CS1320
Creating Modern Web and Mobile Applications

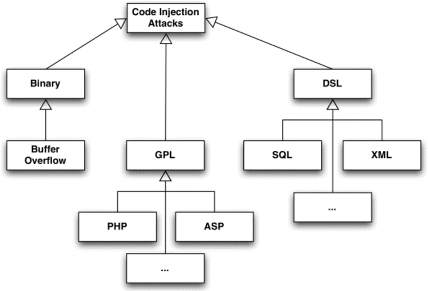
Lecture 29

Security II

Review

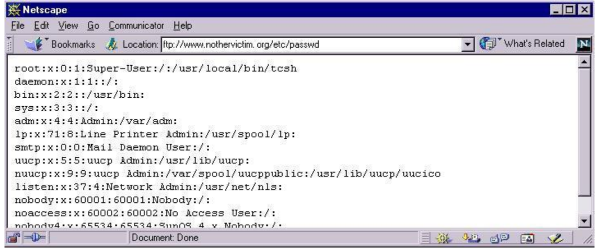
- Security is a major concern
- Lots of obvious problems
- Lots of non-obvious problems
  - SQL injection attacks are the most prevalent
- And there are others …
  - These get more obscure, complex, difficult to address

Code Insertion Attacks



- SQL queries aren't the only place where web apps run arbitrary user-providable code
  - Php, JavaScript, Python have eval(…) statements
  - Back end might run system commands (ls on a directory)
- These have the same vulnerabilities
  - Solutions are similar (but no prepared statements)
  - These should be avoided if at all possible
- DO NOT USE eval OR RUN SYSTEM COMMANDS

File Naming Attacks

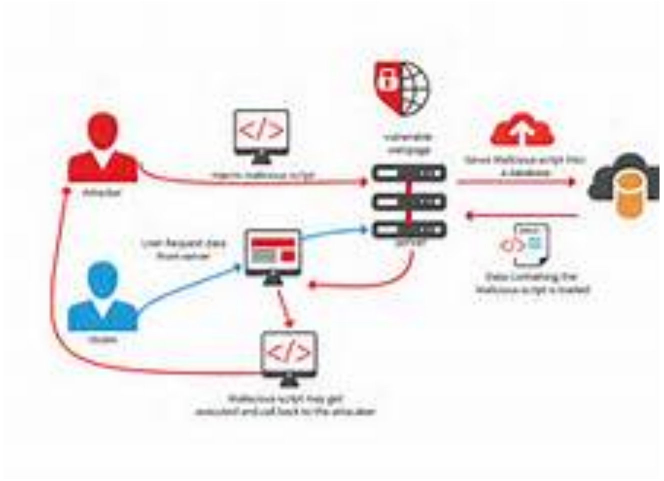

Netscape
File Edit View Go Communicator Help
Bookmarks
Location: ftp://www.nothervictim. org/etc/passwd
What's Related
Document: Done

```
root:x:0:1:Super-User:/:/usr/local/bin/tcsh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
smtp:x:0:0:Mail Daemon User:/:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:37:4:Network Admin:/usr/net/nls:
nobody:x:60001:60001:Nobody:/:
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS 4.x Nobody:/:
```

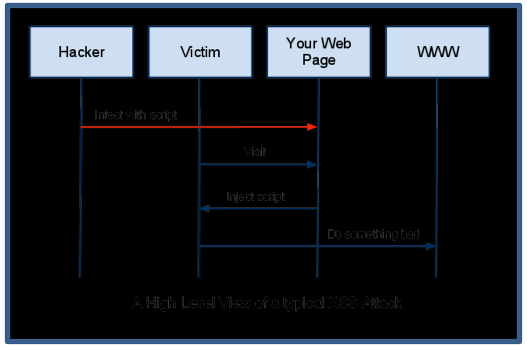
- Suppose your back end opens a particular file
  - Based on the user name
    - Image for user is /web/html/site/user_images/${user}
  - What happens if I use the user name "../../../../etc/passwd"
  - Or "user/../../../../../etc/password"
- Solutions
  - Validate the form of the name
  - Don't use names directly (look up image name in database)
  - Restrict access to the file system
    - chroot provides a virtual root (node.js accessible)
    - php/java/tomcat/ruby security policies

Cross Site Scripting Attacks

- The attacker inserts arbitrary HTML on your web page
  - How can this ever happen?
    - Reviews, feedback, wikis, …
  - XSS
- What can go wrong
  - Disrupt the page or the portion where inserted


Attacker
Injects malicious script
Vulnerable webpage
Server
Send Malicious script into a database
User Request data from server
Data Containing the Malicious script is loaded
Victim
Malicious script has got executed and call back to the attacker

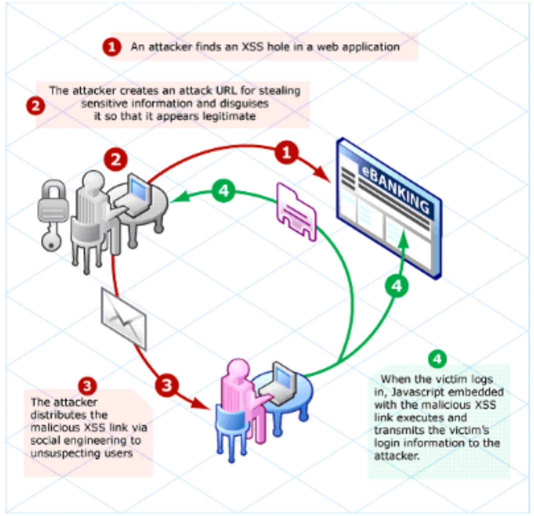
Cross Site Scripting



A High Level View of a typical XSS Attack

- What if the HTML include <script> tags?
  - Replace the page with a new one
    - Fake instance of a page to get passwords, accounts, etc.
  - Pass information from the page to foreign page
    - Cookies, passwords, credit card numbers, session ids
  - Download user's cookies (passwords) for other sites
- Inside a script, the code can do almost anything
  - Effectively take over the browser, spy on the user, …

Cross Site Scripting: How


1 An attacker finds an XSS hole in a web application
2 The attacker creates an attack URL for stealing sensitive information and disguises it so that it appears legitimate
eBANKING
3 The attacker distributes the malicious XSS link via social engineering to unsuspecting users
4 When the victim logs in, Javascript embedded with the malicious XSS link executes and transmits the victim's login information to the attacker.

- Suppose you allow user comments
  - Guest book, ratings, wiki, postings, …
  - Text from user is inserted into HTML
- Suppose instead of typing "I love this page"

```
"I love this page<script
  language='javascript'>document.location='http://bad/';
        </script>"
```

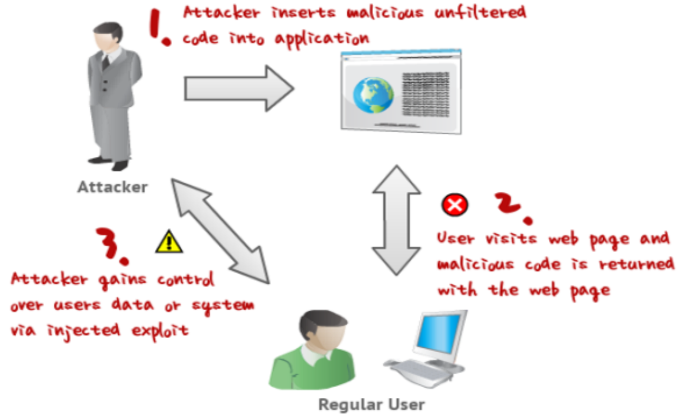
  - What would happen?

Cross Site Scripting Example

- Go to http://bdognom-v2.cs.brown.edu:5002
  - Cross Site Scripting Attack Lesson
  - Provide name to be used in next page
    - Goal is to have it display the given alternative page
  - Try this for 5-10 minutes
  - Raise (virtual) hand or thumbs up when done
  - Recall
    - alert(“message”)
    - doument.location=‘http://bad/’

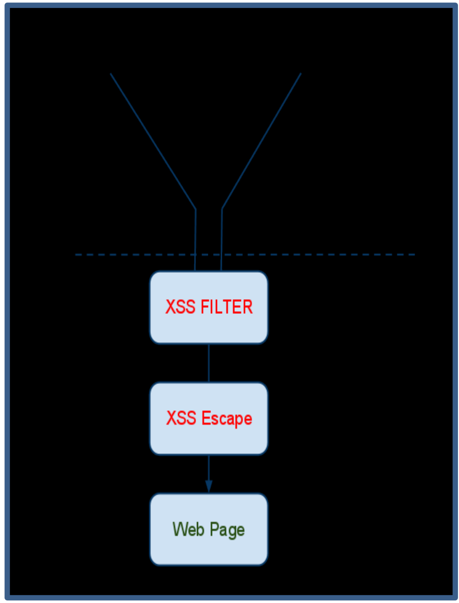
Cross Site Scripting

- What can go wrong
  - Reading data from URL (session id)
  - Replace data in the URL
  - Accessing/Replacing hidden form variables
  - Loading foreign web page into a frame inside your page
    - Using JavaScript to read and manipulate that frame
    - Using code in the frame to monitor your activities
  - Spying on everything the user does on the page
- Taking control of the browser


1. Attacker inserts malicious unfiltered code into application
Attacker
2. User visits web page and malicious code is returned with the web page
3. Attacker gains control over users data or system via injected exploit
Regular User

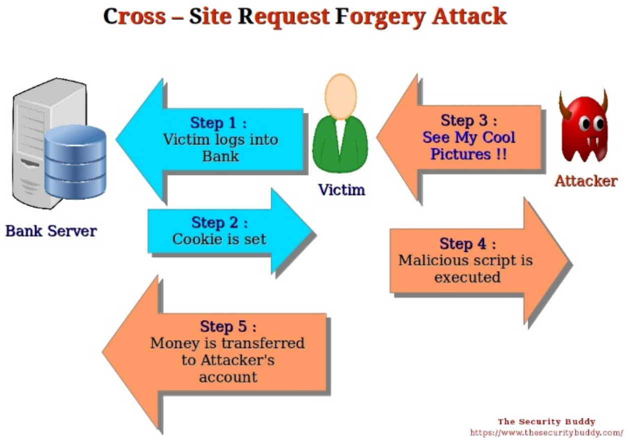
Cross Site Scripting: Prevention

- Don’t allow any HTML to be inserted
  - Back end libraries to strip out all HTML tags
  - But this limits the user in some ways
- Don’t allow malicious HTML to be inserted
  - Back end libraries to sanitize HTML
    - Limited set of allowed tags for formatting
- Use something other than HTML
  - Mark-up languages
  - Map to html on output
  - Still sanitize (don’t trust the input)



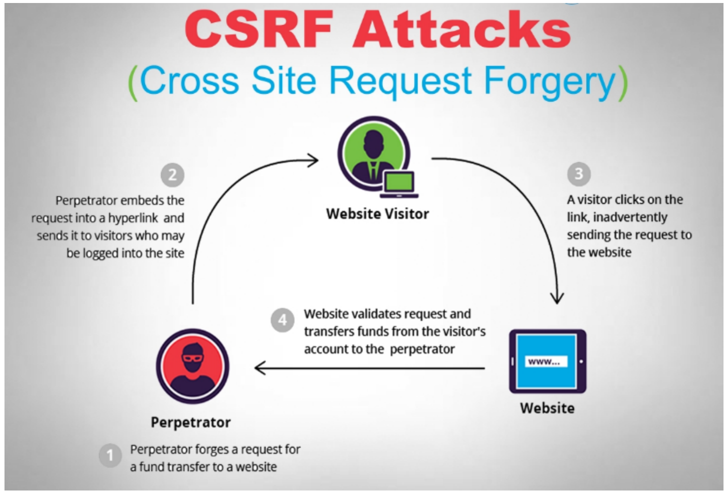
Cross-Site Request Forgery

- Suppose user logs into banking application
  - Cookies used to validate user/session
  - Suppose there is a URL that transfers money
    - transfer?from=checking&to=43434&amt=1000000.00
- Agent puts an ad on a different page
  - Clicking on the ad, generates that URL
- What happens if user clicks on the ad
  - While logged into banking application
  - Back end can't distinguish from real thing


Cross – Site Request Forgery Attack
Bank Server
Step 1 : Victim logs into Bank
Victim
Step 3 : See My Cool Pictures !!
Attacker
Step 2 : Cookie is set
Step 4 : Malicious script is executed
Step 5 : Money is transferred to Attacker's account
The Security Buddy
https://www.thesecuritybuddy.com/

Cross-Site Request Forgery

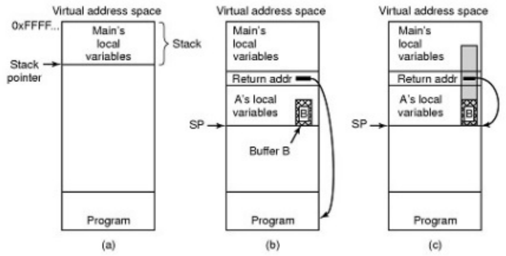
- Use a random value for each request
  - Set on previous request, kept in session
  - Passed back as part of any request
  - Validated by server before the action
  - Effective URL will be different each time
    - Can't be spoofed by another client (easily)
- Can be passed to client in various ways
  - Sent as part of html or XMLHttpRequest
  - Included in a hidden form field
  - Can also be put into header for use by JavaScript
  - Packages exist to support this


CSRF Attacks
(Cross Site Request Forgery)
Website Visitor
2 Perpetrator embeds the request into a hyperlink and sends it to visitors who may be logged into the site
3 A visitor clicks on the link, inadvertently sending the request to the website
4 Website validates request and transfers funds from the visitor's account to the perpetrator
www...
Website
Perpetrator
1 Perpetrator forges a request for a fund transfer to a website

Server Attacks

- Inputs from web page attack the server directly
  - Bugs in the web server
- Outside login to server
  - Weak passwords, user ids
  - Access from other machines
- Server becomes compromised
  - Phishing attacks

Buffer Overflow



- (a) Situation when main program is running
- (b) After program *A* called
- (c) Buffer overflow shown in gray

Lec 19
Fig 1

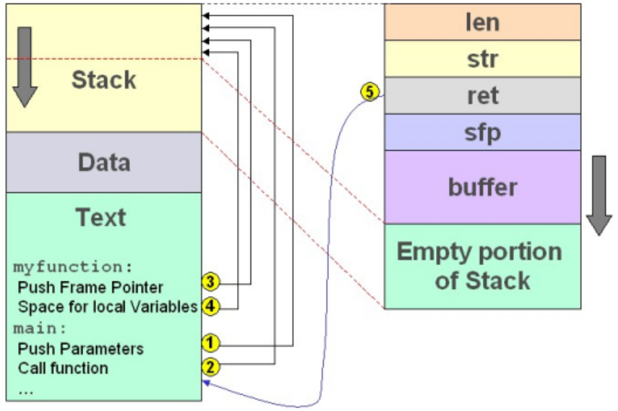
Buffer Overflow Attack

- Code:

```
void function(char* text) {
        char buf[1000];
        strcpy(buf,text);
        // do some editing of buf
        // save result
}
```

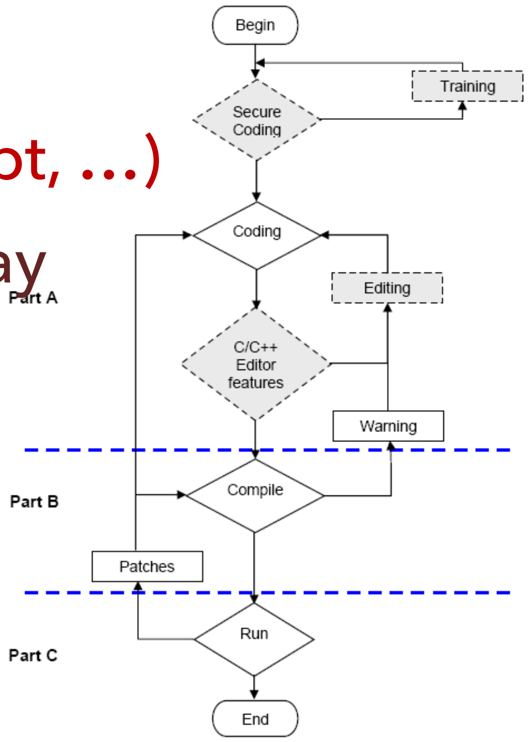
- Stack (high to low)

  8888: <ptr to text>
  8884: <return address>
  8880: <old stack ptr>
  7880: buf[0 .. 999]



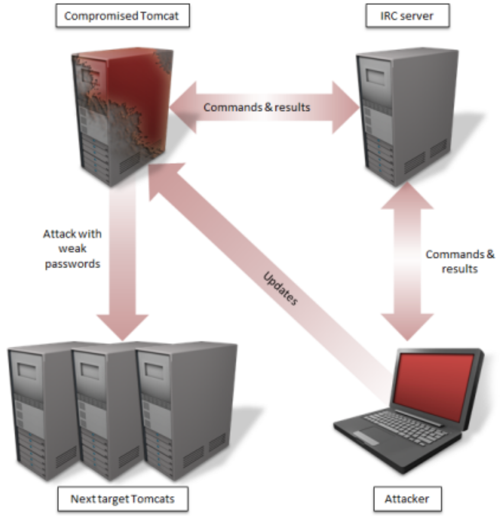
Preventing Buffer Overflow

- Use "safe" languages (Java, C#, JavaScript, …)
- Check sizes of data before putting in array
  - Reads, copies, inputs
  - Use safe functions (strncpy, snprintf, …)
  - Safe programming – don't cut corners
- Randomize code locations between runs
- Don't let data pages be executable



Server Attacks

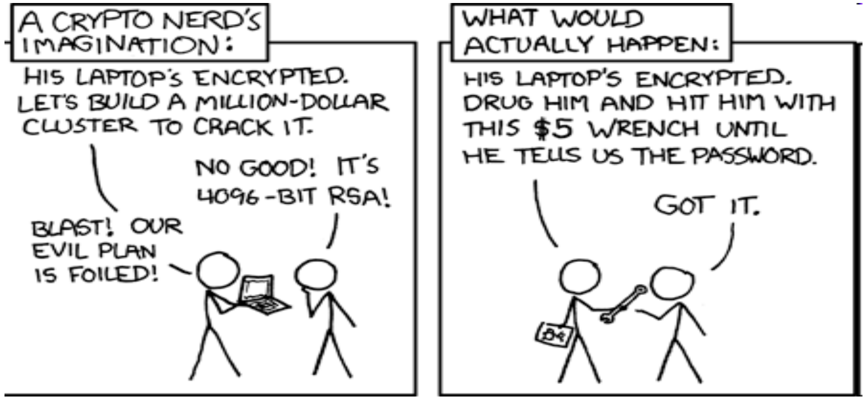
- Can buffer overflow happen in Java? JavaScript? Php?
  - Java/PHP/… security problems
    - File access, exec, eval, …
    - Internal bugs
- Even safe languages can have problems
  - Out of memory
  - Out of file space
  - Run arbitrary code (malicious servlets)
  - Tie up server for long time

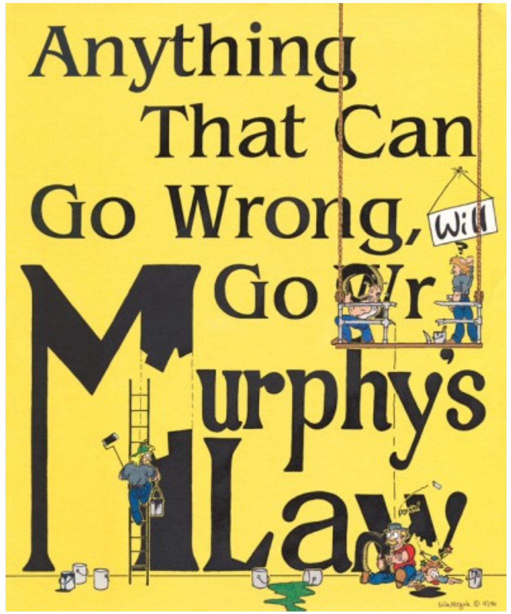


What Else Can Go Wrong?

hack

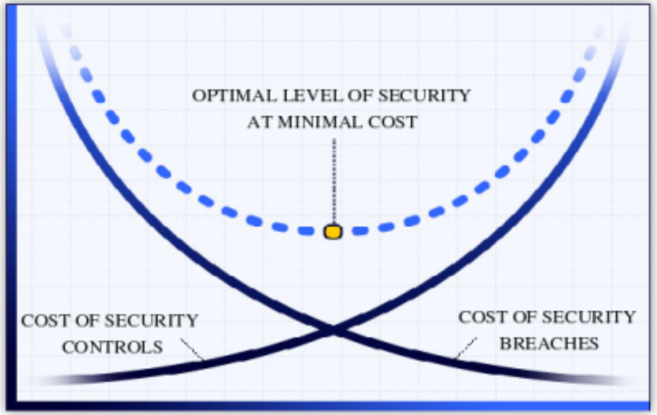
- PEOPLE
- Denial of service
- Timing attacks …

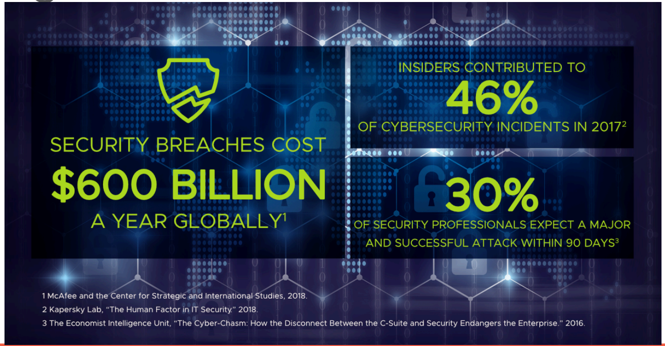

A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.



Security is Relative

- No application is totally secure
  - Any app or system can be broken
  - But you can control the cost to break it
- Make your application as secure as necessary
  - Cost to break much greater than value of breaking it





SECURITY BREACHES COST **$600 BILLION** A YEAR GLOBALLY[1]

INSIDERS CONTRIBUTED TO **46%** OF CYBERSECURITY INCIDENTS IN 2017[2]

30% OF SECURITY PROFESSIONALS EXPECT A MAJOR AND SUCCESSFUL ATTACK WITHIN 90 DAYS[3]

1 McAfee and the Center for Strategic and International Studies, 2018.
2 Kapersky Lab, "The Human Factor in IT Security." 2018.
3 The Economist Intelligence Unit, "The Cyber-Chasm: How the Disconnect Between the C-Suite and Security Endangers the Enterprise." 2016.

Next Time

- Security III (logging in)
- Privacy
- Testing