Web Security

Question 1 ***True/false***

Q1.1 True or False: Under the SOP, it is possible for two webpages with different origins to communicate through narrowly defined APIs.

● True ○ False

Solution: True. This is the postMessage API.

Q1.2 True or False: Under the SOP, the webpage at `https://example.com/randompic.html` cannot fetch the image at `https://cute-cats.com/cutest.jpg` because they have different origins.

○ True ● False

Solution: False. A page can fetch images and content regardless of origin; the SOP only prevents it from determining detailed properties of cross-origin content.

Q1.3 True or False: Suppose the webpage at `https://example.com/index.html` contains a child frame that loads `https://another-example.com/index.html`. Under the SOP, the parent frame can read and modify the properties of the child frame.

○ True ● False

Solution: False. Because `example.com` and `another-example.com` have different domains, they are considered different origins and are therefore unable to access each other directly.

Q1.4 (2 points) True or False: Suppose the webpage at `https://example.com/index.html` contains a child frame that loads `https://example.com/views.html`. Under the SOP, the child frame can read and modify the properties of the parent frame.

● True ○ False

Solution: True. Pages are allowed to directly access child frames from the same origin, and vice versa.

Q1.5 TRUE or FALSE: Suppose the webpage at `https://example.com/index.html` loads and runs an external script from `https://sample.com/script.js`. Under the SOP, the script runs with the same origin as `https://sample.com/script.js`.

○ TRUE ● FALSE

Solution: False. External scripts run with the origin of the page that fetched them (in this case, `https://example.com`).

Q1.6 TRUE or FALSE: Mallory convinces Alice to try out her custom browser, FireFaux. Webpages Alice visits using this browser may no longer be subject to the SOP.

● TRUE ○ FALSE

Solution: True. The SOP is enforced by the browser; if the browser is compromised, there is no guarantee that webpages will play by the rules.

Question 2 ***Cauliflower Smells Really Flavorful***

`califlower.com` decides to defend against CSRF attacks as follows:

1. When a user logs in, `califlower.com` sets two 32-byte cookies `session_id` and `csrf_token` randomly with domain `califlower.com`.
2. When the user sends a POST request, the value of the `csrf_token` is embedded as one of the form fields.
3. On receiving a POST request, `califlower.com` checks that the value of the `csrf_token` cookie matches the one in the form.

Assume that the cookies don't have the `secure`, `HTTPOnly`, or `Strict` flags set unless stated otherwise. Assume that no CSRF defenses besides the tokens are implemented, and that CORS is not in use (if you don't know what that means, do not worry about it). Assume every subpart is independent.

Q2.1 Suppose the attacker gets the client to visit their malicious website which has domain `evil.com`. What can they do?

☐ (A) CSRF attack against `califlower.com`

☐ (B) Change the user's `csrf_token` cookie

☐ (C) Learn the value of the `session_id` cookie

■ (D) None of the above

☐ (E) ——

☐ (F) ——

Solution: The attacker's website is of a different domain so they are not able to change/read any cookies for `califlower.com`. As such, they not able to execute a CSRF attack since they can't guess the value of `csrf_token`.

Q2.2 Suppose the attacker gets the client to visit their malicious website which has domain `evil.califlower.com`. What can they do?

■ (G) CSRF attack against `califlower.com`

■ (H) Change the user's `csrf_token` cookie

■ (I) Learn the value of the `session_id` cookie

☐ (J) None of the above

☐ (K) ——

☐ (L) ——

Solution: Since the attacker's website is a subdomain for `califlower.com`, it can read/set cookies. The attacker can embed Javascript in their page to extract `csrf_token` and form a malicious POST request.

Q2.3 Suppose the attacker gets the client to visit a page on the website `xss.califlower.com` that contains a stored XSS vulnerability (the website `xss.califlower.com` is not controlled by the attacker). What can they do?

■ (A) CSRF attack against `califlower.com`
■ (B) Change the user's `csrf_token` cookie
■ (C) Learn the value of the `session_id` cookie
☐ (D) None of the above
☐ (E) ——
☐ (F) ——

Solution: Utilizing the XSS vulnerability, the attacker can extract the `csrf_token` cookie and cause the user's browser to make a malicious POST request.

Q2.4 Suppose the attacker is on-path and observes the user make a POST request over HTTP to `califlower.com`. What can they do?

■ (G) CSRF attack against `califlower.com`
■ (H) Change the user's `csrf_token` cookie
■ (I) Learn the value of the `session_id` cookie
☐ (J) None of the above
☐ (K) ——
☐ (L) ——

Solution: The attacker can observe `session_id` and `csrf_token` in plaintext and forge a POST request. Also, they can spoof a response to the POST request, and include a Set-Cookie header in the response to change the `csrf_token` cookie.

Q2.5 Suppose the attacker is a MITM and observes the user make a POST request over HTTPS to `califlower.com`. What can they do?

☐ (A) CSRF attack against `califlower.com`
☐ (B) Change the user's `csrf_token` cookie
☐ (C) Learn the value of the `session_id` cookie
■ (D) None of the above

☐ (E) —— ☐ (F) ——

Solution: Nothing, a MITM can't break learn/change the cookie values without breaking TLS.

Q2.6 Suppose the attacker is a MITM. The victim uses HTTP and is logged into `califlower.com` but will not visit `califlower.com` at all. Describe how this attacker can successfully perform a CSRF attack against `califlower.com` when the user makes a single request to any website. *(Hint: Remember a MITM can modify a webpage over HTTP since there are no integrity checks.)*

Solution: The MITM can modify the website's response to add an img tag or some sort of element that will cause the user's browser to make a request to `califlower.com`. The attacker can then extract `session_id` and `csrf_token` from the request.

Then there are two ways the POST request could be made. When the attacker forces the user to visit `cauliflower.com`, they can extract `csrf_token` and embed javascript in the response which makes a POST request alone with the hardcoded value of `csrf_token`. Or once the attacker has `session_id` and `csrf_token` they can make the request themselves.

Question 3 ***SQL Enumeration***

Alice runs a computing cluster. When a user wants to execute some job `$job`, they visit:

`https://alice.com/execute?job=$job`

Alice's server locally stores a SQL table named `dns`:

IP	hostname	jobs
10.120.2.4	gpus.alice.com	matrix-multiplication
10.120.2.75	cpu1.alice.com	matrix-addition
10.120.2.6	cpu2.alice.com	matrix-addition
⋮	⋮	⋮

Upon receiving a request, Alice's server makes the following SQL query:

```
SELECT IP, hostname FROM dns WHERE jobs='$job' ORDER BY RAND() LIMIT 1
```

where `$job` is copied from the request parameter. This SQL query finds all hosts in `dns` whose `jobs` field equals the string `$job`, and randomly returns one of them. If successful, the job is sent to the specified IP, and the following webpage is returned:

Successfully launched job on `hostname`!

Otherwise an error code is returned. `hostname` is copied from the SQL query result.

Q3.1 What type of attack is the server vulnerable to?

● (A) SQL injection

○ (B) ROP attack

○ (C) CSRF attack

○ (D) Path traversal attack

○ (E) None of the above

○ (F) ——

Solution: The query is vulnerable to SQL injection since the statement is not parameterized and no escaping happens.

Q3.2 Alice modifies the server to check that `$job` contains only letters (a-z), dashes (-), quotes (’), and/or spaces ( ). If `$job` contains any other character, it rejects the request without making any SQL queries. Assume that the server’s code includes the entire response from the SQL query in the web page for debugging purposes.

TRUE OR FALSE: It is possible to choose a value for `$job` that will let Mallory learn all hostnames that can handle a `matrix-addition` job in a single visit to the web page. If you choose true, show such a value; if you choose false, explain why it’s no longer possible. *(Hint: `--` starts a SQL comment. Assume that it does not need to be preceded or followed by a space.)*

● (G) True ○ (H) False ○ (I) — ○ (J) — ○ (K) — ○ (L) —

Solution: `matrix-addition'--`

Q3.3 Instead of the checks in the previous part, Alice implements a simple filter on the value of `$job`:

```
1 def sanitize(job):
2     job = job.replace('--', '') // Deletes all occurrences
          of --
3     job = job.replace(';', '') // Deletes all occurrences
          of ;
4     return job
```

After calling `sanitize`, she checks that the result contains only letters (a-z), dashes (-), quotes (’), and spaces ( ), then uses it in the SQL query.

TRUE OR FALSE: It is still possible to choose a value for `$job` that will let Mallory learn all hostnames that can handle a matrix-addition job in a single visit to the web page. If you choose true, show such a value; if you choose false, explain why it’s no longer possible.

● (A) True ○ (B) False ○ (C) — ○ (D) — ○ (E) — ○ (F) —

Solution: `matrix-addition'-;-`