Security

CS 241

May 5, 2014

University of Illinois

Adapted in part from slides by Scott Shenker, Jen Rexford, Ion Stoica, Vern Paxson

“Security” is a very broad topic…

“Security” describes

- Hardware
- Software
- Data
- People
- Policies
- Procedures
- Governance

…even the best software algorithm has several points of failure!

Security goals (an incomplete list)

Availability

- Can I rely on the service being available when I need it?
- Infrastructure compromise, DDoS

Authentication

- Who is this person/machine?
- Spoofing, phishing

Integrity

- Is data preserved in original form?

Confidentiality

- Can adversary read the data?
- Sniffing, man-in-the-middle

Provenance

- Who is responsible for this data?
- Forging responses, denying responsibility
- Not who sent the data, but who created it

Case Study: AACS encryption

AACS: “Advanced Access Content System”

- Copyright protection on HD DVD media

What happened?

Case Study #1: AACS encryption

AACS: "Advanced Access Content System"

- Copyright protection on HD DVD media

What happened?

- PowerDVD and AnyDVD software stored the "master" decryption key in RAM
  - Analysis: "nothing was hacked, cracked, or reverse engineered", "no debugger was used", "no binaries changed"
- **`09F911029D74E35BD84156C5635688C0`**

Cryptographic Hash Function

Any general **hash function**:

- Takes in data and produces a numeric result
- Java: `Object.hashCode()`
  - Used for hash tables, fast string comparisons, etc.

Cryptographic Hash Function

A **cryptographic hash function** should be:

- Easy:
  -
- Hard / Impossible:
  -
  -

SHA-2/256 Examples

(empty string)

- **e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855**

The quick brown fox jumps over the lazy dog

- **d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592**

The quick brown fox jumps over the lazy dog.

- **ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c**

The quick brown fox jumps ovar the lazy dog.

- **02e4625126139fbd3f91e44749fa51a9f7aeabeb63301cb251be1904b7c668c0**

Storing Passwords

How does Facebook store a password?

What's wrong?

"password"

➔ (SHA-256) ➔

`5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8`

Storing Passwords

How does Facebook store a password?

"9rjef98wty4h password"
➔ (SHA-256) ➔

`4318fd81e7c56701df71b49247d560e797306ea355002baa5f39b16a904b8fe6`

Password Salt

A salt is a (usually random) string added to the input before a hash function is applied.

- A different salt must be used for every input.

Why use a salt?

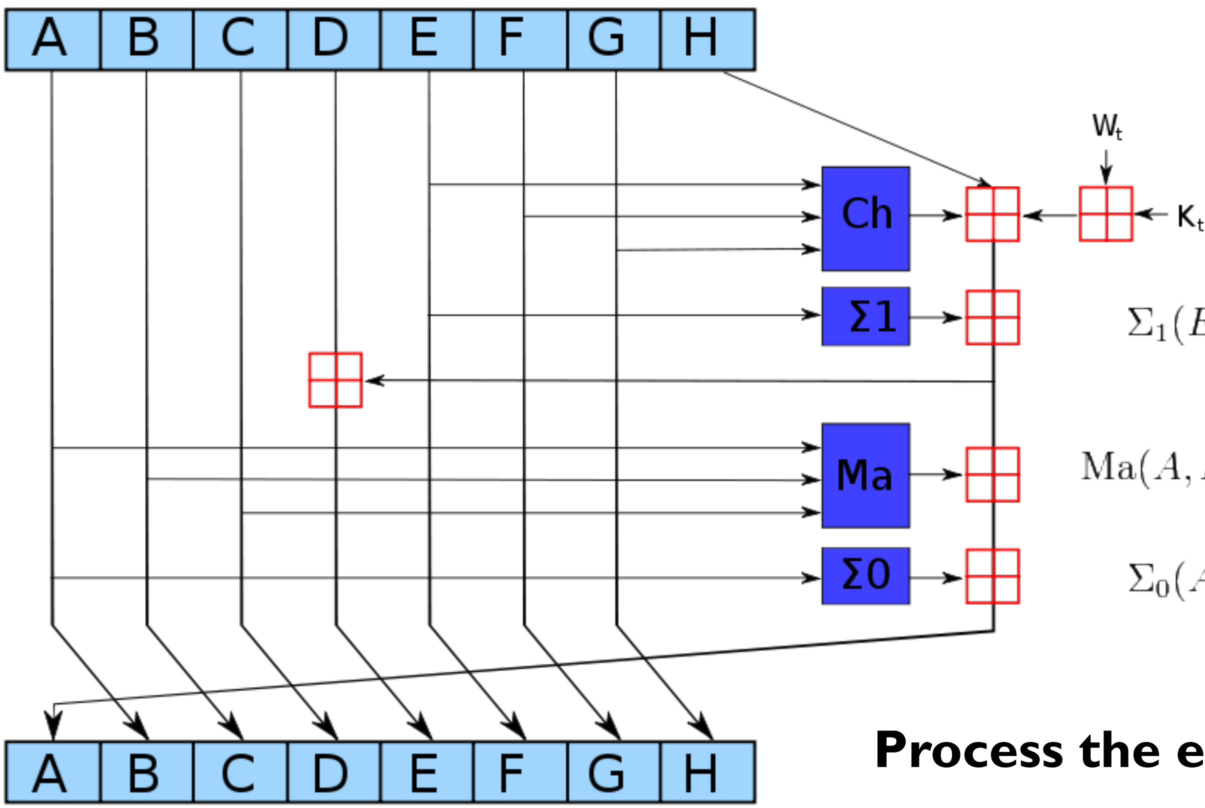
If attacker obtains password hashes and salts,

- Cannot use a known dictionary to crack an individual password
- Need separate attempts to crack each user
- Makes cracking passwords more difficult, not impossible

SHA2

SHA2 is a **public** algorithm

- ***Security in the mathematics, not in keeping the implementation a secret***



$$\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$

$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

$$\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$

$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$

Process the entire message, <u>64</u> times.

SHA2

Right now, SHA2 is considered a secure hash.

- ***Mathematics have not been broken***
- ***The complexity of reversing a hash would take more computing power than has ever been created***

- SHA2 has several variants based on the length of the output desired: SHA-256 (256-bit output) is most common.

Other Algorithms

MD5 (1991):

- 2005-2008: MD5 was mathematically simplified and available processing power could fake hashes
- *“should be considered cryptographically broken and unsuitable for further use”*

SHA-0 (1993):

- 1998: Was shown to be easily simplified; some hashes can be reversed in less than an hour!

SHA-1 (1995):

- Replacement to concerns about SHA-0
- 2005: Theoretical attack developed showing some weakness in the mathematics (reverse in <= 2^{69})

Cryptographic toolkit for security

Cryptographic hashes

Symmetric key cryptography

Asymmetric (public) key cryptography

Digital signatures

Public-key infrastructure (PKI)

Yet still...

Most Significant Operational Threats Experienced

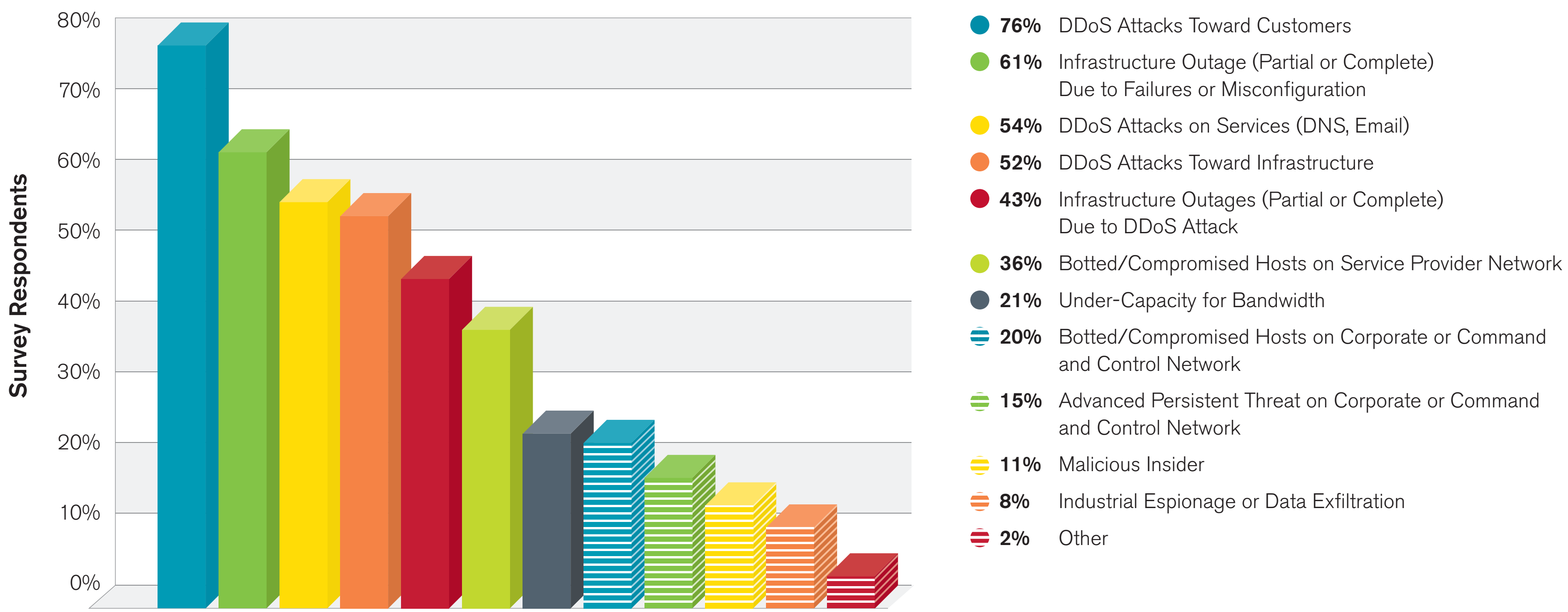


Figure 10 *Source: Arbor Networks, Inc.*

Case study 2: Denial of Service (DoS)

Attacker prevents legitimate users from using something (network, server)

Motives?

- Retaliation
- Extortion (e.g., betting sites just before big matches)
- Commercial advantage (disable your competitor)
- Cripple defenses (e.g., firewall) to enable broader attack

Often done via some form of flooding

Can be done to different systems

- Network: clog a link or router with a huge rate of packets
- Transport: overwhelm victim's ability to handle connections
- Application: overwhelm victim's ability to handle requests

Denial of Service (DoS)

Average Number of DDoS Attacks per Month

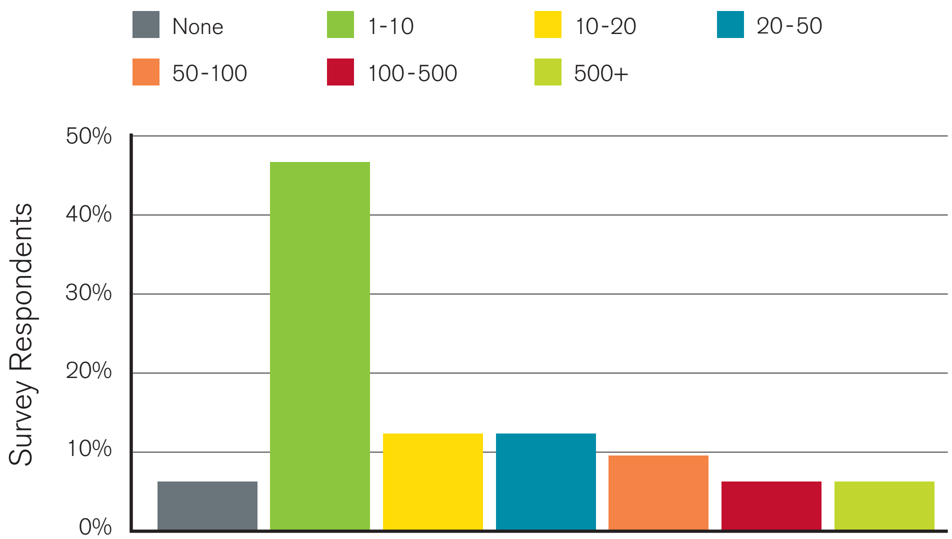


Figure 15
Source: Arbor Networks, Inc.

Layer 7 DDoS Attacks

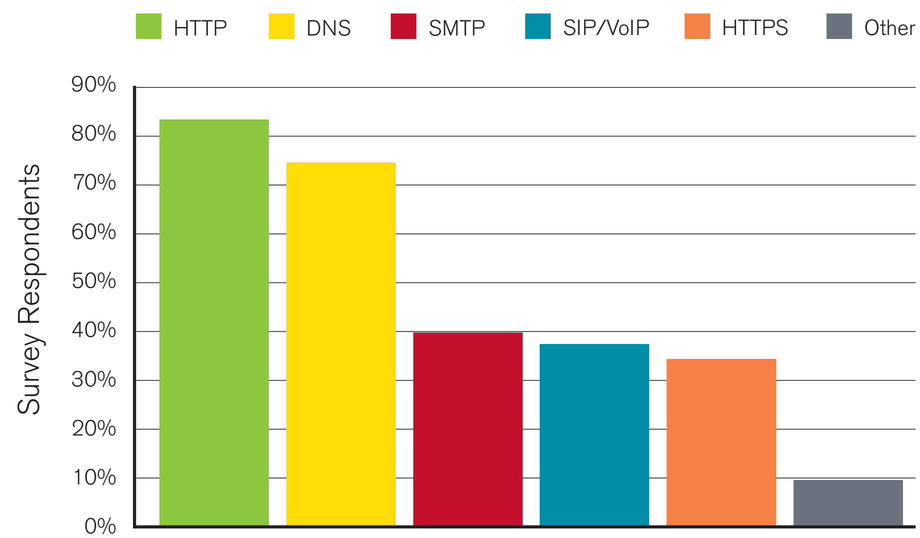


Figure 8
Source: Arbor Networks, Inc.

DoS: Network Flooding

Goal is to clog network link(s) leading to victim

- Either fill the link, or overwhelm their routers
- Users can't access victim server due to congestion

Attacker sends traffic to victim as fast as possible

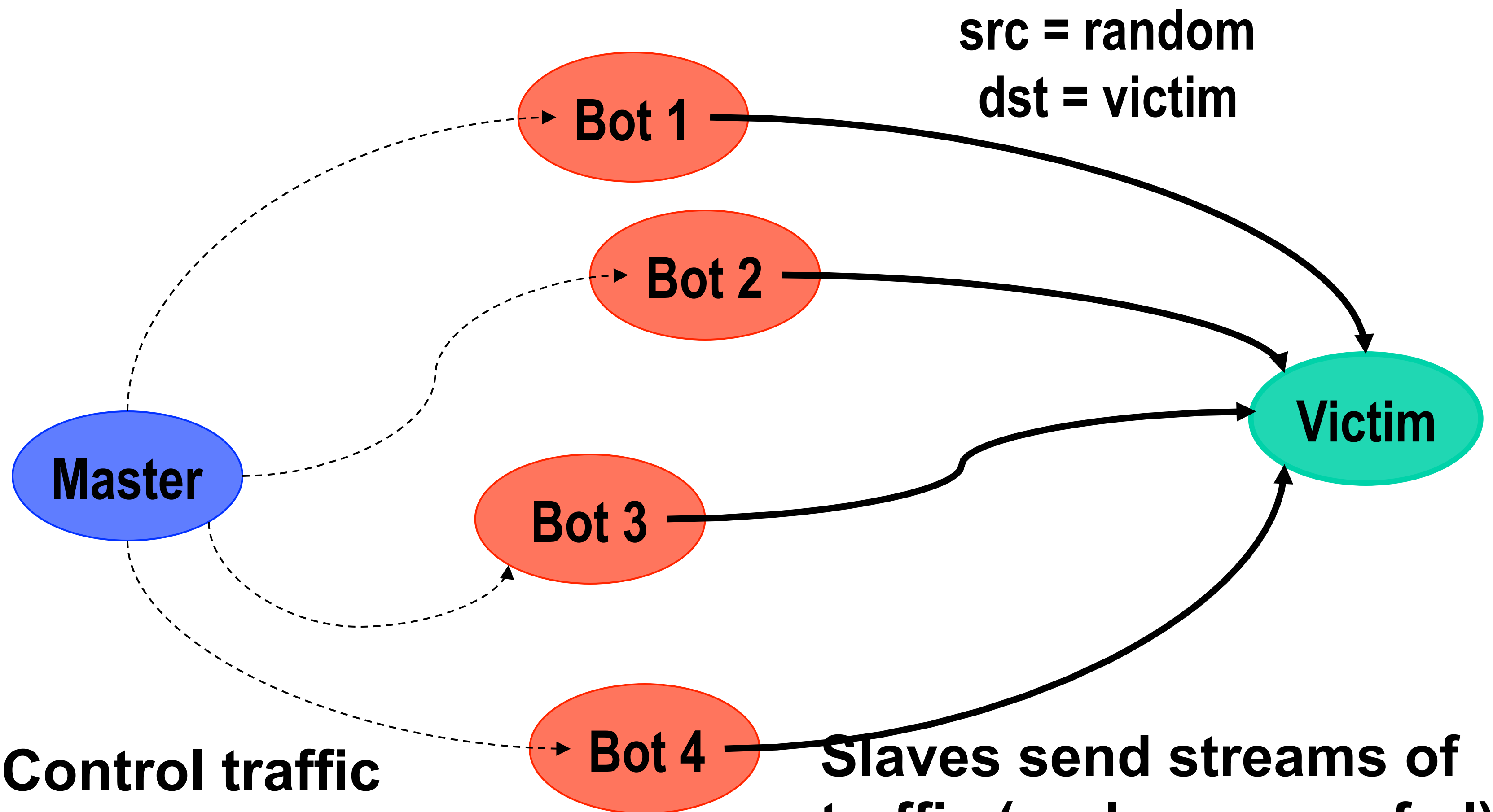
- It will often use (many) spoofed source addresses

Using multiple hosts (slaves, or zombies) yields a Distributed Denial-of-Service attack, aka DDoS

Traffic can be varied (sources, destinations, ports, length) so no simple filter matches it

If attacker has enough slaves, often doesn't need to spoof - victim can't shut them down anyway! :-(

Distributed Denial-of-Service (DDoS)



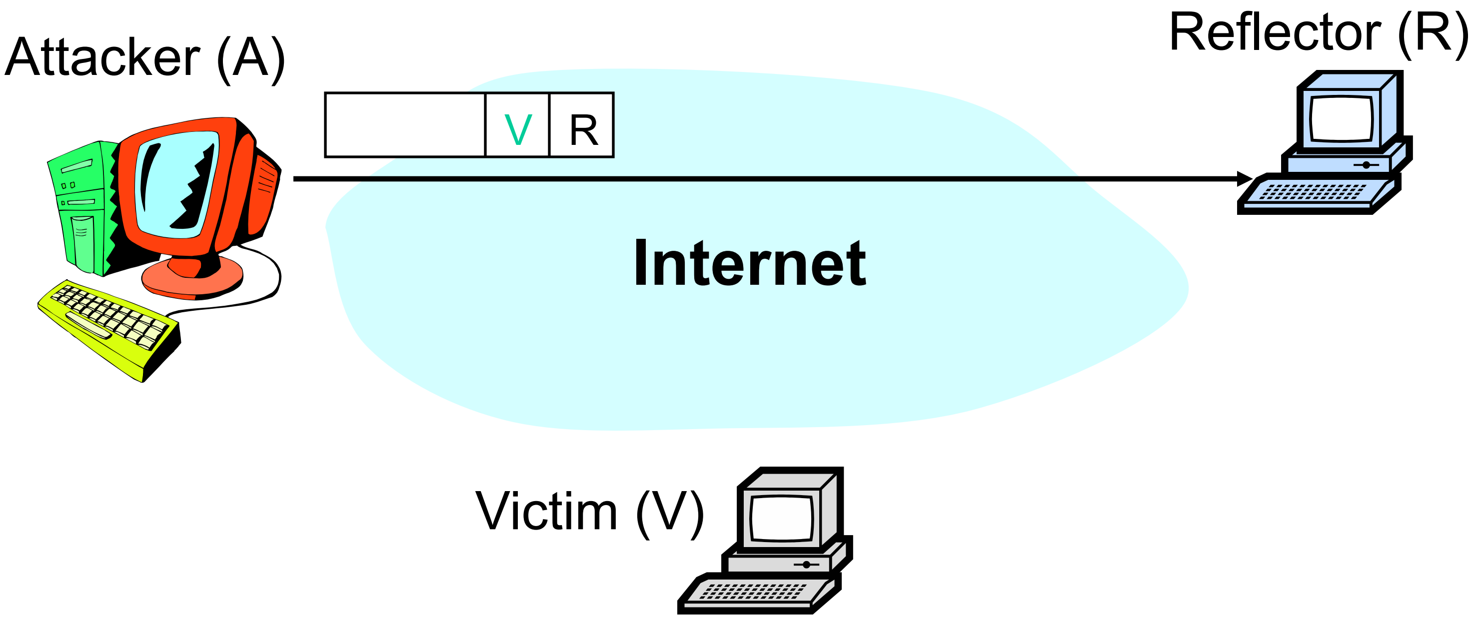
Control traffic directs slaves at victim

Slaves send streams of traffic (perhaps spoofed) to victim

Very Nasty DoS Attack: Reflectors

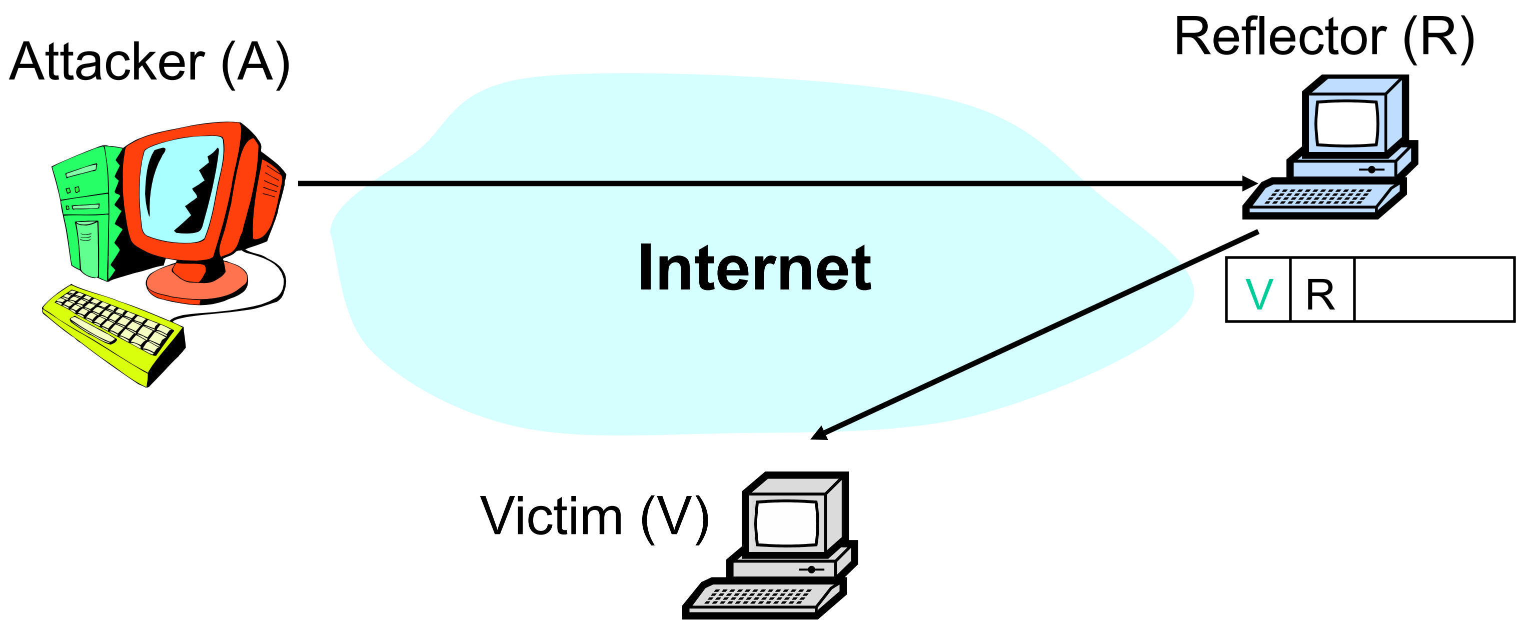
22 *Reflection*

- Cause one *non-compromised* host to help flood another
- E.g., host A sends DNS request or TCP SYN with source V to server R.

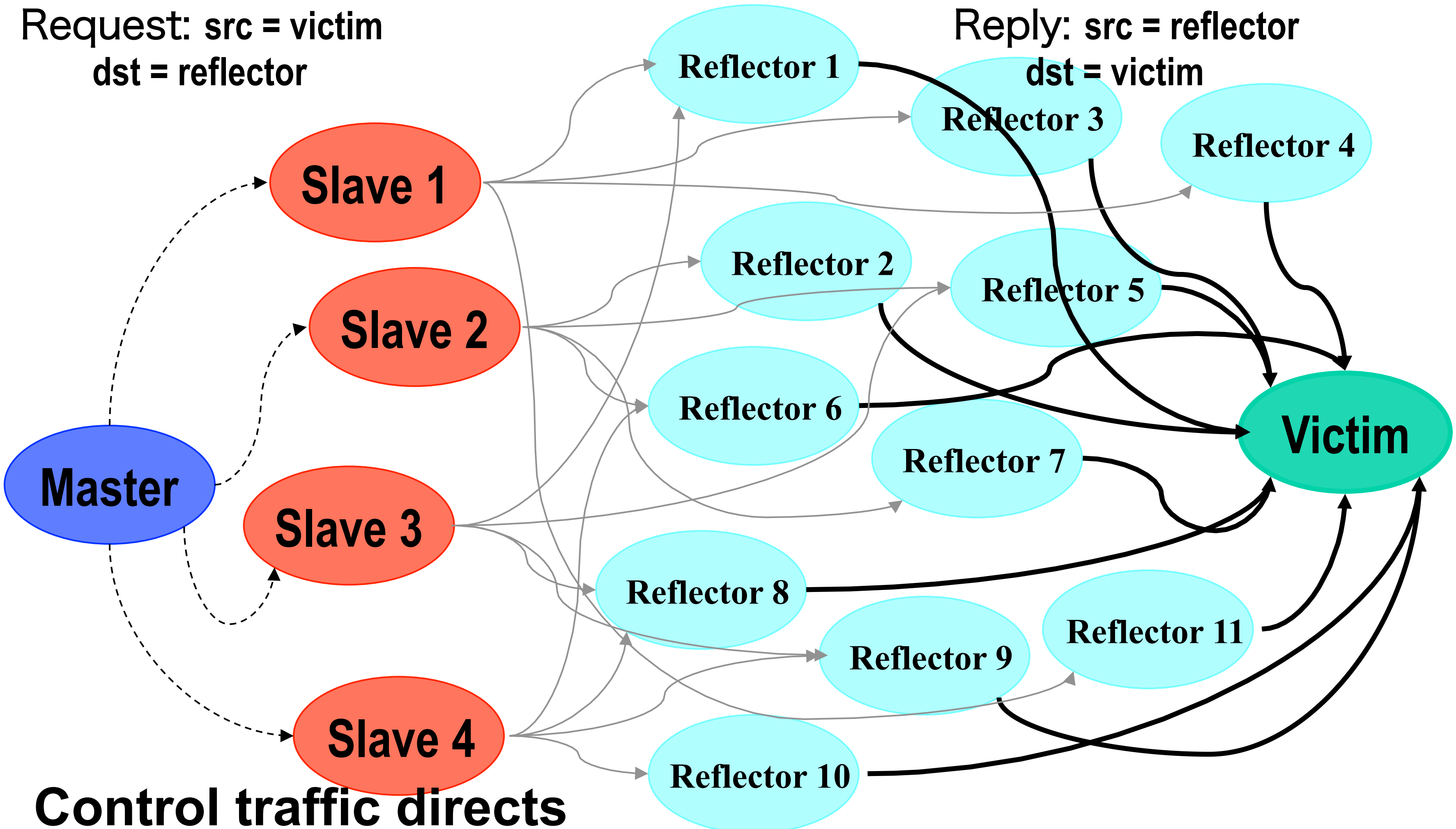


Very Nasty DoS Attack: Reflectors

23 • ***Reflection***
  – **Cause one *non-compromised* host to help flood another**
  – **E.g., host A sends DNS request or TCP SYN with source V to server R.**



Diffuse DDoS: Reflector Attack



Control traffic directs slaves at victim & reflectors

Reflectors send streams of non-spoofed but unsolicited traffic to victim

24

Lessons for building systems

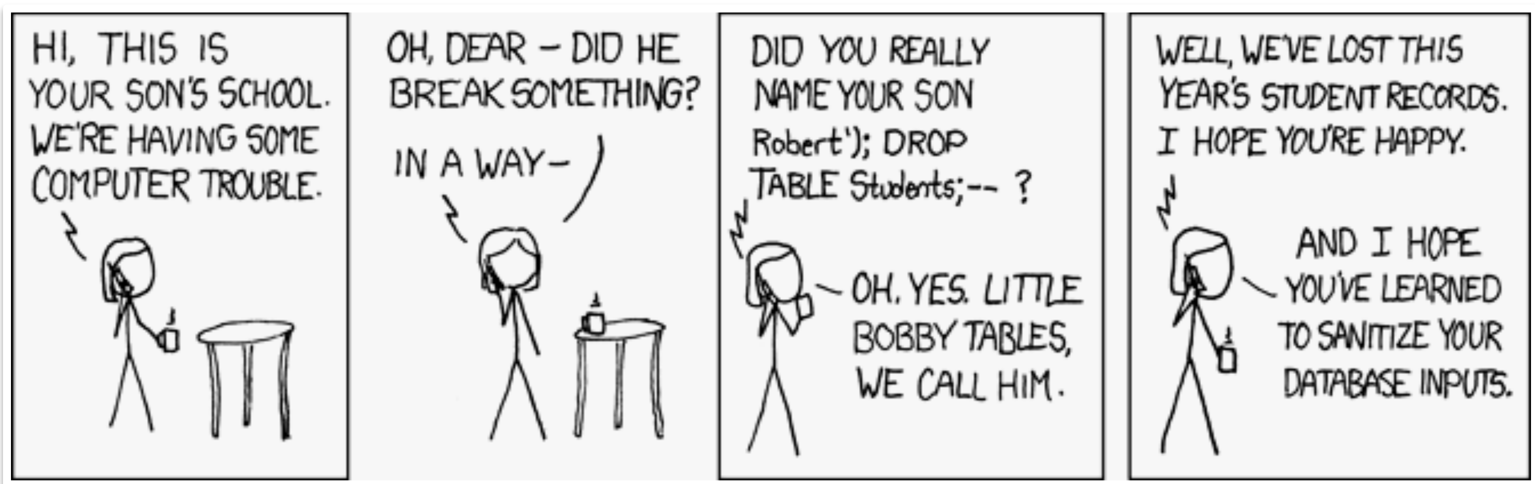
Need to think like an attacker

- Think: If I had the power to do X, can I cause bad event Y?

Defensive programming

- If a user or code module gives you arbitrarily weird input, could it crash or exhibit other undesirable behavior?
- Answering "no" requires well-defined interfaces, good modularization

Think: how could someone crash your web server?


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.