Sample CS 142 Final Examination

Winter Quarter 2016

You have 3 hours (180 minutes) for this examination; the number of points for each question indicates roughly how many minutes you should spend on that question. Make sure you print your name and sign the Honor Code below. During the examination you may consult two double-sided pages of notes; all other sources of information, including laptops, cell phones, etc. are prohibited.

I acknowledge and accept the Stanford University Honor Code. I have neither given nor received aid in answering the questions on this examination.

_____

(Signature)

Solution

_____

(Print your name, legibly!)

_____

(SUID - stanford email account for grading database key)

Problem	#1	#2	#3	#4	#5	#6	#7	#8	#9	
Score										
Max	12	12	8	10	10	8	8	8	12	
Problem	#10	#11	#12	#13	#14	#15	#16	#17	#18	Total
Score										
Max	12	8	8	12	8	10	12	8	14	180

Problem #1 (12 points)

A. (6 points) Explain why cloud computing platforms work well for web application startup companies that are starting small but hoping to make it big in a hurry.
B. (6 points) Explain why it is easier for a web application with many geographically distributed users to deliver read-only content such as images to its users' browsers than non-read-only content.

1A)

There are lots of reasons but one of the dominate one is the pricing model of billing based on resources used. Rather than having to spend a bunch of money by servers the startup hosted in a cloud expenses start low and grow with the resources uses.

1B)

They can use a Content Distribution Network (CDN) to distribute the read-only content to be close to the app's geographically distributed users.

Problem #2 (12 points)

A. (7 points) Give an example of a denial of service attack that a user could do on your Project #8 photo sharing app and describe how you could change the app to defeat the attack.
B. (5 points) Explain why you should make a habit of looking at the URL bar of your browser when using a web application from a trusted site such as a bank. Describe what you are trying to detect.

2A)

One way for a malicious attacker to bring down our photo sharing app would be to completely fill the storage by submitting numerous photos to the app.

To prevent this, we can simply set some kind of resource quotas or rate limiting mechanism, such as a limit for the number of photos per hour per user.

2B)

You are trying to detect a phishing scam. Some malicious websites can appear exactly the same as a legitimate website (such as a bank) but have a different URL.

Another thing to look for is for SSL stripping (HTTPS to HTTP)

Problem #3 (8 points)

When exploring the MongoDB objects of a photo app after a security penetration tester had been running on the system, you noticed several users were created with weird names like `{{1+1336}}`. Give an educated guess at what the security tester was doing by creating a user with weird name like this. Describe the security loophole and what the penetration tester was hoping to see if this loophole was present.

The evidence we have:

1) A security tester is running
2) We discover users with weird names "{{1+1336}}"

The "{{" in the MEAN stack applications is used by the AngularJS to evaluate expressions. One educated guess is the tester is trying to see the Angular express enter as user data ever gets evaluated. For example, if the user names showed up as "1337" the tester would have discovered a way of getting the JavaScript in run Angular expressions in a form of a stored cross-site scripting attack. Most likely in our photo app since we use angular to fill names into templates it would property display the weird name rather than evaluate it.

Problem #4 (10 points)

A. (4 points) What is the difference between HTTP and HTTPS (one sentence)?
B. (6 points) Describe how a web server can tell if an attacker in the browser has tampered with the session information stored in cookies it sends down to the browser.

A. HTTPS is a version of HTTP (both communication protocols) that uses encryption and SSL/TLS to provide more secure communication.

B. A server can provide a message authentication code (MAC) to encrypt the cookie. When it receives the cookie and MAC back from the browser it can check the MAC to see if the information has been tampered with.

Problem #5 (10 points)

A. (4 points) Explain why a web application coding standard might demand that the script tag for including JavaScript libraries look like: `<script src="/library.js">` instead of containing the full site information like: `<script src="http://www.site.com/library.js">.`
B. (6 points) Describe the mechanism used by a web application to prevent its session cookies from being used by a web application from a different company.

5A)

The first form of the script tag takes the protocol, typically HTTP and HTTPS and host name from the enclosing page where the second form hard codes the protocol HTTP and host. If this page is served with HTTPS the second form will generate a mixed content problem by fetching code using HTTP.

5B)

The browser's same origin policy stops a web app from a different company from reading the cookies of a different company's web app.

Problem #6 (8 points)

A. (4 points) Describe two different ways information such as parameters can be sent from the browser to a web server using a HTTP POST request.
B. (4 points) Describe how a browser decides how long it can cache a web page it fetched with a HTTP GET request.

A - Information can be passed through the request body or the request url.
Example from the photo share app:

- The request '/commentsOfPhoto/:photo_id' passed in the id of the photo in the url that was then fetched by request.params.id in webServer.js.
- The api request /admin/login passed in the login information as a JSON object that was later fetched by request.body in webServer.js.

Note: some students wrote $http and $resource. These are two Angular services that are used to make requests ($http for general AJAX, $resource for REST), not different ways information such as parameters are sent in a request.

B - When the browser sends a GET request, it receives a response in return. In the header of that response, the cache-control property specifies how long that data lives in the cache.

Note: Specifically mentioning cache-control wasn't necessary to get full credit. A response that described the HTTP response header was sufficient. Responses that alluded to cache-control without the context of the HTTP response header were given partial credit.

Problem #7 (8 points)

In a MEAN stack applications state, what is the order that the Model, View, and Controller (MVC) components typically arrive at the browser? It is OK to answer that one or more of them arrive at the same time.

Accepted Answers:

1) Controller, view arrive together.

2) Model is fetched afterwards.

OR

1) Controller arrives first.

2) Controller then fetches model and populates the view.

Problem #8 (8 points)

REST apis are frequently described as doing CRUD. What does CRUD mean in this context?

Create
Read
Update
Delete

Problem #9 (12 points)

A. Explain how the node.js Buffer class avoids having to allocate in the JavaScript heap every byte of a file being fetched via an HTTP get request to a browser.
B. Explain why a JavaScript function that takes multiples seconds to compute (e.g. some cryptographic library functions) are problematic in a Node.js server. Describe the problem.

Accepted Answers:

A. The Buffer class returns a buffer and a pointer to the start of the buffer. It also optimizes by using pointers rather than always copying and it supports operations for picking values out or updating them without having to read the entire file so it doesn't need to allocate space for every single byte.
B. Since node.js is single-threaded and uses asynchronous calls to get around that. This is where the problem comes in. Some callbacks depend on the completion of another callback function. This is blocking code and when functions take several seconds, nothing else can execute during that time.

Problem #10 (12 points)

A student doing CS142 Project #6 wrote the below code and discovered it didn't work correctly. When the execution reached the block with "`Process newComments and send response`", it always found `newComments` to be an empty array. Describe why this is the cause and what the student should do to fix the problem.

```
var newComments = [];
async.each(photo.comments, function (com, done_callback) {
    com = JSON.parse(JSON.stringify(com));
    User.findOne({ id: com.user_id}, function (err, user) {
        com.user = JSON.parse(JSON.stringify(user));
        newComments.push(com);
    });
    done_callback(err);
}, function (err) {
    if (err) {
        // Do error response
    } else {
       // Process newComments and send response
    }
});
```

The async.each function has three parameters:

1. Collection: the collection to iterate over
2. Iteratee: a function to apply to each item in the collection
3. Callback: a function that runs when all the iteratees have run, or if an error has arisen

The async function runs the callback function only after each iteratee has called done_callback.

Given the asynchronous nature of JavaScript, the callback function passed into User.findOne is not guaranteed to run before done_callback is called. It is highly likely that done_callback is being called even before User.findOne executes its callback function and pushes com to newComments. As a result, the async function thinks all of its iteratees are done running and executes the 'process newComments and send response' block of code before User.findOne successfully adds all the comments to newComments. This is why newComments is always an empty array.

To fix the problem, move the call to done_callback(err) INSIDE the callback function that's passed into User.findOne. This guarantees that the async function processes newComments and sends the response only when the appropriate data is done being added to newComments.

Problem #11 (8 points)

Give an example of functionality that is implemented using Express middleware. Briefly describe how this middleware works.

Express has a middleware for dealing with the session state called 'express-session'

This middleware handles the creation and fetching of the session state for request handlers, as well as storing/mapping the session state on the backend (session state store) and front-end (cookie).

Problem #12 (8 points)

The definitions that the Mongoose system we used for the photo-share app looked much like an ORM (Object Relational Mapping) that was used in older frameworks like Rails. Explain why it would be incorrect to call the grouping of those definitions a ORM.

Object Relation Mapping names implies there is some kind of mapping to a relational model that is not happening when we use a object database like MongoDB.

Problem #13 (12 points)

A. (6 points) Explain how indexes make database queries go faster.
B. (6 points) Describe the disadvantages of having indexes on every property of an commonly updated object.

A. Indexes make it easier to efficiently retrieve information from a table instead of having to do a row-by-row scan of the database. They provide a map between an index the requested data, allowing for fast lookup.
B. The index must be updated every time the object is updated, which adds overhead to the object update time. Additionally, indexes may be large, and the space used to store them may outweigh the benefits in lookup time.

Problem #14 (8 points)

Explain why we frequently end up validating input in a web app twice: once in browser and then again in the web server.

The validation is done in the web app to provide quick feedback to users about errors. The validation done in the web server is needed to prevent bogus date to be push into the storage. Because the web api is available to untrusted code running in the same browser as the user we can not depend our web app is the only code generate web server API calls.

Problem #15 (10 points)

For each of the following web application security attacks, state if the focus of defeating the attack would be more on a the frontend (i.e. browser-side) or the backend (e.g. web server and storage system) . Justify your answer.

A. CrossSite Request Forgery
B. Cross Site Scripting Attack
C. SQL Injection
D. Phishing Attack
E. Denial Of Service Attack

15A)
We described two techniques for defeating CSRF. One invoked designing the backend web app to avoid HTTP verbs that an attacker could generate (GET and POST with arguments than can be generated via a form). The other was to have the backend generate a secret in the form that is passed to the front-end and checked on the backend.

15B)
It is important than the frontend code not allow user input to be injected into the DOM.

15C)
The backend should only generate proper SQL commands and not let SQL commands embedded in input data be executed.

15D)
Mostly the frontend should provide some indication of where the page came from using HTTPS certificates.

15E)
The backend needs to protect itself from DoS attacks.

Problem #16 (12 points)

Browsers and web servers communicate using the HTTP protocol. Even though all HTTP communication follows a request-response pattern, information can be viewed as either being passed from the browser to the web server or passed from the web server to the browser. For example, the Angular photo sharing application you built in Project #8 and your webServer.js pass back and forth all the major components of the application including: Angular HTML templates, CSS style sheets, JavaScript libraries, model data, and Angular controllers. For each of these components state which **direction** (browser to web server or web server to browser) and which **HTTP method/verb** is used for the transfer. Note some the components are transferred in both directions so list both directions and HTTP methods.

A. Angular HTML templates

   Web server to browser. GET

B. CSS style sheets

   Web server to browser. GET

C. JavaScript libraries

   Web server to browser. GET

D. Model data

   Web server to browser. GET
   Browser to web server. POST

E. Angular controllers

   Web server to browser. GET

Problem #17 (8 points)

A. (4 points) Explain what the pyramid of doom complaint is about Node.js and show an example of it.
B. (4 points) Explain why you typically don't see functions like sleep(10*1000); (sleeps for 10 seconds) in Node.js

17A)

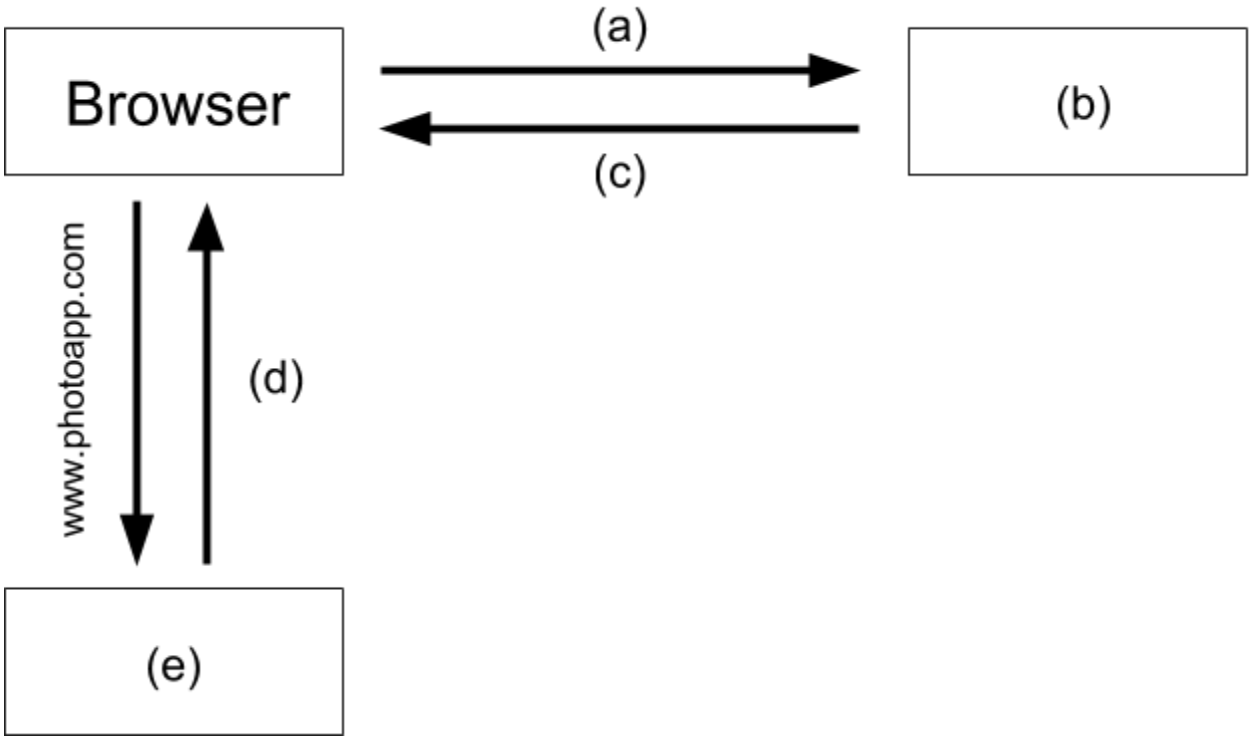
The pyramid of doom is when you have nested callbacks that cause the code to form a pyramid that impairs readability of the code. Example:

```
fs.ReadFile(fileName, function (error, fileData) {
    doSomethingOnData(fileData, function (tempData1) {
        doSomethingMoreOnData(tempData1, function (tempData2) {
            finalizeData(tempData2, function (result) {
                doneCallback(result);
            });
        });
   });
});
```

17B)

Node.js doesn't support blocking calls so something like a sleep call would have a callback function argument that would be called when the sleep is done.

Problem #18 (14 points)



1) This diagram outlines the process of displaying a web page when `www.photoapp.com` is typed into the location bar of the browser for the first time. For each of the letters in parentheses, select a value from the "Possible answers" section below that best describes the service or communication event the letter is marking.
    a) HTTP GET request
    b) Web server
    c) HTTP GET response
    d) DNS response
    e) DNS server
2) Once the arrows are labeled, write down the order that these events (arrows only) happen when `www.photoapp.com` is typed into the browser.
    1. DNS lookup
    2. DNS response
    3. HTTP GET request
    4. HTTP GET response

Possible answers: browser, web server, storage server, memcache server, DNS server, HTTP GET request, HTTP POST request, HTTP GET response, HTTP POST response, database query request, database query response, memcache GET request, memcache GET response, DNS lookup, DNS response