# An Overview of Network Security

**Vern Paxson**

***International Computer Science Institute / Lawrence Berkeley National Laboratory / EECS, UC Berkeley***

**vern@icsi.berkeley.edu / vern@ee.lbl.gov**

**April 18, 2006**

## Security Dimensions

- General notion: network security = ensuring that a network is used as desired/intended
- Authentication: who is this actor?
  - Attacker counterpart: *spoofing*
- Authorization: is this actor allowed to do what they request?
  - Attacker counterpart: *compromise*
- Accountability/Attribution: who did this activity?
  - Attacker counterpart: *framing* (not discussed today)

## Security Dimensions, con't

- Integrity: does a message arrive in its original form?
  - Protections are cryptographic (not discussed today)
- Confidentiality: is communication free from eavesdropping?
  - Same: cryptographic protections (not discussed today)
  - Attacker counterpart: *sniffing, man-in-the-middle*
- Availability: can you use the network / a service when you want to?
  - Attacker counterpart: *Denial-of-Service* (DoS), *theft-of-service*
- Audit/forensics: what occurred in the past?
- Abuse: misuse that doesn't violate the rules (e.g., spam)

## Network Security Themes

- Much of the field has evolved in an *ad hoc* manner
- Security is about *policy*, not about *bullet-proofing*
  - Much of the effort concerns "raising the bar" and *trading off resources*
- Threat model: what you are defending against
  - E.g., UCB: SB1386 personal identity information disclosure
  - E.g., LBL: embarrassing newspaper articles ☑ DC $$$
- Networks connect *disparate parties*
  - They have different notions of policy and threat models
  - Crucial to keep in mind domains of trust, responsibility, control
  - Many will not cooperate unless it's in their business interest

## Network Security Themes, con’t

- Network use is always more diverse than you expect
  - Zillions of applications/services
  - Rife with weird/broken traffic (“crud”)
  - Rife with “background radiation” (incessant probing for vuln.)
  - Breadth of diversity increases with size of user base
  - Exacerbates the problem of “false positives”
- The problem is fundamentally *adversarial*
  - This can radically change system design considerations …
  - … but keep in mind “raising the bar” and threat model vs. bullet-proofing
- Major challenge of *manageability*
  - Complex policies + *churn* + false positives + zillions of devices + “bolt-on” mechanisms + innovating attackers = HUGE challenges

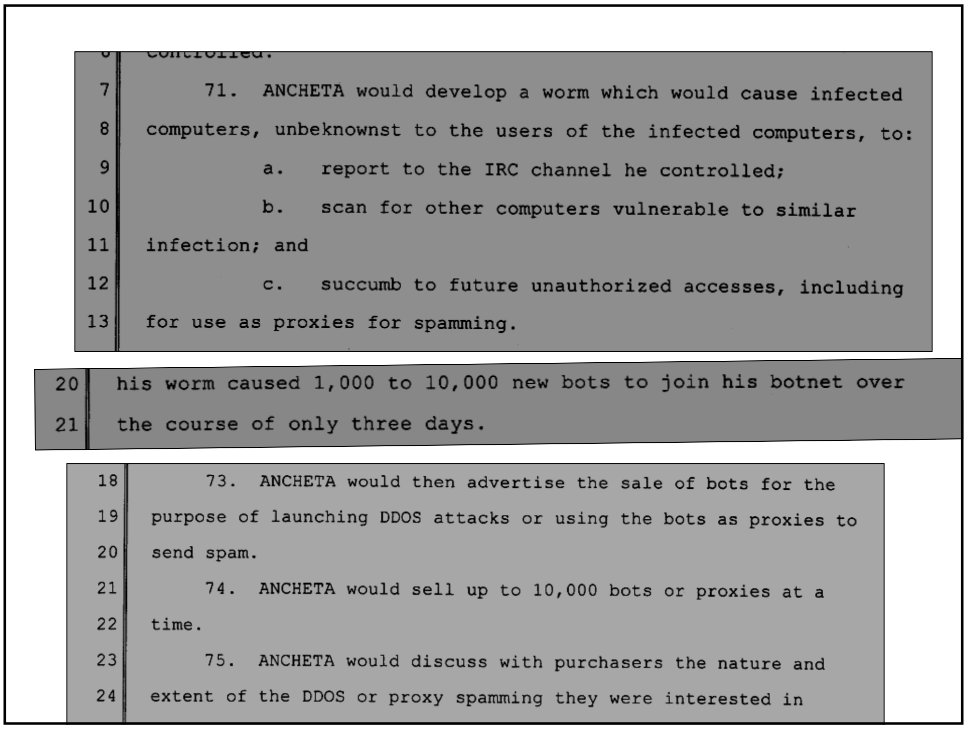
## Who Are The Bad Guys, Historically?

- Vandals/juveniles:
  - In it for showing off, kicks
    - Often can catch break-ins because they set up IRC servers
  - Historically, hugely deriviative (“script kiddies”) with slow pace of innovation
  - Historically, very prevalent in over-the-network attacks, viruses
- Insiders
  - Already have some form of site access

    ➔ Powerful & feared
  - Historically, under-reported
  - Threat includes “exfiltration” of sensitive information

# Who Are The Bad Guys, Today?

- (Not: terrorists, political protesters)
- Espionage
  - Theft of information for commercial / national gain
- Militaries
  - Very hard to gauge, but clearly an area of activity
  - Our study of a “worse case” worm attack launched by a nation state yielded defensible $50B+ damages
- Crooks …



controlled.

71. ANCHETA would develop a worm which would cause infected computers, unbeknownst to the users of the infected computers, to:

a. report to the IRC channel he controlled;

b. scan for other computers vulnerable to similar infection; and

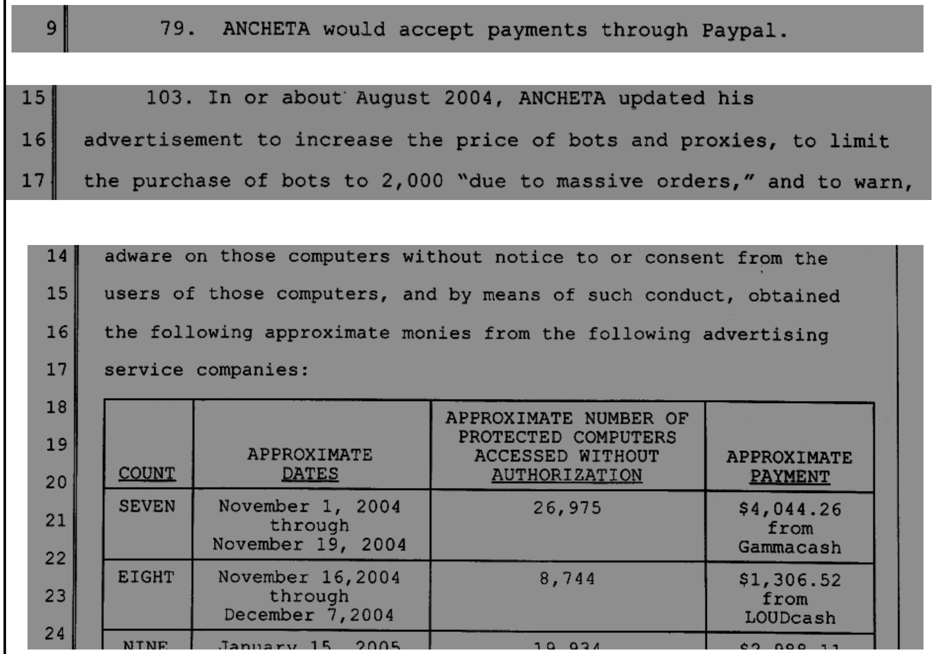
c. succumb to future unauthorized accesses, including for use as proxies for spamming.

his worm caused 1,000 to 10,000 new bots to join his botnet over the course of only three days.

73. ANCHETA would then advertise the sale of bots for the purpose of launching DDOS attacks or using the bots as proxies to send spam.

74. ANCHETA would sell up to 10,000 bots or proxies at a time.

75. ANCHETA would discuss with purchasers the nature and extent of the DDOS or proxy spamming they were interested in

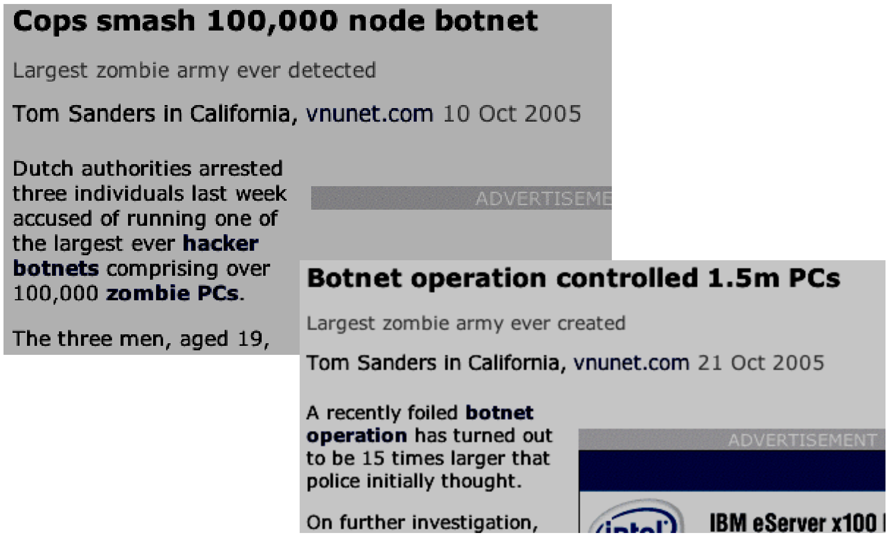


79. ANCHETA would accept payments through Paypal.

103. In or about August 2004, ANCHETA updated his advertisement to increase the price of bots and proxies, to limit the purchase of bots to 2,000 "due to massive orders," and to warn,

adware on those computers without notice to or consent from the users of those computers, and by means of such conduct, obtained the following approximate monies from the following advertising service companies:

| COUNT | APPROXIMATE DATES | APPROXIMATE NUMBER OF PROTECTED COMPUTERS ACCESSED WITHOUT AUTHORIZATION | APPROXIMATE PAYMENT |
|---|---|---|---|
| SEVEN | November 1, 2004 through November 19, 2004 | 26,975 | $4,044.26 from Gammacash |
| EIGHT | November 16,2004 through December 7,2004 | 8,744 | $1,306.52 from LOUDcash |
| NINE | January 15, 2005 | 19,934 | $2,988.11 |



## Cops smash 100,000 node botnet

Largest zombie army ever detected

Tom Sanders in California, vnunet.com 10 Oct 2005

Dutch authorities arrested three individuals last week accused of running one of the largest ever **hacker botnets** comprising over 100,000 **zombie PCs**.

The three men, aged 19,

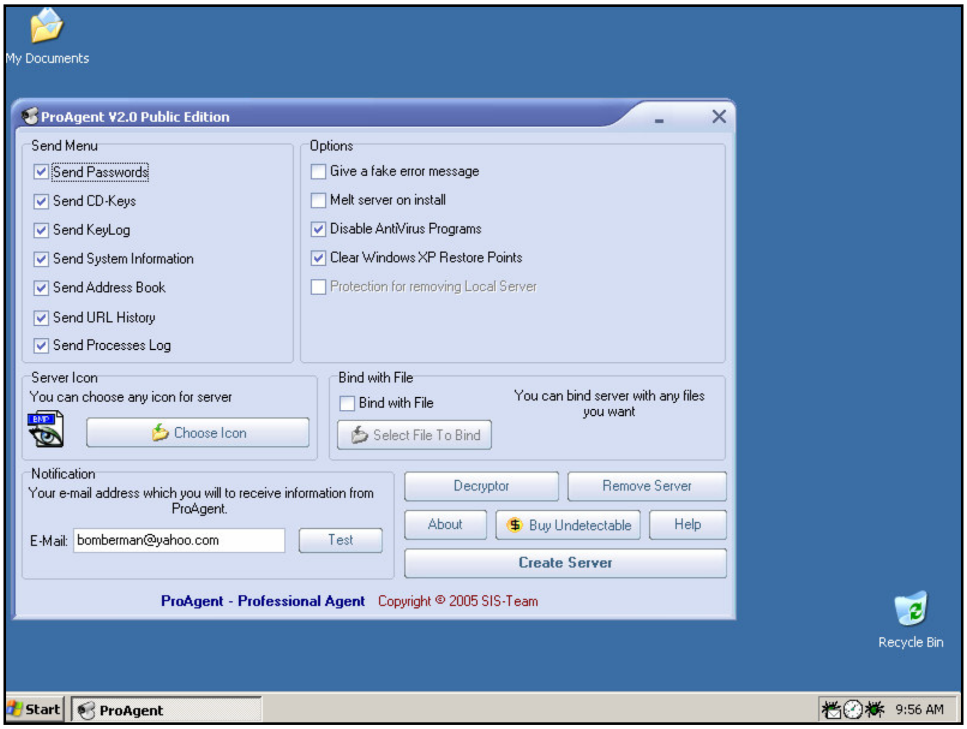
## Botnet operation controlled 1.5m PCs

Largest zombie army ever created

Tom Sanders in California, vnunet.com 21 Oct 2005

A recently foiled **botnet operation** has turned out to be 15 times larger that police initially thought.

On further investigation,



ProAgent V2.0 Public Edition

Send Menu

- ☑ Send Passwords
- ☑ Send CD-Keys
- ☑ Send KeyLog
- ☑ Send System Information
- ☑ Send Address Book
- ☑ Send URL History
- ☑ Send Processes Log

Options

- □ Give a fake error message
- □ Melt server on install
- ☑ Disable AntiVirus Programs
- ☑ Clear Windows XP Restore Points
- □ Protection for removing Local Server

Server Icon

You can choose any icon for server

Bind with File

□ Bind with File

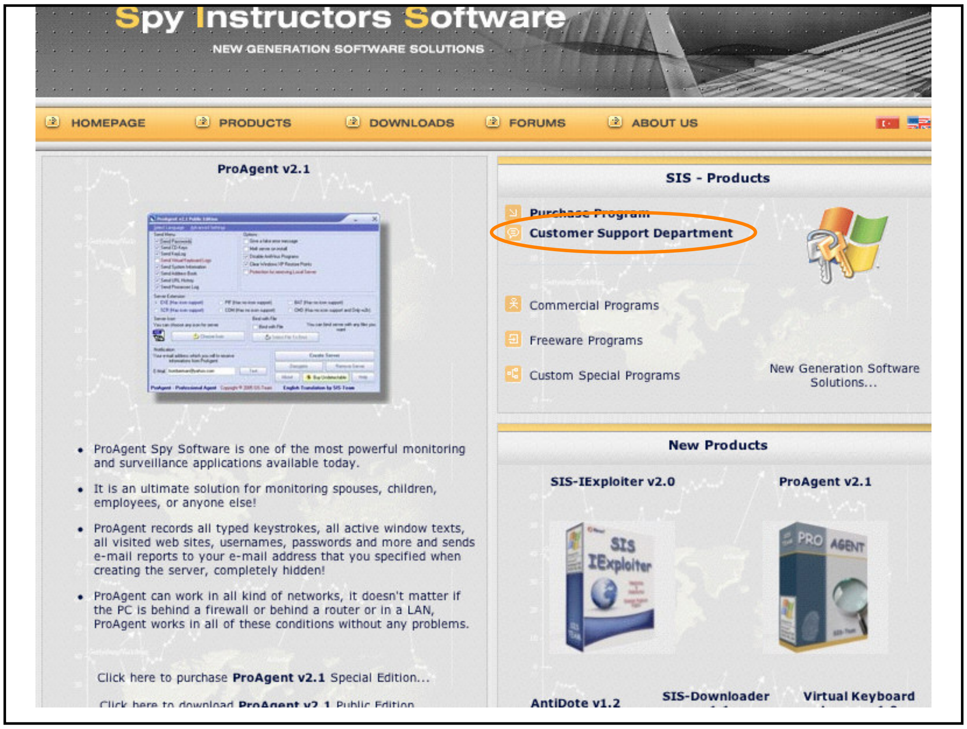
You can bind server with any files you want

Notification

Your e-mail address which you will to receive information from ProAgent.

E-Mail: bomberman@yahoo.com

ProAgent - Professional Agent Copyright © 2005 SIS-Team



Spy Instructors Software

NEW GENERATION SOFTWARE SOLUTIONS

ProAgent v2.1

- ProAgent Spy Software is one of the most powerful monitoring and surveillance applications available today.
- It is an ultimate solution for monitoring spouses, children, employees, or anyone else!
- ProAgent records all typed keystrokes, all active window texts, all visited web sites, usernames, passwords and more and sends e-mail reports to your e-mail address that you specified when creating the server, completely hidden!
- ProAgent can work in all kind of networks, it doesn't matter if the PC is behind a firewall or behind a router or in a LAN, ProAgent works in all of these conditions without any problems.

Click here to purchase **ProAgent v2.1** Special Edition...

Click here to download **ProAgent v2.1** Public Edition

SIS - Products

Purchase Program

Customer Support Department

Commercial Programs

Freeware Programs

Custom Special Programs

New Generation Software Solutions...

New Products

SIS-IExploiter v2.0

ProAgent v2.1

AntiDote v1.2

SIS-Downloader

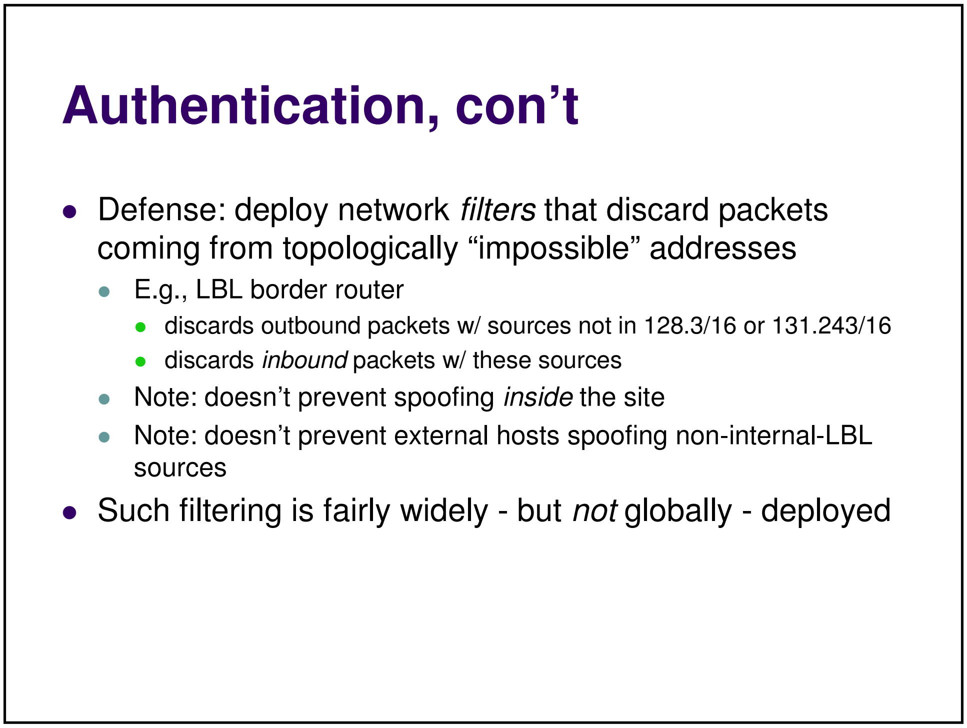
Virtual Keyboard

# Who Are The Bad Guys, Today?

- (Not: terrorists, political protesters)
- Espionage
  - Theft of information for commercial / national gain
- Militaries
  - Very hard to gauge, but clearly an area of activity
  - Our study of a “worse case” worm attack launched by a nation state yielded defensible $50B+ damages
- Crooks
  - Very worrisome trend in attackers figuring out how to **make money** with network attacks
  - Fuels *innovation* and *specialization*, driving an *economy*

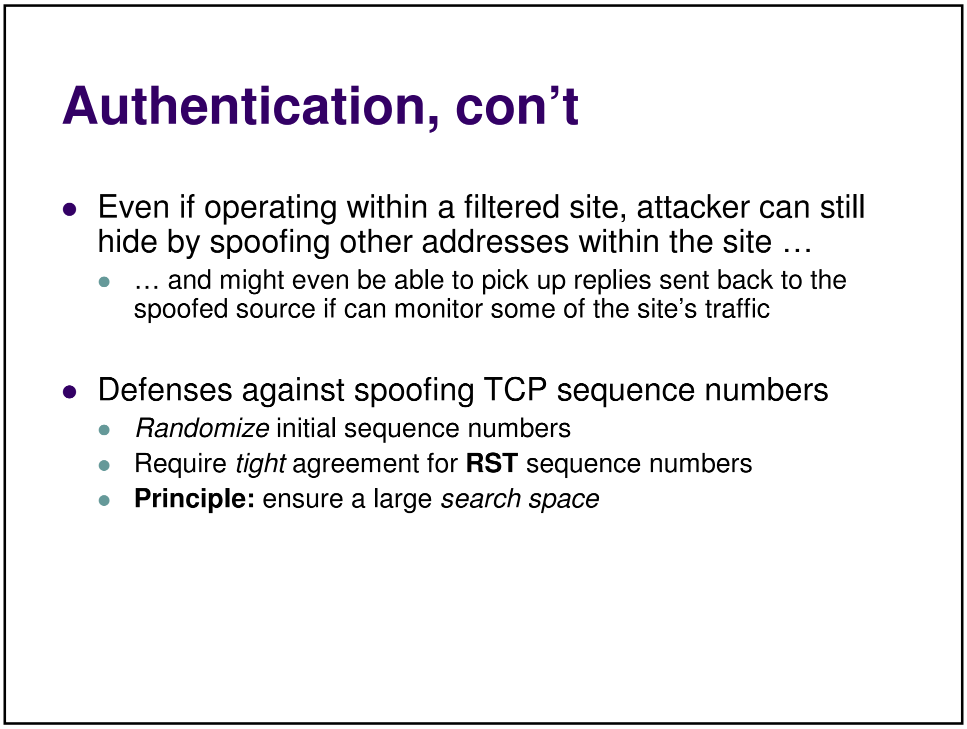
# Authentication: Who is this Actor?

- Notion is absent from Internet architecture
  - It ensures that packets go to their destination addresses …
  - … but **not** that they came from their source addresses
  - Yet, absent an alternative, much authorization is based on source address
- What can an attacker achieve by spoofing a source address?
  - Denial-of-service floods that
    - Can’t be attributed to the machine sending them
    - Can’t be filtered based on their source
  - Impersonation of other machines
    - Tear-down established connections via TCP **RST**s
    - *Establish* connections if can guess TCP *Initial Sequence Number*
    - Devious stealth-*scanning* that looks like it comes from someone else



## Authentication, con’t

- Defense: deploy network *filters* that discard packets coming from topologically “impossible” addresses
  - E.g., LBL border router
    - discards outbound packets w/ sources not in 128.3/16 or 131.243/16
    - discards *inbound* packets w/ these sources
  - Note: doesn’t prevent spoofing *inside* the site
  - Note: doesn’t prevent external hosts spoofing non-internal-LBL sources
- Such filtering is fairly widely - but *not* globally - deployed



## Authentication, con’t

- Even if operating within a filtered site, attacker can still hide by spoofing other addresses within the site …
  - … and might even be able to pick up replies sent back to the spoofed source if can monitor some of the site’s traffic
- Defenses against spoofing TCP sequence numbers
  - *Randomize* initial sequence numbers
  - Require *tight* agreement for **RST** sequence numbers
  - **Principle:** ensure a large *search space*

## Social Engineering: Confusing Humans Regarding Authentication

- Attacks on DNS names
  - E.g., register www.gooogle.com
  - Now passively wait for someone to mistype …
  - … and feed them whatever fake Google experience you wish
- Attacks on DNS reverse lookups
  - E.g., you receive a packet from 1.2.3.4. *Who is that?*
  - If you look up the corresponding hostname, you really are querying **4.3.2.1.in-addr.arpa**.
  - Whoever controls the corresponding name server can return whatever they like.
  - Suppose this name server is at 1.2.3.10, and an attacker has compromised both 1.2.3.4 and 1.2.3.10.
  - Then the answer returned might well be *www.google.com*

## Social Engineering, con’t

- Powerful technique for *targeted attacks*
  - E.g., find out the name and mailstop of one of a company’s system administrators …
  - … mail out a CD of a trojaned system image to a company employee with a note that it contains an important security update.
  - Employee trusts the source of the update, applies it, and now you have a backdoor of arbitrary design into the company
  - Attacks like this are well known to **often work**

- More generally, the (very big) problem of ***phishing*** is an instance of ongoing social engineering attacks.
- General defense: user education :-(
- Phishing-specific defenses: active area for startups

# Authorization: Is This Actor Allowed To Do What They Request?

- Much authorization is based on looking up *identity* in an **access control list** (ACL).
  - Hence, strength hinges on strength of authentication technology
- Firewalls: *inline* authorization enforcement mechanism
  - Can allow/disallow traffic based on IP addresses (“white lists” and “black lists”)
  - Can allow/disallow traffic based on TCP/UDP port numbers
- Latter assumes can service the **service** associated with a connection from the port number used by the server
  - Increasingly this is *no longer the case*
    - Adversarial applications: e.g., file-sharing, *Skype*
    - Use of **tunneling** to encapsulate one protocol within another

# Authorization Without Identity

- *Capabilities*: objects that convey authorization to do something by their very possession
- E.g., car keys
  - Possession enables operation of the car, regardless of identity
    - Capabilities can be *delegated*
  - It’s (very) hard to create the capability by guessing
  - You can make different ones with related but different properties (e.g., key starts the car and will/won’t open the trunk)
  - Capabilities can be *copied*
- Example from a network security context:
  - ***http://www.icir.org/vern/tmp/*** is an unlistable directory
    - If you know the full URL to an item in it, you can access it
    - But you likely can’t guess it
- However, for network security main interesting uses are hypothetical
  - We will revisit this when discussing denial-of-service defenses

# Authorization Without Identity, con’t

- *Network Access Control:* you’re only allowed to connect to a network if you demonstrate good security hygiene
  - E.g., when you try to access a wireless network, the access point *scans you* or contacts a server previously installed on your laptop or PDA
  - Scan must reveal up-to-date security patches, sound local access configuration, policy compliance
- Highly appealing as it addresses “loss of the perimeter”, i.e., that sites can no longer rely on controlling access to their resources by controlling their Internet link
- Lots of vendor buzz
  - Cisco: “Network Admission Control”
  - Microsoft: “Network Access Protection”

# Circumventing Authorization

- Major means of undermining authorization is **compromise**: tricking a host into executing on your behalf.
- We can think about these in terms of *what* is attacked (server or client) and the *semantic level* at which it is attacked
- Attacks on servers: client sends subversive requests
  - Happens at attacker’s choosing
- Attacks on clients: server (attacker) *waits* for client to connect, sends it subversive replies
  - Perhaps server “chums” to entice clients to visit (e.g., claiming to have popular pirated content)

## Semantic Level of Compromise

- *Buffer overflows*
  - Part of the request sent by the attacker too large to fit into buffer server uses to hold it.
  - Spills over into memory beyond the buffer
  - Can alter corresponding program state, particularly
    - Return address associated with current function call
    - Change this to branch into other overwritten memory, **executing the attacker's code**
  - Large class of attacks, with a variety of defenses
    - Host-based: randomized layouts, detection of overwritten memory, execution of network payload, impossible call stacks
    - Network-based: signatures, semantic analysis, post-attack activity
  - Violates semantics of underlying programming language

## Semantic Level of Compromise, con't

- *Logic errors*
  - E.g., suppose your Web server passes any argument named "**`rev`**" in a URL request to a backend script called *munch* via the equivalent of

    ```
            sh munch $rev
    ```

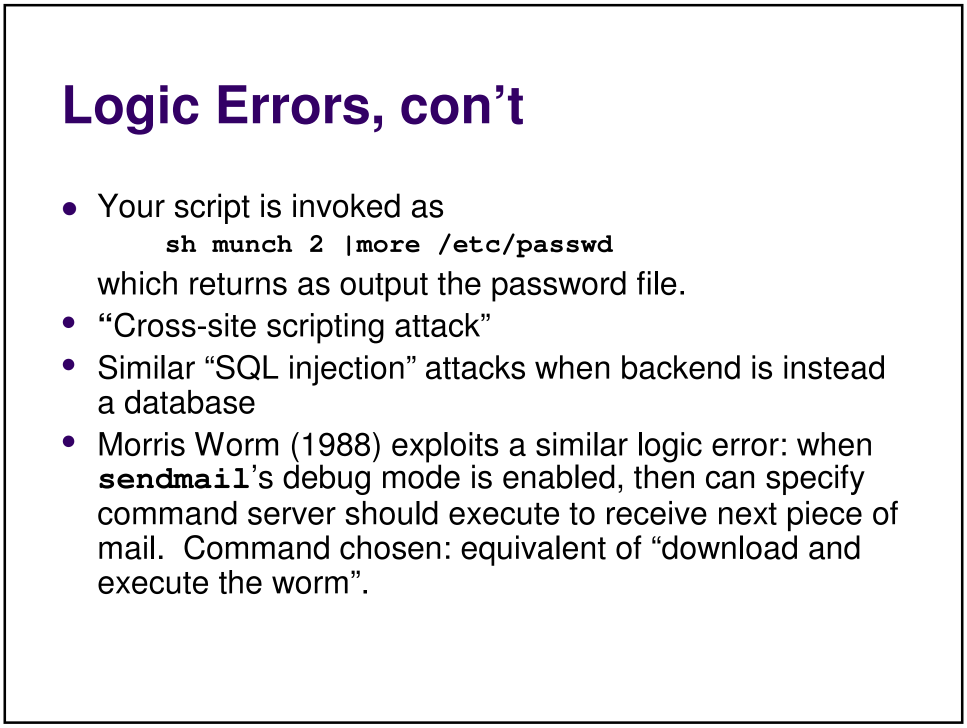
    where **`$rev`** is the URL argument, returning its output

  - Now suppose you receive the following request:

    ```
    GET /bin/TWikiUsers?rev=2%20|more%20/etc/passwd
    ```

    which decodes to

    ```
      $rev = "2 |more /etc/passwd"
    ```

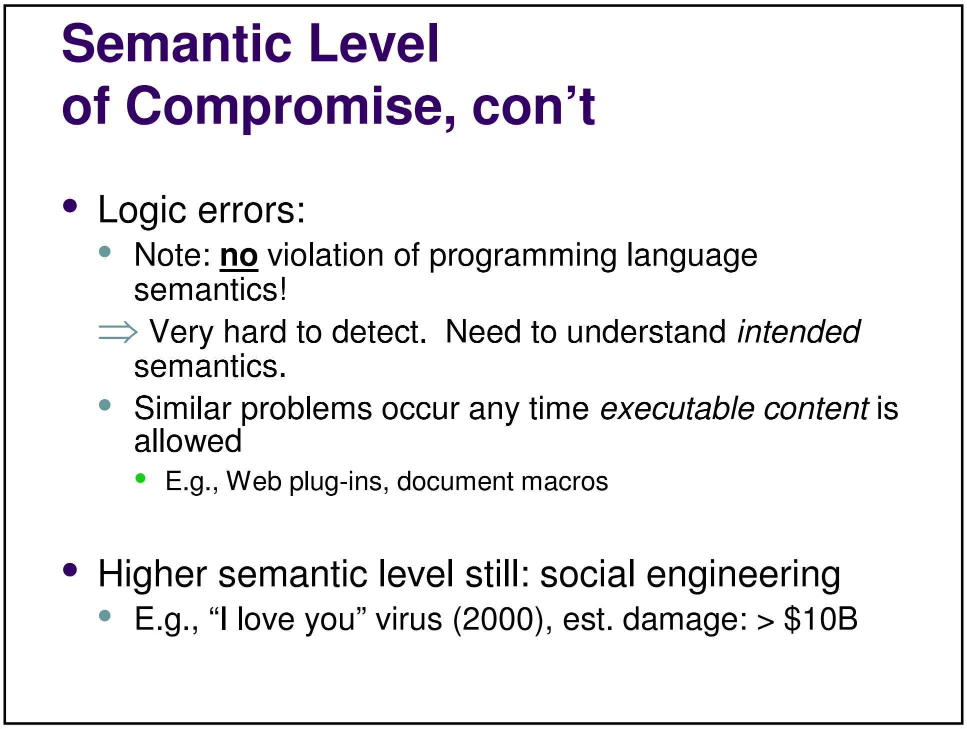


## Logic Errors, con’t

- Your script is invoked as

  ```
  sh munch 2 |more /etc/passwd
  ```

  which returns as output the password file.
- “Cross-site scripting attack”
- Similar “SQL injection” attacks when backend is instead a database
- Morris Worm (1988) exploits a similar logic error: when **`sendmail`**’s debug mode is enabled, then can specify command server should execute to receive next piece of mail. Command chosen: equivalent of “download and execute the worm”.

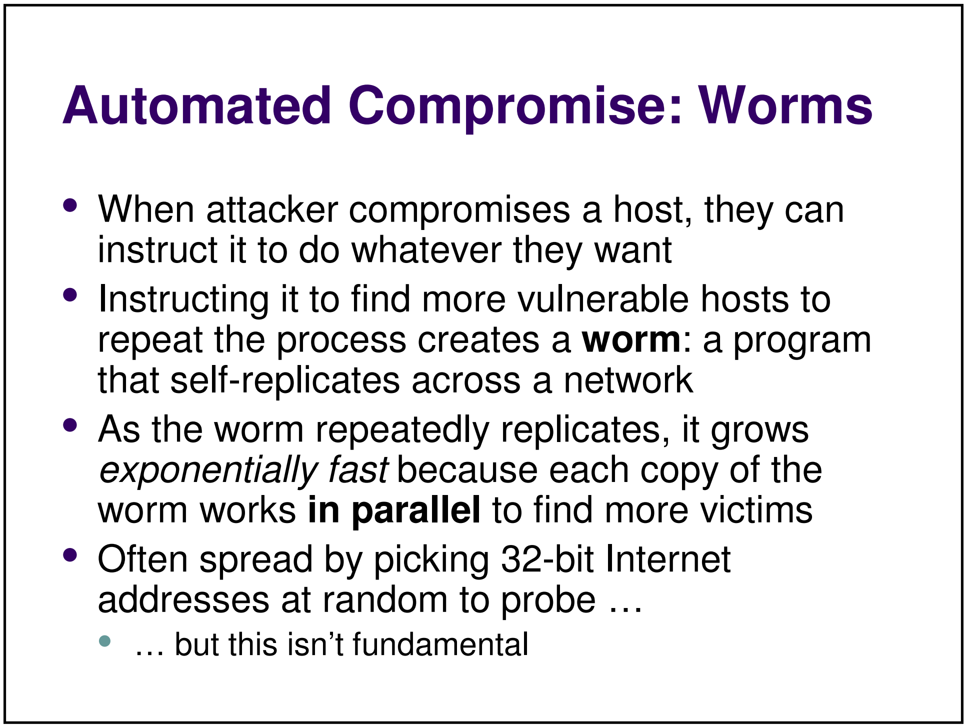


## Semantic Level of Compromise, con’t

- Logic errors:
  - Note: **<u>no</u>** violation of programming language semantics!

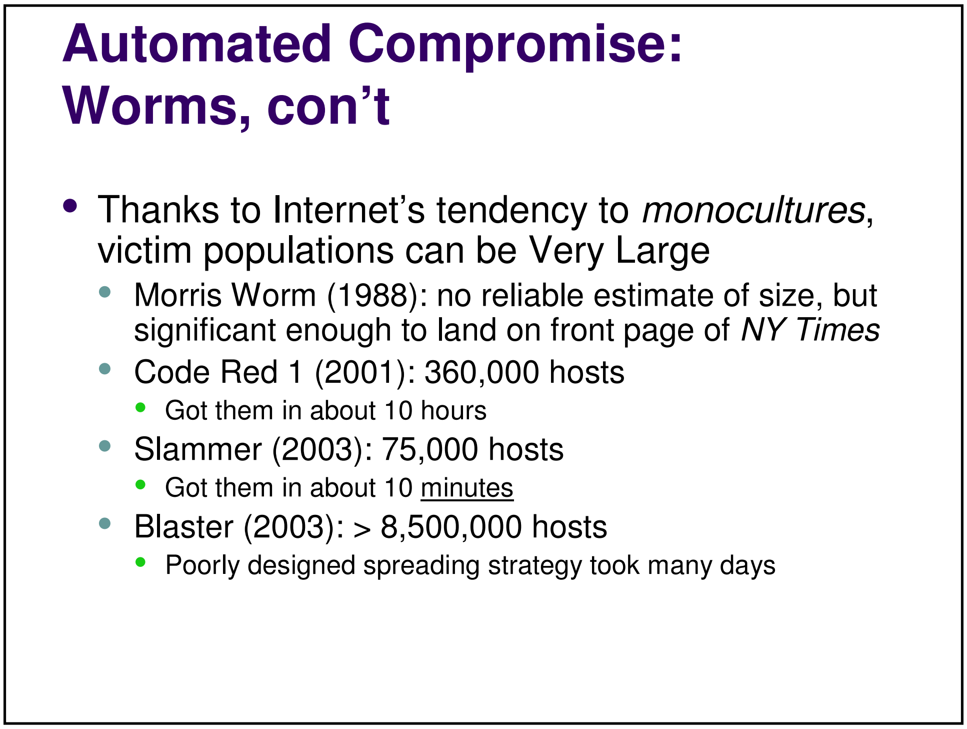
  ⟹ Very hard to detect. Need to understand *intended* semantics.
  - Similar problems occur any time *executable content* is allowed
    - E.g., Web plug-ins, document macros

- Higher semantic level still: social engineering
  - E.g., “I love you” virus (2000), est. damage: > $10B



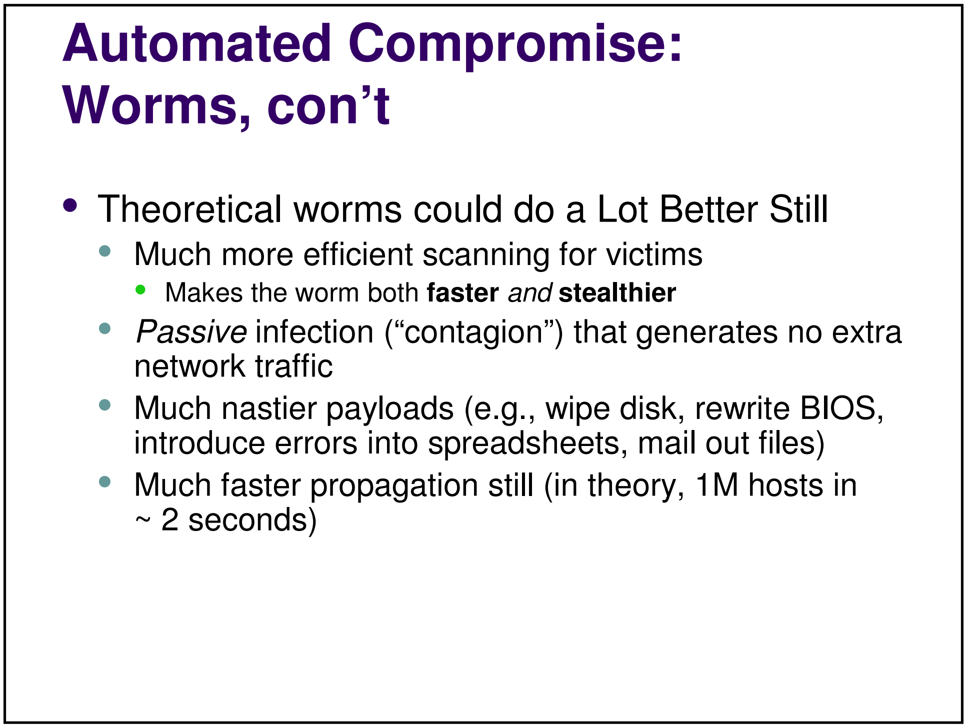
# Automated Compromise: Worms

- When attacker compromises a host, they can instruct it to do whatever they want
- Instructing it to find more vulnerable hosts to repeat the process creates a **worm**: a program that self-replicates across a network
- As the worm repeatedly replicates, it grows *exponentially fast* because each copy of the worm works **in parallel** to find more victims
- Often spread by picking 32-bit Internet addresses at random to probe …
  - … but this isn't fundamental



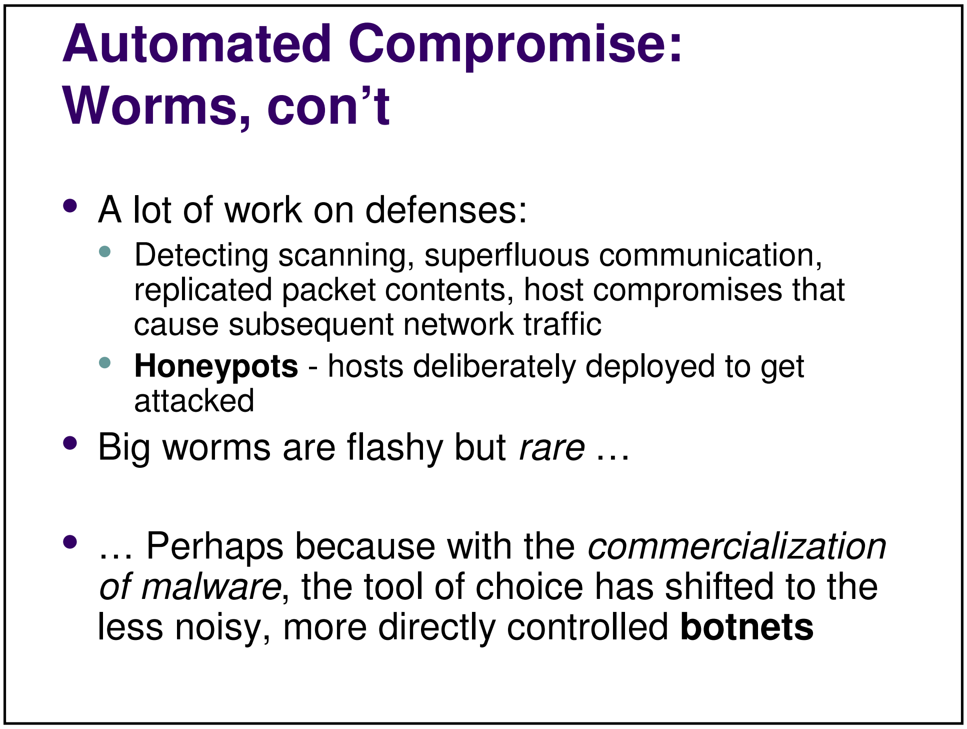
# Automated Compromise: Worms, con't

- Thanks to Internet's tendency to *monocultures*, victim populations can be Very Large
  - Morris Worm (1988): no reliable estimate of size, but significant enough to land on front page of *NY Times*
  - Code Red 1 (2001): 360,000 hosts
    - Got them in about 10 hours
  - Slammer (2003): 75,000 hosts
    - Got them in about 10 minutes
  - Blaster (2003): > 8,500,000 hosts
    - Poorly designed spreading strategy took many days



## Automated Compromise: Worms, con’t

- Theoretical worms could do a Lot Better Still
  - Much more efficient scanning for victims
    - Makes the worm both **faster** *and* **stealthier**
  - *Passive* infection (“contagion”) that generates no extra network traffic
  - Much nastier payloads (e.g., wipe disk, rewrite BIOS, introduce errors into spreadsheets, mail out files)
  - Much faster propagation still (in theory, 1M hosts in ~ 2 seconds)



## Automated Compromise: Worms, con’t

- A lot of work on defenses:
  - Detecting scanning, superfluous communication, replicated packet contents, host compromises that cause subsequent network traffic
  - **Honeypots** - hosts deliberately deployed to get attacked
- Big worms are flashy but *rare* …
- … Perhaps because with the *commercialization of malware*, the tool of choice has shifted to the less noisy, more directly controlled **botnets**

## Automated Compromise: Bots

- When host is (automatically) compromised, don’t continue propagation, but instead install a “command and control” *platform* (a **bot**)
- Note, can use a worm to get bots, can use a botnet to launch worms (or scan for more bots)
- Now can *monetize* malware by selling access to the bots
  - Spamming, phishing web sites, flooding attacks
  - “Crook’s Google Desktop”: sell capability of searching the contents of 100,000s of hosts

## Network Detection Of Attacks

- Far and away, most traffic travels across the Internet unencrypted.
- Communication is <u>layered</u> with higher layers corresponding to greater semantic content.
- The entire communication between two hosts can be reassembled: individual *packets* (e.g., TCP/IP headers), application *connections* (TCP byte streams), user *sessions* (Web surfing).
- You can do this in real-time.

## Tapping links, con’t:

- Appealing because it’s *cheap* and gives broad coverage.
- You can have multiple boxes watching the same traffic.
- Generally (not always) undetectable.
- Can also provide insight into a site’s general network use.

## Problems With Passive Monitoring

- Reactive, not proactive
  - However, this is changing w/ intrusion *prevention* systems
- Assumes network-oriented (often “external”) threat model.
- For high-speed links, monitor may not keep up.
- Depending on “vantage point”, sometimes you see only one side of a conversation (especially inside backbone).
- Against a skilled opponent, there is a fundamental problem of evasion: confusing / manipulating the monitor.

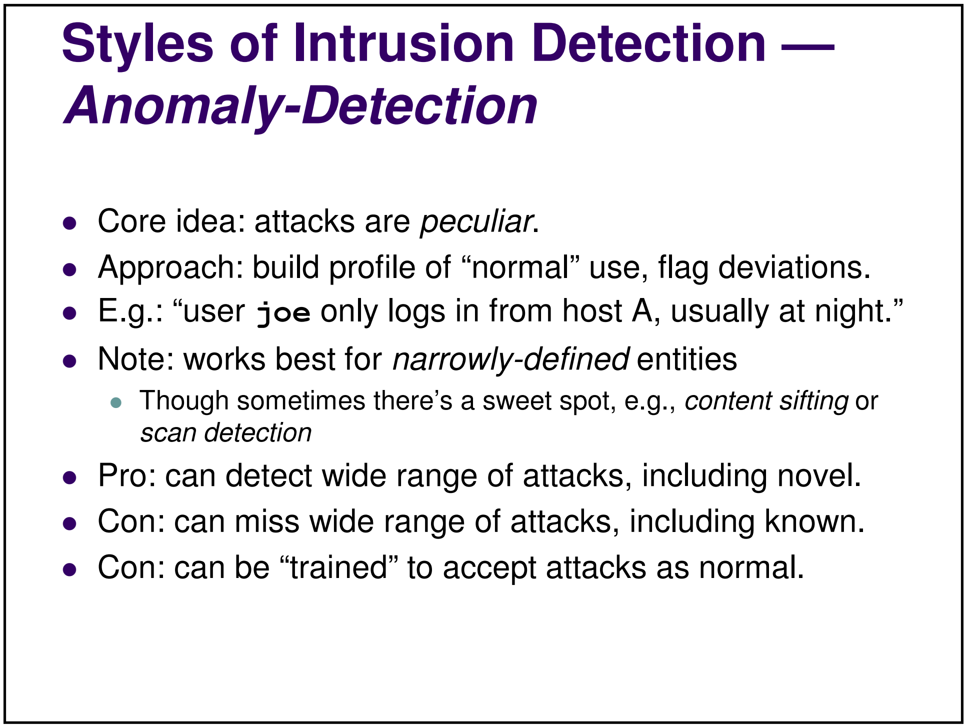
## Styles of Intrusion Detection — *Signature-Based*:

- Core idea: look for specific, known attacks.
- Example:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET
  139 flow:to_server,established
content:"|eb2f 5feb 4a5e 89fb 893e 89f2|"
msg:"EXPLOIT x86 linux samba overflow"
reference:bugtraq,1816
reference:cve,CVE-1999-0811
classtype:attempted-admin
```

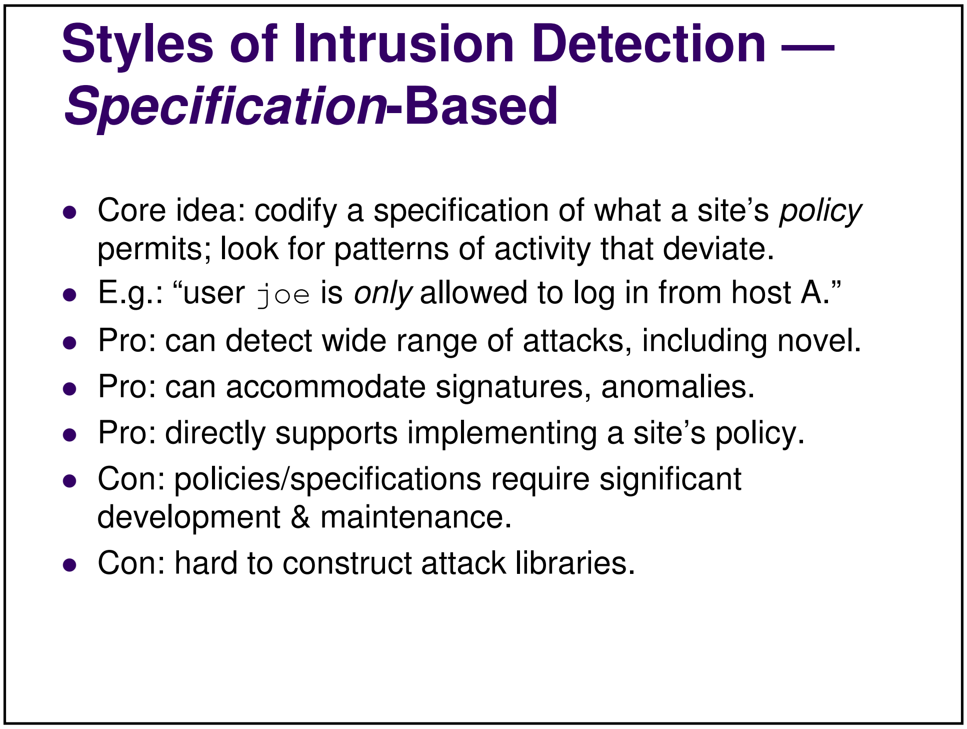
## Signature-Based, con't:

- Can be at different semantic layers, e.g.: IP/TCP header fields; packet payload; URLs.
    - Higher semantic levels yield more detection power & greater ability to avoid false positives (e.g., checking replies to requests).
- Pro: good attack libraries, easy to understand results.
- Con: unable to detect new attacks, or even just variants.  Low-level sigs prone to false positives.



## Styles of Intrusion Detection — *Anomaly-Detection*

- Core idea: attacks are *peculiar*.
- Approach: build profile of “normal” use, flag deviations.
- E.g.: “user **`joe`** only logs in from host A, usually at night.”
- Note: works best for *narrowly-defined* entities
  - Though sometimes there’s a sweet spot, e.g., *content sifting* or *scan detection*
- Pro: can detect wide range of attacks, including novel.
- Con: can miss wide range of attacks, including known.
- Con: can be “trained” to accept attacks as normal.



## Styles of Intrusion Detection — *Specification*-Based

- Core idea: codify a specification of what a site’s *policy* permits; look for patterns of activity that deviate.
- E.g.: “user `joe` is *only* allowed to log in from host A.”
- Pro: can detect wide range of attacks, including novel.
- Pro: can accommodate signatures, anomalies.
- Pro: directly supports implementing a site’s policy.
- Con: policies/specifications require significant development & maintenance.
- Con: hard to construct attack libraries.

# A Stitch in Time: Prevention Instead of Detection

- Big *win* to not just detect an attack, but **block it**
- However: *Big lose* to block **legitimate traffic**

- Mechanisms:
  - NIDS spoofs connection tear-down/denial messages
  - NIDS contacts firewall/router, requests block (race condition)
  - NIDS is *in-line* and itself drops offending traffic (no race, but performance and robustness issues)

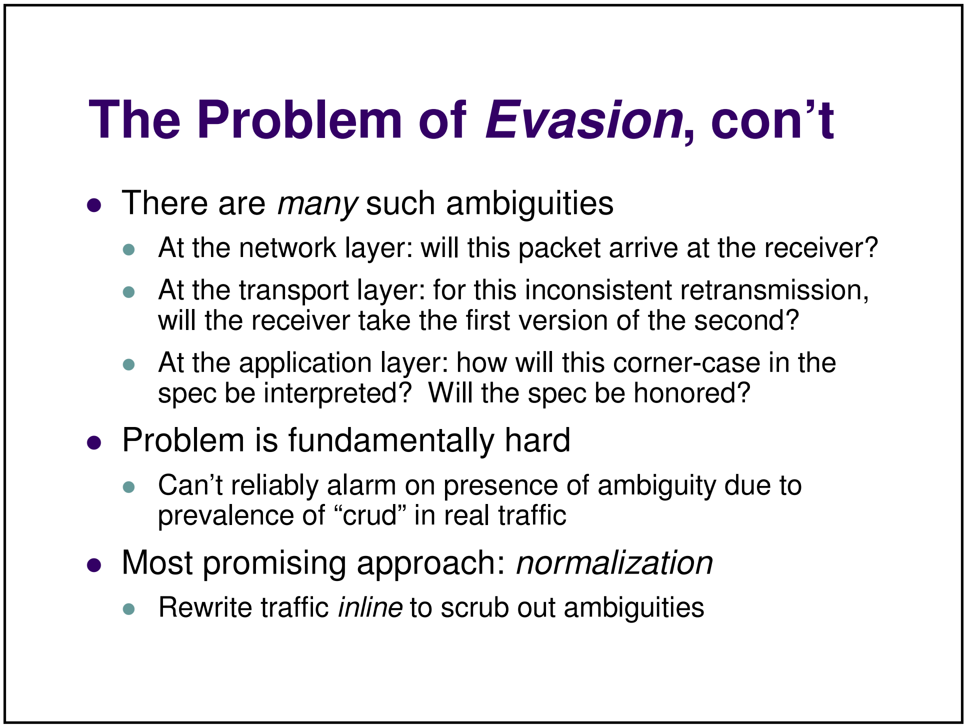
- Increasing trend in industry …
- … but requires highly accurate algorithms

# The Problem of *Evasion*

- Consider the following attack URL:
  http://…./c/winnt/system32/cmd.exe?/c+dir
- Easy enough to scan for (e.g., “cmd.exe”), right?

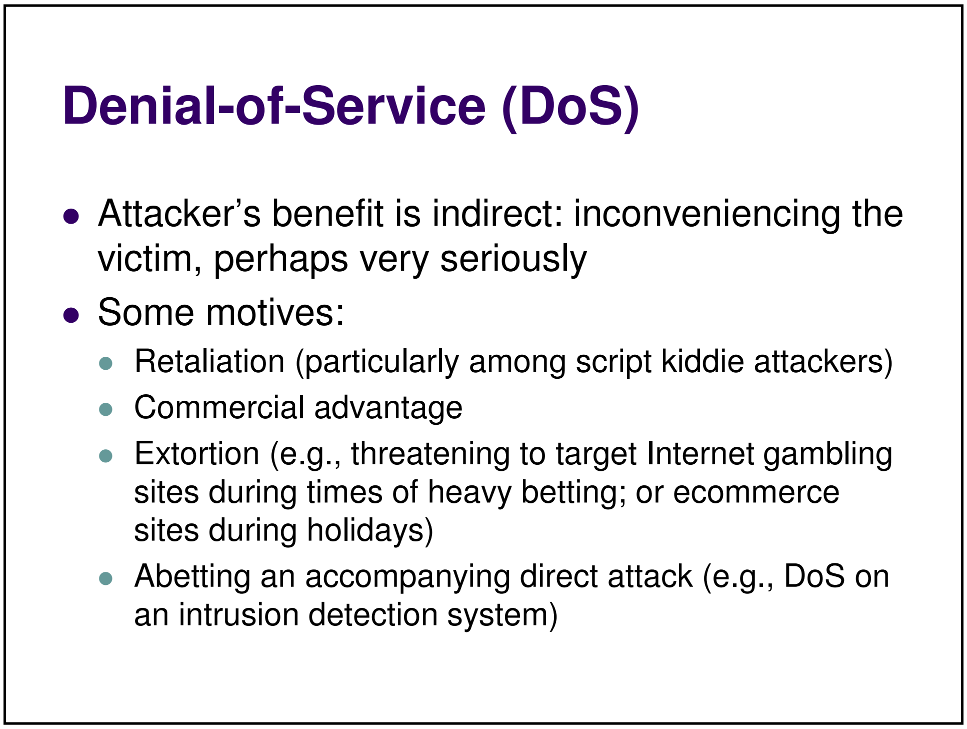
- But what about
  http://…./c/winnt/system32/cm%64.exe?/c+dir
- ▪ Okay, we need to handle % escapes.

- But what about
  http://…./c/winnt/system32/cm%25%54%52.exe?/c+dir
- ▪ Oops.  Will server double-expand escapes … or not?



## The Problem of *Evasion*, con't

- There are *many* such ambiguities
  - At the network layer: will this packet arrive at the receiver?
  - At the transport layer: for this inconsistent retransmission, will the receiver take the first version of the second?
  - At the application layer: how will this corner-case in the spec be interpreted? Will the spec be honored?
- Problem is fundamentally hard
  - Can't reliably alarm on presence of ambiguity due to prevalence of "crud" in real traffic
- Most promising approach: *normalization*
  - Rewrite traffic *inline* to scrub out ambiguities



## Denial-of-Service (DoS)

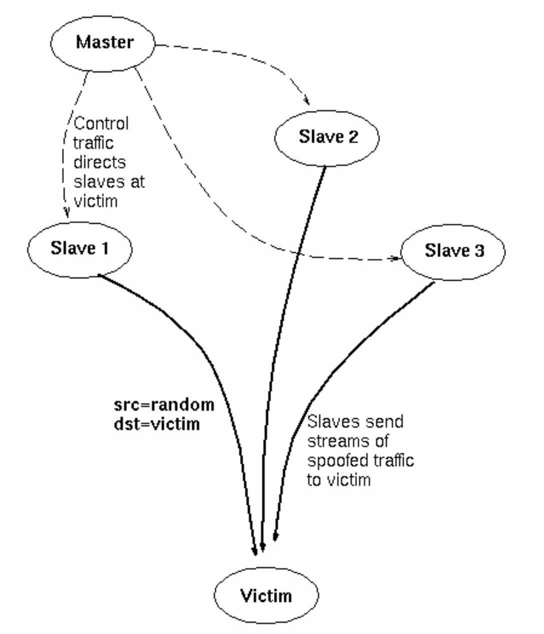
- Attacker's benefit is indirect: inconveniencing the victim, perhaps very seriously
- Some motives:
  - Retaliation (particularly among script kiddie attackers)
  - Commercial advantage
  - Extortion (e.g., threatening to target Internet gambling sites during times of heavy betting; or ecommerce sites during holidays)
  - Abetting an accompanying direct attack (e.g., DoS on an intrusion detection system)

## Denial-of-Service (DoS), con’t

- Can be ***very*** hard to defend against
  - Particularly if attacker can muster a large army of *zombies* (a “botnet”)
- Basic mechanisms
  - “Single packet”: exploit bug that crashes server
  - “Flooding”: overwhelm an element with too much traffic
- Flooding can occur at different layers:
  - Network: prevent traffic from reaching server
  - Server: overtax server to prevent it from having resources to tend to legitimate requests

## DoS: Network Flooding

- Goal is to clog network link(s) leading to victim
  - Either fill the link, or overwhelm their routers
- Attacker sends spoofed traffic to victim as fast as possible
- Using multiple hosts (“slaves”) yields a *Distributed Denial-of-Service* attack, aka **DDOS**
- Traffic is varied (sources, destinations, ports, length) so no simple filter matches it


Master
Control traffic directs slaves at victim
Slave 1
Slave 2
Slave 3
src=random
dst=victim
Slaves send streams of spoofed traffic to victim
Victim

# Network Flooding: Traceback

- How do we find the sources of a spoofed flood?
  - Approach #1: add marks to all packets as they traverse the network, analyze the marks at the victim to find the sources
    - Can work with remarkably small marks (a few bits), leveraging the large volume of marks received at victim
  - Approach #2: routers remember every packet seen
    - Remarkably, this is doable using clever data structures (Bloom filters) and large-but-not-humongous state for a memory of 10s of seconds
  - Approach #3: ISPs monitor their edge routers, find where flood enters their network, phone up the peer sending it to them, repeat

# Network Flooding: Traceback

- Of these approaches, only #3 (ISPs contacting one another) is used in practice
- Why? Because *finding the source isn't useful*
  - What will you do with the information?
    - Getting it turned off requires a lot of leg work; completely impractical if there are **1000s** of sources
    - Finding the source *never* reveals the person behind the attack, because they *launder* their tracks through a series of machines ("stepping stones"), making *attribution* infeasible
- On the other hand, finding ingress into a network <u>is</u> useful, since it facilitates installing upstream filters to ameliorate the flood

# Network Flooding: Defenses

- **Hop-count filtering**
  - Idea: TTL field in IP packets reflects the *hop count* of traffic at a given point.
    - **E.g., if packet sent with TTL=127, then at a point 12 hops along the path (say, near the victim's server) it will = 115.**
  - Remember TTL values seen for different IP addresses
  - When under stress, drop all packets that differ from remembered TTL, or that don't have an entry
- **Pros:**
  - Hard for attacker to guess correct TTL value, so many of their packets discarded
- **Cons:**
  - Requires a lot of *state* to remember previous values
  - Doesn't protect new legit sources, or traffic w/ routing changes
  - Only reduces attacker's flood, doesn't eliminate it

## Network Flooding: Defenses

- **Capabilities**
  - Idea: routers give priority to packets that carry special markings, which the sender can only get from the receiver.
  - If the receiver likes a connection, it sends the markings and that connection receives priority from the network. Otherwise, it doesn't, and traffic continues at low priority.
- **Pros:**
  - Provides "opt in" protection for good guys rather than "opt out" filtering of bad guys.
- **Cons:**
  - Bootstrap problem of how to get capability in the first place (since initial request lacks priority)
  - Requires significant infrastructure

## Network Flooding: Defenses

- **Other defenses**
  - Overlays: spread ***access*** to the service across many locations, so flooding them all takes much greater resources
  - Distribution: spread the service itself across many locations
  - Pushback: routers that detect a stressed link signal next upstream hop to rate limit; messages recursively propagate towards sources
  - ***Overprovisioning*****:** fat enough pipes that it's hard for an attacker to fill them.
    - This turns the problem into **service flooding**

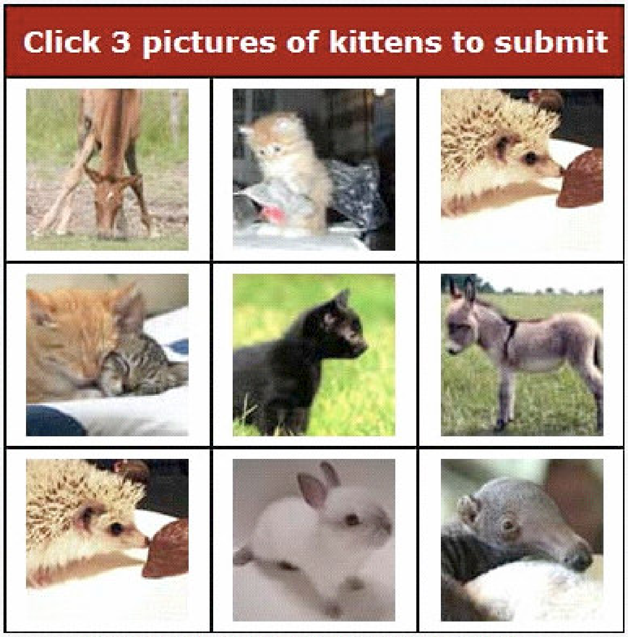
## Service Flooding: Defenses

- **SYN cookies:** server avoids creating any *state* for a new connection until client completes SYN handshake
  - Done by encoding all initial state in the SYN ACK sequence number that the client must echo
  - Flooders that spoof source addresses never receive SYN ACK so can't complete handshake and consume server resources
- **Puzzles:** server requires client to prove it has conducted some significant computation before accepting request
  - Idea: level the playing field between legit clients and flooders by limiting each to a slow rate of requests

## Service Flooding: Defenses

- **CAPTCHAs**
  - Idea: "Reverse Turing Test"
    - Prove that a client is a human rather than a machine
  - Based on known-hard AI problems that humans solve readily

  - Drawbacks:
    - If visual, discriminates against blind users
    - Depending on the problem, an arms race …



Click 3 pictures of kittens to submit

The KittenAuth system. Source: ThePCSpy.com.

## Network Security Summary

- Very wide range of problems/issues spanning essentially every facet of networking
- Key considerations:
  - Dealing with an **adversary** / arms race
    - This is getting much more serious now that attackers can **make money**
  - Security has a major **policy** component
    - Vital to consider **threat model**, trading off resources
  - Architecture lacks support for fundamental notions such as authentication, authorization, accountability
    - Security bolted on post facto lacks coherence

# Authentication Technology

- IP ⇒ IPSEC
  - Layer between IP header and transport header
  - Can provide authentication, confidentiality, integrity
  - Management of crypto keys a big headache
  - Also difficult to secure communication between parties who don't know each other (e.g., public Internet services)
- DNS ⇒ DNSSEC
  - Adds signing of DNS data to authenticate who provided it
  - Includes mechanisms for key distribution
  - Management remains a significant headache
  - Pesky problem of DNS replies now can exceed 512-byte limit

# Authentication Technology, con't

- Telnet, Rlogin ⇒ SSH
  - Connection negotiates crypto session key based on public key exchange, encrypts all subsequent traffic
  - Pointwise deployment model that involves parties who know one another has led to success
- HTTP ⇒ SSL, TLS (standardized version)
  - Model similar to SSH
  - Usually, only the server authenticates to the client, not vice versa
  - This plus browsers containing *built-in public keys* from which they can establish third-party trust in server's key leads to deployment success
- Routing: BGP ⇒ SBGP? Work in progress …