer Propagation



Slammer Defenses

- Within hours of its first sighting, UDP port 1434 was monitored and filtered, helping to prevent future spread
- This port number could easily be changed in future versions
- Because human response pales in comparison to propagation speed, some sort of automated detection is necessary

Slammer Defenses



- Humans can limit spread, but are too slow

Slammer Weaknesses

- Potential bug in the random number generator limited the scope of the IP Address space that could be infected
- Eventually, it began to consume so much bandwidth that it limited its own spread

Trinoo

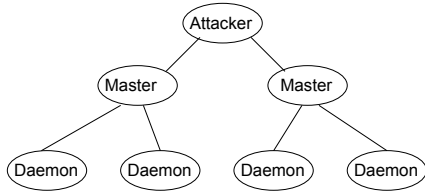
- Also written as Trin00
- Unleashed on August 17, 1999
- Swamps the network at the University of Minnesota
 - Renders it unusable for two days

Trinoo Exploits

- Targets Solaris 2.x systems, exploiting buffer overrun bugs in the RPC services "statd", "cmsd" and "ttddiscovered"

Trinoo Structure

- The structure is that of the bi-directional model, with a tree of control



Trinoo Communication

- Trinoo uses TCP for communication between the attackers and the masters
- UDP is used between the masters and the daemons (and vice versa)
- It is conjectured by the author that two different protocols are used to make it harder to detect the bug and trace it back to the attacker

Trinoo Commands

- When a daemon starts, it sends a **“Hello”** back to the master, which maintains a list of functional daemons
- The master then gets commands from the attacker and is able to issue a different command (with the same meaning) to the daemon

Trinoo Commands

- These commands include:
 - mtimer N
 - Set DoS timer to N seconds.
 - msize
 - Set the buffer size for packets sent during DoS attacks.
 - mdos <ip1:ip2:ip3>
 - Multiple DoS. Sends a multiple DoS command.

Trinoo Password Protection

- As mentioned before, passwords are used to prevent the network from being hijacked.
- Encrypted password is compiled in both the master and the daemon
- Clear-text password is then checked with these before commands are executed

Trinoo Defenses

- It would be very difficult to monitor all high (in number) UDP ports
- The proposed solution is to monitor UDP packets and look for patterns of communication between a master and a daemon

Trinoo Weaknesses

- Passwords are easily visible
- Once a daemon is located, you have a list of all masters IP addresses
- It is possible to hijack an active command session
- Detection is possible by mimicking a master's PING command and looking for PONG Responses
- Only one IP address can be hit at a time
- Exclusively does UDP attacks, none of the others

Shaft

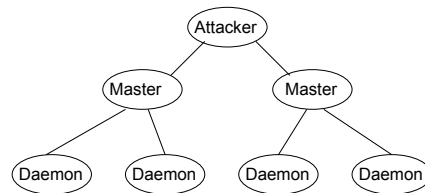
- He was one bad mother *shut your mouth*
- Unleashed in November of 1999
- Flooded the internet with packets, which meant that very little legitimate traffic could get through (which would have more dramatic effects during normal working hours)
- Also, there was reported poor (if any) DNS response.

Shaft Exploits

- Suspected to have entered just as Trinoo did
- Shaft was used along with a root kit, an inetd-based Trojan, a trojaned secure shell daemon, and a set of Unix shell scripts

Shaft Structure

- The structure is that of the bi-directional model, with a tree of control



Shaft Communication

- Shaft uses a telnet session for communication between attackers and the masters
- UDP is used between the masters and the daemons (and vice versa)
- It is conjectured by the author that two different protocols are used to make it harder to detect the bug and trace it back to the attacker

Shaft Commands

- When a daemon starts, it sends a "new <upshifted password>" back to the master, which maintains a list of functional daemons
- When a command is issued, it is sent to all of the daemons that it knows about
- Like Trinoo, the attacker can determine the time of attack and packet size
- Unlike Trinoo, the attacker can select the type of attack
 - TCP SYN, UDP, IMCP, or all three

Shaft Commands

- After these parameters are established, the attacker can issue commands like
 - `own tijgu a.a.a.a 5 5256`
 - 'own' tells it to begin an attack
 - 'tijgo' is the password
 - 'a.a.a.a' is the victim's IP address
 - '5' is the socket number
 - '5256' is the ticket number
 - Used to keep track of parameters and statistics

Shaft Password Protection

- A simple letter-shifting Caesar cipher is used
 - The password "shift" is bumped to either "rghes" or "tijgu"

Mstream

- A very primitive and poorly written DoS tool, obviously in the early stages of development
 - Despite this, it was disruptive to the victim and the network
- Unleashed in April of 2000

Shaft Statistics

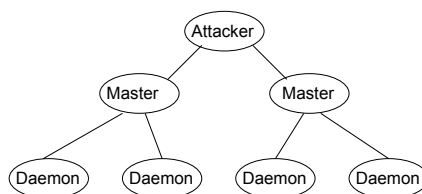
- Unlike many other DoS tools, the author of Shaft paid special attention to statistics
- It is possible to request that a certain victim return statistics about a current attack
- This prevents over-allocation of resources

Shaft Defenses

- Detection based on certain keywords using 'ngrep' (i.e. the passwords)
- Search for specific known sequence numbers
- Scan the network for open port 20432
- Send out an 'alive' message on the correct port and look for responses

Mstream Structure

- The structure is that of the bi-directional model, with a tree of control



Mstream Communication

- Attacker and handler communicate over TCP
- Handler/Agent communicate over UDP
- Commands must be contained entirely in the payload and can not be broken up
 - i.e. No telnet sessions

Mstream Commands

- When a daemon starts, it sends a “newserver” back to the master, which maintains a list of functional daemons
- Mstream attacks with a TCP ACK flood

Mstream Commands

- Commands between the handler and attacker are space separated
- Commands between the handler and agent are slash separated
- Very primitive commands
 - stream <hostname> <seconds>
 - Starts a DoS attack against the specified host for the time specified.

Mstream Password Protection

- Mstream is unique in its password protection
 - Unlike other DDoS tools, here, handlers are informed of access, successful or not, by competing parties
 - Can't simply hijack a session
- Despite this, non-encrypted passwords are used

Mstream Weaknesses

- Bad commands will cause a seg fault
- Flooding will saturate the file handle limit and cause it to become unresponsive
- It is Single-threaded, so it can't process incoming commands while in mid-flood
- All hosts are flooded equally for the same amount of time

Worm Propagation

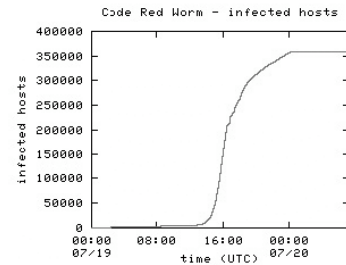
- A key to the success of a worm is how quickly and widely it is able to spread
- Code Red Worm mirrors the classical epidemic model
 - Description of worm
 - Problem with existing model
 - Proposed two-factor worm model
- Potential modifications to make future worms propagate even more efficiently

Code Red

- Started January 19, 2001
- Attacked a vulnerability in Windows IIS
- Started up 100 threads
 - 99 would randomly pick IP addresses to attempt to infect
 - 1 would deface the a website on the server

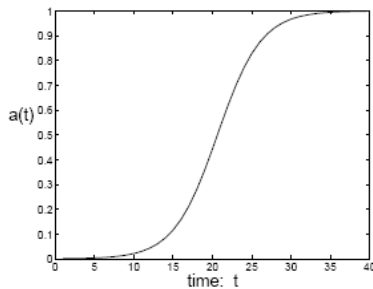
Code Red

- Here's what the propagation pattern looked like:



Code Red

- Here is a classical epidemic model
 - Notice the differences from the previous plot

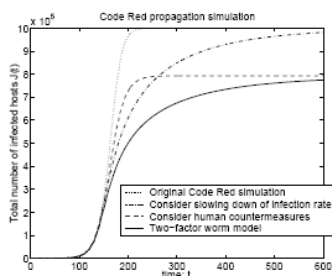


Problems with Epidemic Model

- Does not take into account human countermeasures
 - Removing susceptible computers from internet, patching them, filtering traffic, etc.
- Assumes a constant growth rate
 - In real life, the Code Red scanning process slowed down routers, thus, the rate of infection trailed off over time
 - Begins to slow down when only 50% of hosts are infected

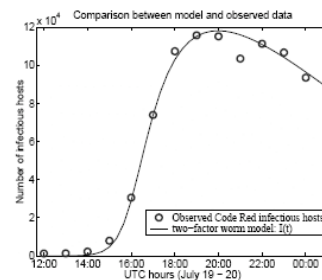
Two-Factor Propagation Model

- A new model was created that addresses both of these issues.



Two-Factor Propagation Model

- Comparison of the model with the actual data from the Code Red attack.

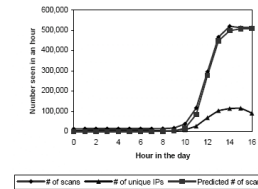


How to further speed up propagation...

- Although the Slammer worm and the Code Red worm were really fast, several propositions have been made about how to make a worm spread EVEN FASTER
 - Hit-list scanning
 - Permutation scanning
 - Topological scanning

Hit-list Scanning

- One problem that worms face is getting off of the ground
 - Worms spread exponentially, but they need time to infect the first ~10,000 hosts



Hit-list Scanning

- To overcome this, the worm author can collect a list of potentially vulnerable machines (10,000 – 50,000)
- When a vulnerable machine is found, the list is divided in half, with the original machine keeping half and continuing to search, and the newly-infected machine getting the other half.

Permutation Scanning

- Another common problem is that many addresses are probed multiple times
 - Slammer faced this
- So, if worms were able to avoid attempting to infect previously infected machines, the worm would be able to spread much more rapidly

Permutation Scanning

- All worms would share a common permutation of the IP address space
- If a worm attempted to infect a host and realized that it was already infected, it would know that a different instance of the worm had already covered the current and all following permutations of the IP address
- It would then reseed itself and start on a new branch of IP addresses
- When numerous conflicts are reached, the worm could shut down

Topological Scanning

- Topological scanning involves using information contained in the victim's machine to select new targets
 - Once these local machines were infected, the worm could proceed to permutation scanning.

Conclusion

- Worms are scary
- They are only going to get scarier
 - Faster
 - Potential to be more damaging
 - More far-reaching
 - More difficult to detect
- We need to automate worm detection, since humans are not going to be fast enough

References

- Inside the Slammer Worm, D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, N. Weaver, IEEE Security and Privacy, pages 33-39, July/August 2003.
- Analyzing Distributed Denial Of Service Tools: The Shaft Case, Sven Dietrich, Neil Long, David Dittrich, Proceedings of the 14th Systems Administration Conference (LISA 2000), New Orleans, Louisiana, December 2000, pages 329-339.
- The DoS Project's "trinoo" distributed denial of service attack tool David Dittrich, University of Washington, <http://staff.washington.edu/dittrich/misc/ddos/>, October 1999.
- The "mstream" distributed denial of service attack tool, David Dittrich, George Weaver, Sven Dietrich, Neil Long, University of Washington, <http://staff.washington.edu/dittrich/misc/ddos/>, May 2000.
- A Worst-Case Worm, N. Weaver, V. Paxson, Proceedings, Third Annual Workshop on Economics and Information Security (WEIS04), May 2004
- A Taxonomy of Computer Worms, Nicholas Weaver, Vern Paxson, Stuart Staniford, Robert Cunningham, Proceedings of the 2003 ACM workshop on Rapid Malcode, Washington, DC, October 2003, pages 11-18.
- How to Own the Internet in Your Spare Time, Stuart Staniford, Vern Paxson, Nicholas Weaver Proceedings of the 11th USENIX Security Symposium, December 2002.
- Code Red Worm Propagation Modeling and Analysis, C. C. Zou, W. Gong, and D. Towsley, Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, November 2002, pages 138-147.