CS142 Final Exam Spring 2018 Solutions

Problem #1 (10 points)

Part A - 4 points
The key advantage is that the session is readily available with the request.
Minor but plausible advantages without justification received 2 points (e.g. less server memory, etc.)
Note: Students that stated a minor advantage but motivated it well received full credit.

Part B - 6 points
Attack: User has access to session state and can modify it, e.g. to impersonate another user and hijack their session
To prevent: Use a MAC to detect tampering

[3] for describing the attack
[3] for describing the prevention mechanism

Problem #2 (10 points)

Part A - 4 points
We were using $scope variables to keep track of whether a user was logged in, so a refresh purged the front end JS variables in memory.
Note: request.session would actually still be populated with the logged in user, but the front end doesn't know that

[0] student argues that refresh purges the Node.js (server) memory
[0] student argues that "session state is destroyed" on refresh (too vague)

Part B - 6 points
Two full-credit solutions are accepted:

1. Server and front-end strategy
    a. Add a GET /admin/login endpoint or equiv. that returns whether a user is already logged in by examining request.session
    b. Whenever MainController first runs (which happens at the start of every refresh), send a GET to /admin/login to check if the user is logged in
2. Front-end only strategy
    a. Use localStorage or sessionStorage (both are robust to refresh) to store the notion of a logged in user instead of a $scope variable.

Problem #3 (10 points)

The browser communicates with the web server using HTTP which is a request-response protocol. We use this for fetching models so modeling fetching requests originate from the browser (specifying what model data is needed) and "pull" the model data from web server.

The case where a "push" would be useful is when something changes on the backend and we would like to update the view in the frontend browsers. For example, when one user posts new content like photos or comments it would be desirable if the browser could be informed and have these update models "pushed" to it.

Problem #4 (8 points)

Being "model data" just means it is part of some view shown to the user. Session data is distinguished by its usage and lifetime. It is only kept as long as the current session is active. There is nothing that precludes a view from wanting to display something that is part of the session state. For example, in our Photo Sharing app we keep the logged in user id. We could certain have a view that displays that information making it model data as well.

Problem #5 (8 points)

We should simply require that our web application be accessed via https, and redirect all requests that come in over http to their https counterparts. Communication between our servers and browsers will be encrypted (TLS) during https sessions.

Common deductions:
[-1] Describing something similar to https without mentioning https
[-2, -3, -4] Vague or lacking detail (commonly related to key exchange)

NOTE: Public key encryption is very slow compared to symmetric encryption, so just using public key cryptography would not be scalable in a real system.

Problem #6 (8 points)

The attacker does not actually control www.mybank.com, and will not be able to obtain a valid certificate associating www.mybank.com with the bogus IP address. When we attempt to connect to the attacker's server over https, our browser will warn us that the certificate is not valid. As long as we don't click through that warning, we are protected.

Common deductions:
[-8] Incorrect reasoning and no mention of certificates.
[-6] Incorrect reasoning about certificates.
[-2, -3, -4] Mostly correct answer but some flaws with reasoning.

Problem #7 (8 points)

Since the attacker has full control over bankofthevvest.com, they could have obtained a certificate from a certificate authority that certifies that they own the server for the site. So a browser would not be able to detect that the site is not real.

With extended validation - which shows up with the certified owner in the url - it may be possible to tell that the site is not controlled. There would still be no active warning.

General Rubric (with some leeway given for showing more/ less understanding of how certificates and phishing work)
[8] Answers that said no because the site would have a certificate or yes because of extended validation (explaining that the site did have a certificate)
[6] Answers that said no, but didn't quite give a good explanation of certificates, phishing, and certificates
[3] Answers that said yes, for reasons of a bad certificate or anti-phishing warnings and gave the plausible ways this could show up (warning dialogues, etc.)
[1] Answers that said there would be no https lock symbol

Problem #8 (10 points)

Tables would map to collections of objects of the same class/ schema
Rows would map to instances of the objects
Columns would map to properties of objects

Primary indices would map to each objects' userID, are how the data is organized in the table, for primary lookup
Secondary indices would map to other unique properties of the object that would be used in some sort of lookup table to speed up querying

[+2] For each explanation of mappings for **tables, rows, columns, primary indices, and secondary indices** (credit given for showing an understanding of how each of these things work)

Other common Deductions:
[-2] For just saying primary and secondary indices would map to the same thing in ORM (no explanation of what they do given)
[-1] For not explaining differences between the primary and secondary indices

Problem #9 (8 points)

- Full points were awarded for answers that, in addition to the below, mentioned Angular's watch and digest cycles.
- 6 of 8 points were awarded to those who grasped the conceptual purpose of $scope.$apply, i.e., mentioned the need to update variables, two-way binding, etc.
- Additional points were taken off depending on vagueness of the answer, repeating the question, etc.

Problem #10 (8 points)

- 4 points were awarded to those that noted that the code could have returned an empty array. The other 4 points were awarded to those that noted that the code could be fixed by moving the response.status line to inside the callback.

Problem #11 (8 points)

In general the backend must validate everything to protect itself. For validation in the backend pretty much any example explaining this protection is needed was accepted. For frontend only validation examples checks done but not sent to the backend like the confirmation password and the OK to delete dialog were good examples.

Problem #12 (8 points)

In order to read an HTTP request you need to parse enough of the request it get the length from the header. This means that some HTTP parse (and knowledge of how to do it) is present in both the read and parse modules.

Problem #13 (8 points)

We need HTTP to communicate with the server in order to retrieve data/model, perform database queries, manage sessions, etc [+2]
Due to same origin policy, a webpage fetched with “file://” protocol cannot initiate HTTP (a different protocol) requests, thus no communication with the server would be possible [+6]

Partial Marks:
“file://” protocol won’t be safe if it is allowed to be used within web pages fetched with HTTP, security risk [+1]
Real webapp doesn’t exist on local disk [+2]

Problem #14 (10 points)

A. CSRF is an attacker where the attacker takes advantage of the cookie/session of a legitimate site that is stored in the browser. Because all HTTP requests to a certain origin will have its cookie attached to it, an attacker can trick the user into submitting a form to the legitimate site (given the user has logged in to that site, session exists) while the browser will attach the right cookie, this request will seem legitimate to the victim site and actions will be performed (e.g transfer money to a bank) [+5]
B. JavaScript frameworks disallow requests to different origins (same origin policy), so any JavaScript utilities like XMLHTTPRequest or $resource can’t be taken advantage of. This is enforced by the browser! Not server [+5]

Acceptable answers with partial credit (even if mention multiple, will still only grant mark for one: the one with maximum point):
Get requests shouldn’t change the state of the server, aka requires server to be implemented properly [+3]
Since CSRF needs to be done from form submission, the request types are limited to POST and GET [+2]
Form submissions cannot generate requests akin to ones generated by JavaScript frameworks with JSON bodies, e.g. $resource, so the server can just NOT accept form POST submissions [+2]
Forms can be generated with tokens to identify its legitimacy, this is hard to forge [+2]

Problem #15 (12 points)

Solutions:
A. foo
B. N/A (also accept status code 404)
C. Many acceptable answers (e.g., GET /test2/10)
Note: Answers must include a valid url param after the /test2 (e.g., GET /test2 or GET /test2/ does not trigger the desired response. also, anything of the form :something in GET /test2/:something is not a valid url param)
D. GET /test2/4?test3=5
Note: In particular, GET /test2/4?q=5 is incorrect

Rubric (max: -12):
A: (max: -3)
-3: incorrect answer
-3: supplied multiple responses
B: (max: -3)
-3: incorrect answer
-3: supplied multiple responses
C: (max: -3)
-1: incorrect answer
-1: url param is :4 instead of 4
-2: forgot to include valid url param after the /test2 (see example in solutions about this)
-1: url path does not start with /test2
D: (max: -3)
-1: incorrect answer
-1: incorrect/missing url params
-1: incorrect/missing query string (note: need to supply a query string)
-1: incorrect verb
-1: url path does not start with /test2
-1: url param is :4 instead of 4

Problem #16 (10 points)

Solutions:
Part 1.
4

4
2
2

Part 2.
Yes.
4
4
0
1

Rubric (max: -10):
Part 1: (max: -5)
-1: missing exactly 4 numbers
-2: incorrectly listed order (exactly two 4s should be on top, followed by at least one predicted loop indices)
-2: missing exactly two instances of '2'

Part 2: (max: -5)
-5: answering no
-1: missing exactly 4 numbers
-1: incorrectly listed order (exactly two 4s should be on top, followed by at least one predicted loop indices)
-3: missing exactly one instance of 0 or missing exactly one instance of 1, or presence of a number other than 0, 1, 4

Problem #17 (8 points)

GraphQL is better suited for these network conditions for two reasons. GraphQL allows clients to specify which properties of the resource it is interested in retrieving (good for low bandwidth). In addition, GraphQL allows clients to fetch from many different resources in the same request (i.e. entire model in one query), which reduces the number of roundtrips to the server (good for high latency).

Problem #18 (8 points)

CDNs deliver static content/files from a distributed network of servers which are located close to clients (e.g. inside ISP server racks). It takes time for changes to propagate, so it only works for content that doesn't need to change often.

The model would be most inappropriate to place in a CDN, because for example in our photo sharing application, the model data is frequently changing as users add comments and photos. Both controller JS files and view html templates would be appropriate to put in a CDN. However, it is reasonable to explain these files could be expected to change frequently due to code updates/changes by developers, and thus developers would choose not to put it in a CDN.

Incorrect or missing justification for answers lost partial credit. In addition, the answer should reference MVC as the question asks, not just talk about CDNs in general.

Problem #19 (10 points)

(1) Cross Site Scripting attack launching an SQL Injection Attack does not make sense. XSS attack is mainly an attack on the front end / browser, while SQL Injection is attack on the backend. A lot of people answered that it is possible to run a script inside the victim’s browser which sends SQL queries that carry out the injection attack. However, if that is possible, the attacker can just launch the injection attack directly (running the script), without the need to rely on XSS. 5 points given to correct answer, partial credit given to answers that are not completely correct but are giving a more or less reasonable explanation.

(2) SQL Injection launching XSS makes sense, since if the attacker manages to insert a field with a script tag into the database, and the web app is using model data from the database to render its template, then chances are that the script tag would be inserted into the DOM and get executed, launching an XSS attack.5 points given to correct answer, partial credit given to answers that are not completely correct but are giving a more or less reasonable explanation.

Problem #20 (10 points)

We are expecting “phishing” (10 points with reasonable explanation). Partial / full credit given to students who answered otherwise but managed to give a convincing explanation.