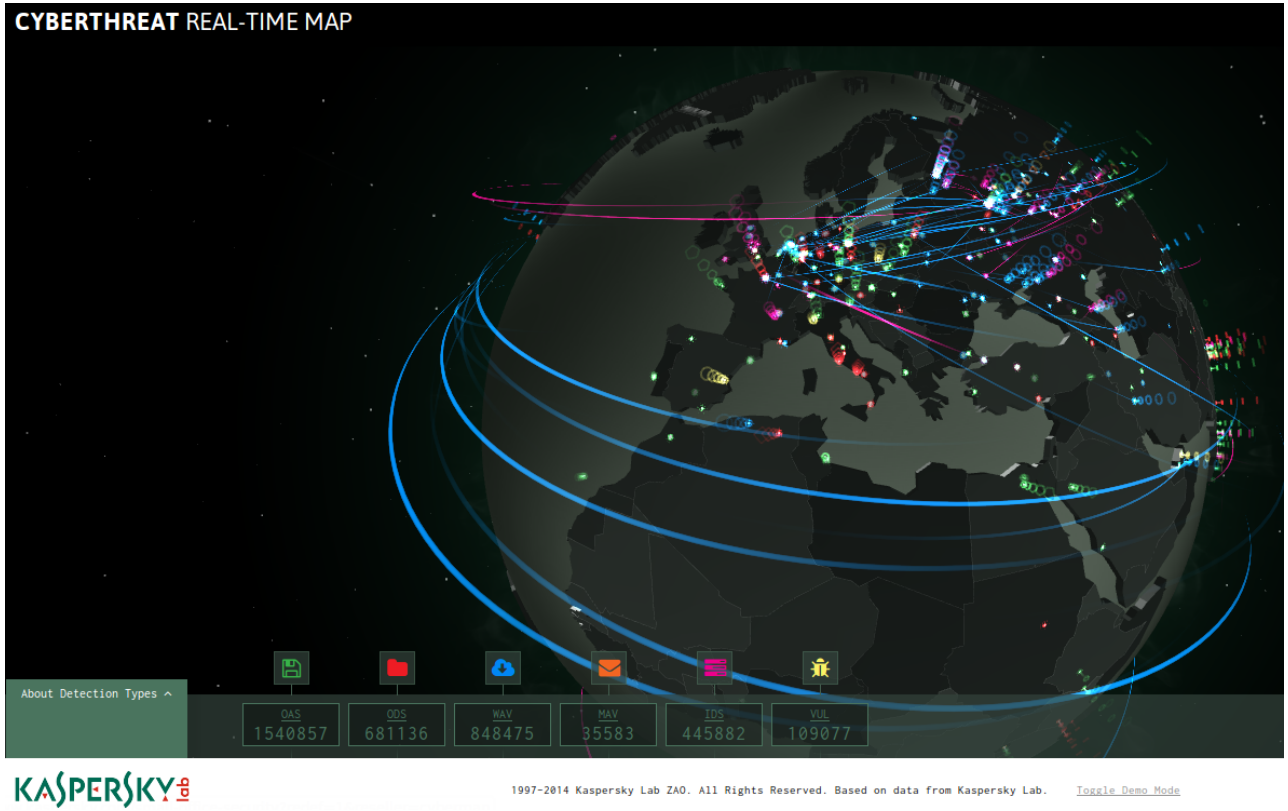
# 50.020 Security
# Lecture 6: Web Security I

# What do we mean by web security?






- Security of servers and clients on the public internet
  - anyone with public IP is exposed to the world
  - Users cannot directly be held responsible
  - Attacks on availability, confidentiality, banking, scams, ...

# What attacks are possible on the Internet



- Attacks on communication
  - Strong attackers can mount Man-In-The-Middle attacks
    - ISPs, Governments can eavesdrop or manipulate traffic
  - Scriptkiddies, Criminal Hackers are not able to eavesdrop or manipulate victim's traffic
- Attacks on end hosts and servers
  - Common Attacks: DoS (availability)
  - Active attacks (sending you malicious traffic)
  - Passive attacks (tricking you to contact them)
- Attacks on traffic can be prevented by TLS (discussed later)
- This week, we focus on attacks on hosts and servers

# Main attack vectors



- Server side:
  - Weak access control
  - Processing of user-provided input
  - Denial-of-service attacks
- User PC side:
  - Execution of downloaded content
    - Impersonation of trusted sources
  - Accidental exposure of interfaces
- User accounts:
  - Private data
  - Impersonation

# Open Web Application Security Project



OWASP Top 10 (The Ten Most Critical Web Application Security Risks) in 2017

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monotoring

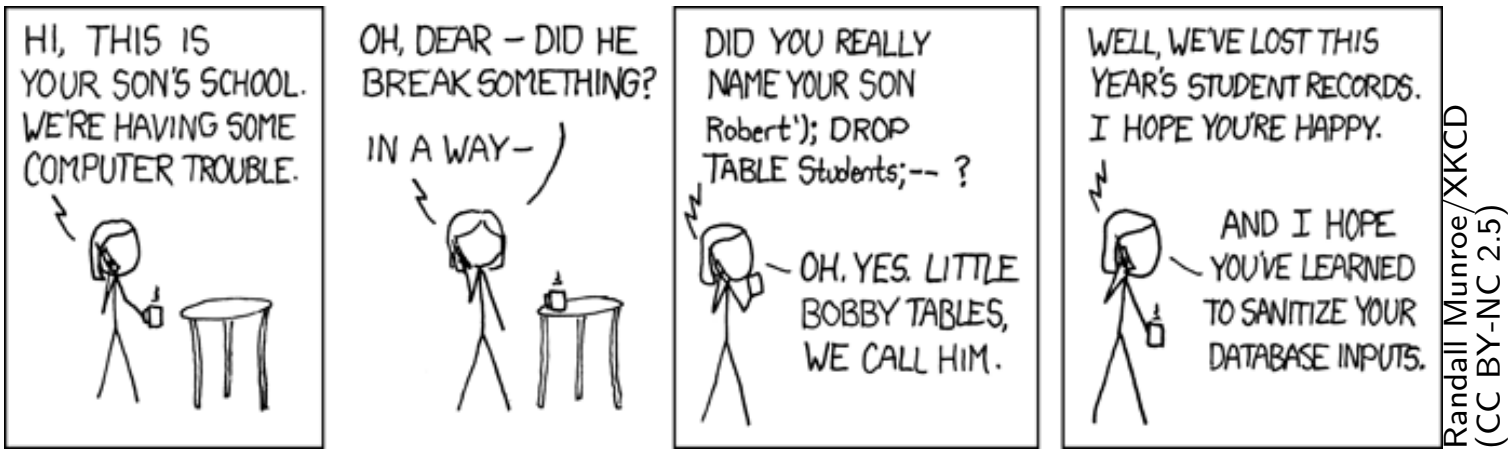
# Server security: Injection (user provided input)



- Processing input from untrusted sources is dangerous
  - Buffer overflow (strings, images, . . . ) (upcoming lectures)
  - SQL injection
- But this is the server's main job!
- Even harder: presenting user content to users
  - Cross-site scripting (XSS)
  - Language filtering, image filtering, copyright, . . .
- Example attacks using user provided input
  - Buffer overflows
  - SQL injection
  - Cross-site scripting (XSS)

# SQL Injection






- SQL is a *query language* for databases, based on ASCII strings
- SQL injection attacks rely on incorrect validation of input data
- If user input is directly inserted in interpreted code (SQL)
  - Attackers can try to change the code
  - Could allow attacker to do anything with database

# SQL injection example



- Given a SQL query with user-provided string *userName*

```
SELECT * FROM Students WHERE name = '$userName';
```

- With normal input, e.g. *userName="Robert"*, the query is

```
SELECT * FROM Students WHERE name ='Robert';
```

- With *userName = "Robert'; DROP TABLE Students;- -"*

the query is

```
SELECT * FROM Students WHERE name ='Robert';
DROP TABLE Students;--';
```

- What will be the result?

# SQL injection example (Continue)



- As result of the previous attack, all student entries would be deleted
- Could also be used to read out content from other tables
  - Especially if all results of SQL query are returned to the user
  - Attacker will have to learn about table names etc first
- We will look at that in the lab.

# SQL injection countermeasures



- Restriction (+validation) of character set for values
  - E.g. only a-z, A-Z, 0-9
- Proper escaping of special characters
  - E.g. turn ' into '' and so forth, troublesome
- Semantic analysis of query for execution
- Restrictive configuration of database
- Disallow dropping or selection on sensitive data
- Use of prepared statements (with parameterized queries).
  On example:
  SELECT * FROM Students WHERE name = ?;
  - More parameterized query examples:
    `https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Query_Parameterization_Cheat_Sheet.md`

# Server security: XSS



- In Cross-Site Scripting (XSS) attacks, user content enables attacks on other users
- Great introduction to the topic
  - "Privacy or Transparency? / Samy Kamkar at Mindshare LA"
  - `https://youtu.be/dm4KvmOfK_8`

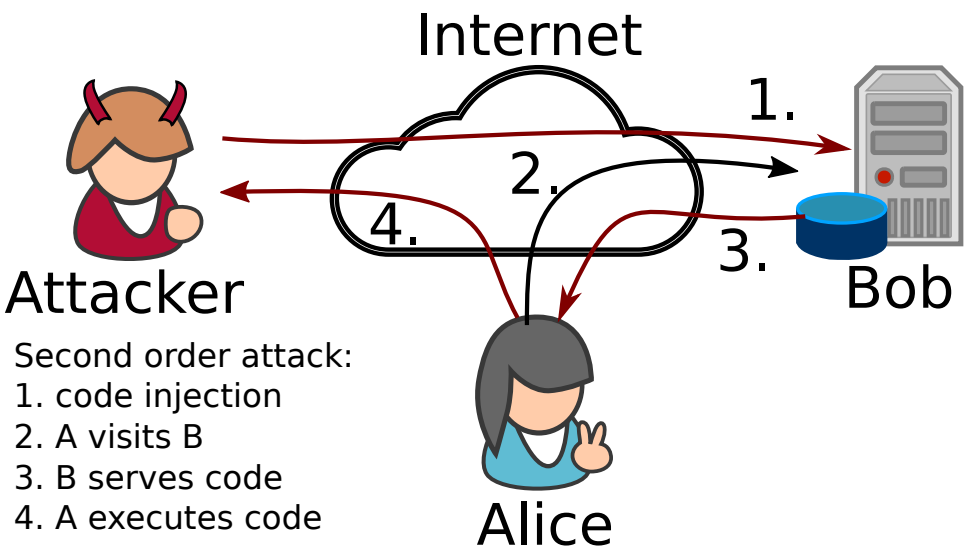
# Cross-Site Scripting (XSS)



- In XSS attacks, a server sends out the script of an attacker
- Target is the user, not web server. Code executed by browser.
- Dangerous, as the user *trusts* content from known websites
  - Browsers also use *origin-policies* to restrict access to one site
  - These policies can be bypassed with XSS attacks
- Enabled by improper validation/escaping of user data
- XSS types:
  - Persistent/ second order XSS attacks
  - Reflected/ First order XSS attacks

# Persistent/ second order XSS attacks



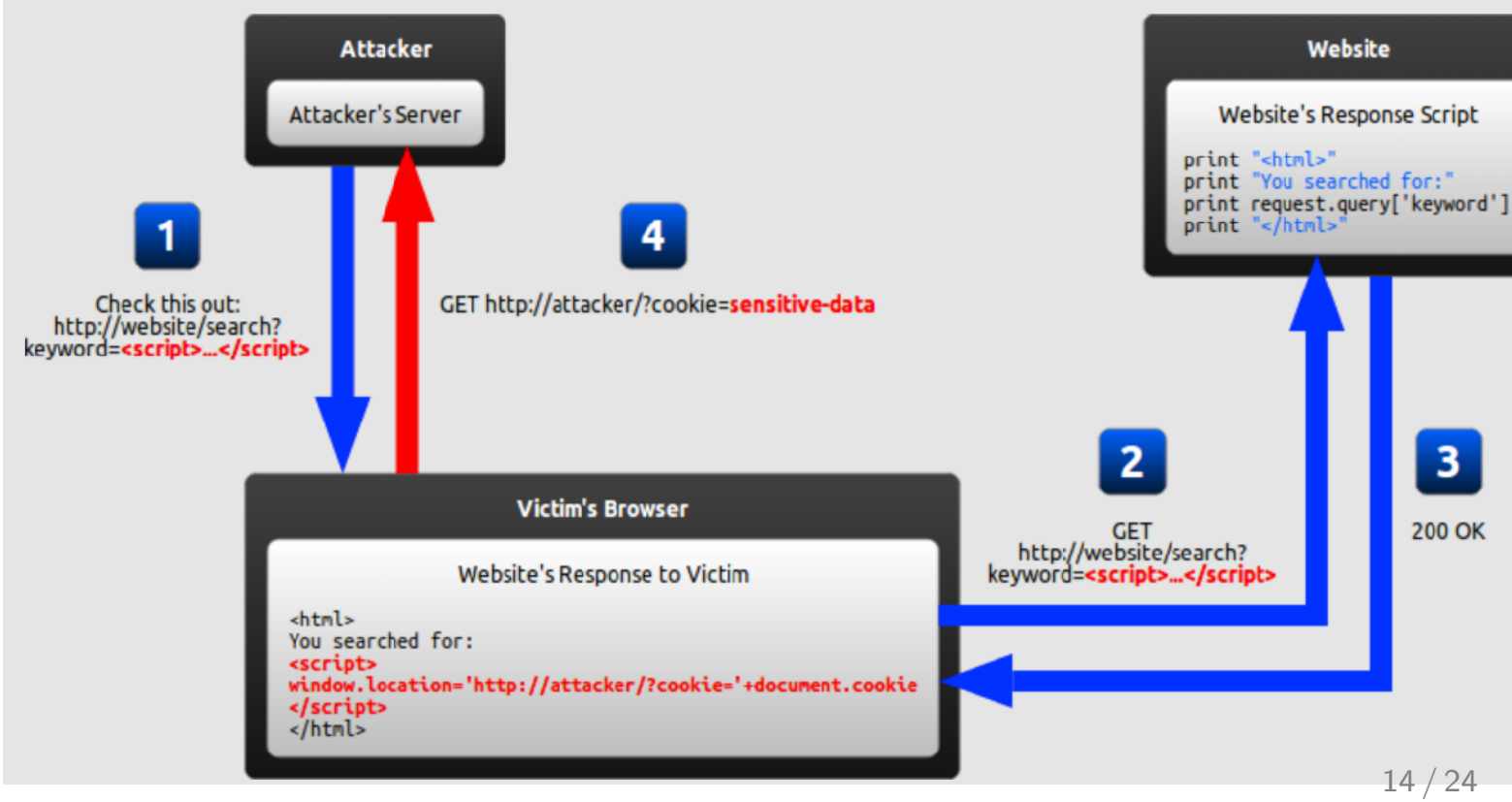



- Code will be **stored** by the server, and sent out from now on.
- Attack delivery method: Upload attack, users who view it are exploited
- Example: "Samy" Myspace worm

# Reflected/ First order XSS attacks

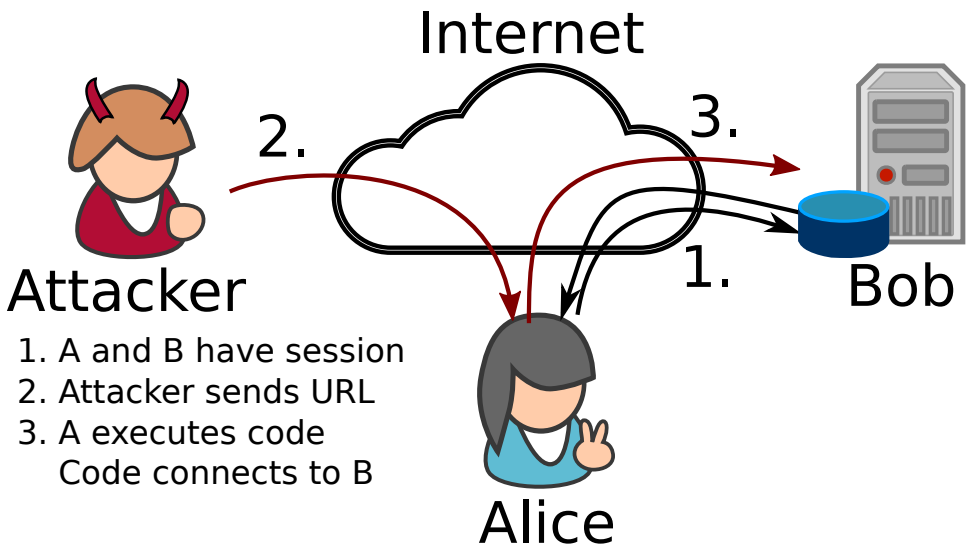


- Non-persistent code injection. Server **reflects** back the injected code.
- Attack delivery method: Send victims a link containing XSS attack
- One example:



# Cross-site request forgery (CSRF)






- Similar to XSS, CSRF attacks are injecting code (e.g. via URL)
- **XSS abuses users' trust in server, CSRF abuses server's trust in user**
- The code in CSRF is the executed by the victim's browser
  - Connects to third party server (e.g. facebook)
  - Uses existing autenticated session with that server

# Weak access control



- Servers and services need to be managed remotely
- We already discussed password guessing attacks
- Number of attempts and block duration decided by policy
- Automated scripts can be prevented with captchas
- Usability vs security trade-off
- How expensive are attacks for the attacker?

# Attack effort on the internet



- One problem of the internet: communication is cheap
  - Attacker can use lots of traffic for brute force attack
- This is exploited in email spam (conversion rate: once per 12 million)[1]
- This is also exploited for forum spam and brute force atacks
- How to make communication just a little bit more expensive?
  - Find something which is easy for user, expensive for computer
    - Captchas
    - Client-side puzzle solving (e.g. Proof-of-work, Bitcoin!)

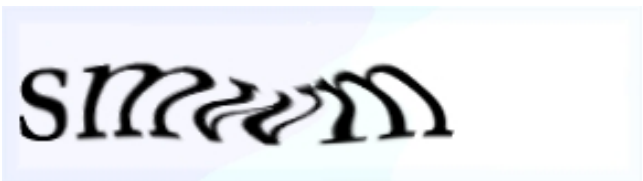
[1] http://www.icir.org/christian/spamalytics/

# Captchas



"Completely Automated Public Turing test to tell Computers and Humans Apart"

- Automatically generated images with known content
- Humans are usually better in character recognition
- Automated OCR is made harder by distorting the image
- Problems with captchas
  - Accessibility for impaired people
  - Automatic distortion can make images unreadable



"smwm" text as captcha

# How to break captchas



- Assumption: attacker is presented with captcha
  - needs easy and cheap way to solve fast
- Any suggestions?

# How to break captchas



- Assumption: attacker is presented with captcha
  - needs easy and cheap way to solve fast
- Any suggestions?

---

- automatic recognition (OCR)
- "Employ a village" in cheap labour countries
- C&C outsourcing to victims

`http://www.inwyrd.com/blog/2010/03/hijacking-koobfaces-captcha-solver/`

# Improving captachs



- Captchas have several disadvantages
  - Usability can be low
  - Prevents access for visually impaired people
- Any suggestions for better captchas?
- Should be easy to use, hard to attack

# Improving captachs



- Captchas have several disadvantages
  - Usability can be low
  - Prevents access for visually impaired people
- Any suggestions for better captchas?
- Should be easy to use, hard to attack

- Images
- 3D perception (order objects by size)
- Knowledge associations
- your topic here!

# Information leakage



Which one is the more secure way for error messages:

- "The second character of your password is wrong"
- "Your password is wrong"
- "Your username or password is wrong"
- "No matching entry in database"

# Information leakage



Which one is the more secure way for error messages:

- "The second character of your password is wrong"
- "Your password is wrong"
- "Your username or password is wrong"
- "No matching entry in database"

- $->$ Make sure that you do not disclose too much information
- Timing sidechannels (we will discuss later)
- Don't disclose data otherwise, e.g. in account creation

# CVE



The Common Vulnerabilities and Exposures (CVE) system is a US database for vulnerabilities

- Provides unique identifier for vulnerabilities
- Gathers information on vulnerability and affected systems
- Threat levels can be described by CVSS scores

`http://www.cvedetails.com/`

# How to defend servers against attacks?



- Minimize attack surface/ exposed interfaces
- Ensure that security patches are applied asap
- Restrict number of users, enforce strong authentication
- Ensure that all control of server is using secure connections
  - SSH instead of telnet
  - HTTPs for web-based configuration
  - Secure tunnels for remote desktop control (e.g. TLS)
- Properly parse and escape untrusted input

# Conclusion



- Web security is both about servers on the internet, and users
- Servers are mostly threatened by
  - Processing of user-provided input
  - Account compromise
  - (Distributed) Denial-of-Service attacks
- User PCs are threatened by
  - Execution/processing of downloaded content
    - Untrusted content
    - Trusted, but compromised, content
  - Accidental exposure of interfaces
- User accounts contain valuable information
  - User info is *traded* on the internet!