CSci 5271 Introduction to Computer Security Malware and Denial of Service

Stephen McCamant University of Minnesota, Computer Science & Engineering

Outline

Malware and the network

Announcements intermission

Denial of service and the network

Malicious software

- Shortened to Mal... ware
- Software whose inherent goal is malicious Not just used for bad purposes
- Strong adversary
- 🖲 High visibility
- Many types

Trojan (horse)

Looks benign, has secret malicious functionality
 Key technique: fool users into installing/running
 Concern dates back to 1970s, MLS

(Computer) viruses

- Attaches itself to other software
- Propagates when that program runs
- once upon a time: floppy disks
- More modern: macro viruses
- Have declined in relative importance

Worms

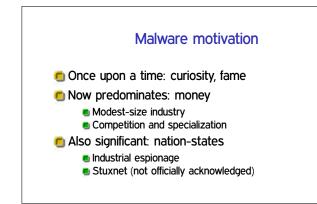
- Completely automatic self-propagation
- 🖲 Requires remote security holes
- 🖲 Classic example: 1988 Morris worm
- 🖲 "Golden age" in early 2000s
- Internet-level threat seems to have declined

Fast worm propagation Initial hit-list Pre-scan list of likely targets Accelerate cold-start phase Accelerate cold-start phase Systematic but not obviously patterned Systematic but not obviously patterned Pseudorandom permutation Approximate time: 15 minutes Warhol worm Too fast for human-in-the-loop response

Lower-level/higher-privilege code can deceive normal code

Getting underneath

- Rootkit: hide malware by changing kernel behavior
- MBR virus: take control early in boot
- Blue-pill attack: malware is a VMM running your system



User-based monetization

- 🖲 Adware, mild spyware
- Keyloggers, stealing financial credentials

Ransomware

- Application of public-key encryption
- Malware encrypts user files
- Only \$300 for decryption key

Bots and botnets

Bot: program under control of remote attacker

- Botnet: large group of bot-infected computers with common "master"
- Command & control network protocol
 - Once upon a time: IRC
 - Now more likely custom and obfuscated
 - **•** Centralized \rightarrow peer-to-peer
 - Gradually learning crypto and protocol lessons

Bot monetization

- 🖲 Click (ad) fraud
- Distributed DoS (next section)
- 🖲 Bitcoin mining
- 🖲 Pay-per-install (subcontracting)
- 🖲 Spam sending

Malware/anti-virus arms race

- "Anti-virus" (AV) systems are really general anti-malware
- 🖲 Clear need, but hard to do well
- No clear distinction between benign and malicious
- Endless possibilities for deception

Signature-based AV



Would work well if malware were static

🖲 In reality:

- 🖲 Large, changing database
- Frequent updated from analysts
- Not just software, a subscription
- Malware stays enough ahead to survive

Emulation and AV Simple idea: run sample, see if it does something evil Obvious limitation: how long do you wait? Simple version can be applied online More sophisticated emulators/VMs used in backend analysis

Polymorphism

- Attacker makes many variants of starting malware
- Different code sequences, same behavior
- One estimate: 30 million samples observed in 2012
- But could create more if needed

Packing

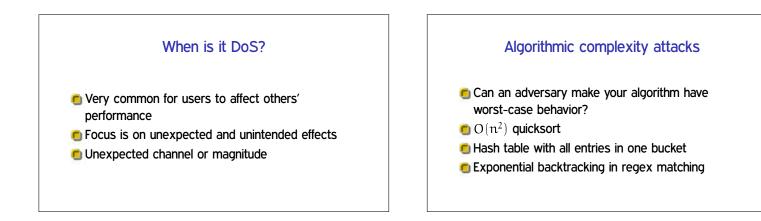
Sounds like compression, but real goal is obfuscation

- Static code creates real code on the fly
- Or, obfuscated bytecode interpreter
- Outsourced to independent "protection" tools

Fake anti-virus

- Major monentization strategy recently
- Your system is infected, pay \$19.95 for cleanup tool
- For user, not fundamentally distinguishable from real AV

Outline Note to early readers Malware and the network This is the section of the slides most likely to change in the final version Announcements intermission If class has already happened, make sure you have the latest slides for announcements Denial of service and the network Outline DoS versus other vulnerabilities Effect: normal operations merely become impossible Malware and the network Software example: crash as opposed to code injection Announcements intermission Less power that complete compromise, but practical severity can vary widely Denial of service and the network Airplane control DoS, etc.



XML entity expansion

SML entities (c.f. HTML <) are like C macros

#define B (A+A+A+A+A)
#define C (B+B+B+B+B)
#define D (C+C+C+C+C)
#define E (D+D+D+D+D)
#define F (E+E+E+E+E)

Compression DoS

- Some formats allow very high compression ratios
 Simple attack: compress very large input
- 🖲 More powerful: nested archives
- Also possible: "zip file quine" decompresses to itself

DoS against network services Common example: keep legitimate users from viewing a web site Easy case: pre-forked server supports 100 simultaneous connections Fill them with very very slow downloads Tiny bit of queueing theory Mathematical theory of waiting in line Simple case: random arrival, sequential fixed-time service M/D/1 If arrival rate ≥ service rate, expected queue length grows without bound

SYN flooding

- SYN is first of three packets to set up new connection
- Traditional implementation allocates space for control data
- However much you allow, attacker fills with unfinished connections
- Early limits were very low (10-100)

SYN cookies

- Change server behavior to stateless approach
- Embed small amount of needed information in fields that will be echoed in third packet
 MAC-like construction
- Other disadvantages, so usual implementations used only under attack

DoS against network links Try to use all available bandwidth, crowd out real traffic Brute force but still potentially effective Baseline attacker power measured by packet sending rate

Traffic multipliers

- Third party networks (not attacker or victim)
- One input packet causes n output packets
- Commonly, victim's address is forged source, multiply replies
- Misuse of debugging features

"Smurf" broadcast ping

ICMP echo request with forged source
 Sent to a network broadcast address
 Every recipient sends reply

Now mostly fixed by disabling this feature

Distributed DoS

- Many attacker machines, one victim
- 🖲 Easy if you own a botnet
- Impractical to stop bots one-by-one
- May prefer legitimate-looking traffic over weird attacks
 - Main consideration is difficulty to filter

Next time

Network anonymity with overlay networks