BASELINE SECURITY TEST PLAN FOR EDUCATIONAL WEB AND MOBILE APPLICATIONS

Published by Tony Porterfield
Feb 1, 2015.

Overview

The intent of this test plan is to evaluate a baseline set of data security practices that are straightforward for end users to observe from an authenticated session of the web service. Quantifying observable security practices of online services can give a metric for comparison and evaluation of the services. Furthermore, a measure of the rigor of the security practices that an end user can observe (and an intruder can exploit) provides insight in to the rigor of the overall security practices of the web service.

The tests defined in this document are focused on data security of browser-based web applications, and mobile apps that use HTTP. Most of the tests are based on the OWASP Application Security Verification Standard, 2014 (ASVS) and are a subset of the "Opportunistic" verification level defined in the ASVS.
The ASVS is available at: https://www.owasp.org/index.php/ASVS

The tests are divided in to two sections: Objective Tests and Subjective Tests. Note that this testing will require a valid user account for the web service under test.

Objective tests are straightforward to perform using a browser or app, proxy and cookie editor. Compliance to these requirements can be evaluated without much ambiguity or subjectivity. For 3rd party apps running within a host web app, a subset of these tests can be performed based on the functionality of the interface between the host web app and 3rd party web app.

Subjective tests require more complex or free form analysis, and evaluating compliance may be subjective. These are listed in rough order of importance.

The scoring or rating system is not defined here and will be added in later versions of the test plan. Scoring for vulnerabilities should be weighted based on the severity of the vulnerability and the sensitivity of the information that would be exposed if exploited. The Common Vulnerability Scoring System (CVSS) may provide a starting point for rating severity.

Summary of objective test sections

1. Login and authentication: Problems here can lead to passwords or sensitive information being snooped or logged, or session authentication tokens stolen.
2. Password reset: Sending the old or new password after a lost password request can expose the account's password to attackers. Furthermore, a site that can send the current password to a user is not properly hashing the password at the server.
3. Username enumeration: This tests how easy it is for potential attacker to test potential usernames for validity. This would be the first step in a brute-force password attack.
4. Auth. token/session ID and cookie handling: Incorrect protection or handling of authentication tokens could lead to an attacker stealing them and hijacking the session.
5. Sensitive information outside of message body: Sensitive information in URLs can be stored in browser histories and caches, and server logs, where they could be viewed by potential attackers and possibly leveraged to get more access.
6. Verify pages with sensitive information disable caching: Sensitive information left in browser disk caches could be viewed by others who use the same browser. Shared use computers are commonly used by students and teachers at schools. Students may also log in on shared computers in public places such as libraries.
7. Check for clickjacking protection: Clickjack attacks superimpose an attacker's page in an iframe over the original page. This is a low risk in educational applications but a straightforward protection to add.
8. Single access code grants access to sensitive data: many educational sites generate classroom codes that students can use to enter class groups on the site. If these allow access without a secondary check such as teacher approval, an attacker who gets or enumerates valid codes could enter and gather sensitive information about the other students in the class.
9. Password complexity: Though younger children may have trouble remembering long passwords, extremely short passwords increase the chance of a brute force password attack.
10. Excessive headers: Server and platform details in response headers can reveal information that could be useful to an attacker attempting to compromise the system.

Summary of subjective test sections

1. Direct object reference: can allow an attacker to access data without authorization by enumerating or otherwise constructing the URL for the page.
2. Session timeout: Protects a user against forgetting to log out on a shared computer, and against an attacker getting a session authentication token and using it indefinitely to access the account
3. Error messages: Error messages can reveal important details about a system to an attacker
4. XSS: Cross Site Scripting/Injection: An attacker can inject malicious scripts or HTML code to a user's browser.
5. Unsecured APIs: APIs that allow queries without proper authentication can allow attackers to extract significant amounts of information from the system by crafting API requests.
6. Brute-force login check: If login attempts are not rate limited, an attacker can brute force the passwords at a much higher rate. It's worth noting that passwords used by young students or assigned by schools are often simple or formulaic.

Notations:

- ASVS requirements are enclosed in (parens)
- Testing tools are enclosed in [brackets]
    - Client: web browser or mobile app
    - Cookie editor: allows viewing and modification of cookies, such as "Edit this cookie" for Chrome
    - Proxy: Allows inspection and modification of HTTP traffic generated by the client. OWASP ZAP, Burp Suite Proxy and Fiddler2 are examples of proxy programs.
    - Asafaweb: www.asafaweb.com: automated test for some security practices

TEST PLAN test date & notes

Web application name and URL:

Personal information that can be collected:

SSL Labs SSL score (https://www.ssllabs.com/ssltest/index.html)

Objective tests

1. Login and authentication:

Not Tested 1.1. Verify that password is not echoed to screen, and autocomplete is not enabled for password (V2.2) [Client]

Not Tested 1.2. Verify that login form is served using HTTPS (V2.16) [Client, Proxy]

Not Tested 1.3. Verify that login credentials posted using HTTPS (V2.16) [Client, Proxy]

Not Tested 1.4. Verify that authenticated sessions continue to serve pages using HTTPS. (V2.16)[Client]

Not Tested 1.5. Verify that all login checks (authentication controls) are enforced on server side (V2.4)[Client, Proxy]

1.6. Notes:

2. Lost password:

Not Tested 2.1. Request password reset, verify that old or new password is not sent in plain text to user (V2.17) [Client]

2.2. Notes:

3. Username enumeration (V2.18) [Client]

Not Tested 3.1. Attempt login with {valid username,invalid password} and {invalid username,invalid password}. Verify that responses are different for valid usernames and invalid usernames

Not Tested 3.2. Request lost-password reset for valid username and invalid username. Verify that responses are not different

Not Tested 3.3. Request lost username help and verify valid usernames can not be enumerated (if applicable)

3.4. Notes:

4. Authentication token/session ID and cookie handling

Authentication cookie name:

Not Tested 4.1. Auth/session cookie(s): httpsOnly flag (V3.14) [Client, Cookie editor]

Not Tested 4.2. Auth/session cookie(s): secure flag (V3.15) [Client,Cookie editor]

Not Tested 4.3. Verify that strict transport security headers are present (V3.15) [Proxy]

Not Tested 4.4. Verify auth/session is not sent in url (V3.6)[Client,Proxy]

Not Tested 4.5. Log out, restore auth/session cookie and verify it does not allow access (V3.2)[Client,Cookie Editor]

Not Tested 4.6. Verify that all pages that require authentication have logout links (V3.4) [Client]

4.7. Notes:

5. Sensitive information sent outside of message body

Not Tested 5.1. Examine URLs of HTTP requests for sensitive information (V9.3) [Client,Proxy]

5.2. Notes:

6. Verify that forms and other pages containing sensitive information have disabled client-side (disk) caching using Cache-Control directives (V9.1) [Proxy]

Not Tested 6.1. no-cache, no-store to prevent disk caching of sensitive information

6.2. Notes:

7. Check for X-Frame-Option header to protect against clickjacking

Not Tested 7.1. Examine response headers for XFO option (V11.8) [Client, Proxy, asafaweb.com]

7.2. Notes:

8. Check whether access to sensitive information can be gained with a single access code, and no other form of authentication

Not Tested 8.1. Examine how classroom groups or other user groups are formed – does a single class code provide access to sensitive information without further verification by teacher or other mechanism? [Client]

Not Tested 8.2. Are there other ways were an individual's or group's information can be accessed with a single authentication code. [Client]

8.3. Notes:

9. Check minimum password length and complexity requirements

9.1. Attempt to set password of length 1 character, if rejected record minimum length required. [Client]

9.1.1. Perform test during account creation (if possible)

9.1.2. Perform test for password reset

9.2. Notes

10. Excessive headers

Not Tested 10.1. Check for unnecessary information about server platform in response headers. [asafaweb.com]

10.2. Notes:

"Subjective" or more complex tests

Test specifics will be added to this section in future revisions

1. Access control/Direct object reference/Enumeration (V4.1-V4.5) [Client,Proxy]

Not Tested 1.1. Look for numeric user IDs in URLs, then change them to see if information beyond the scope of the account is provided, such as other users' information. [Client]

Not Tested 1.2. Look for other names, IDs or fields that can be changed to provide unauthorized access to information or resources [Browser]

2. Authenticated sessions time out (V2.3) [Client]

Not Tested 2.1. Verify that session will time out

3. Error messages (V6.1) [Client, asafaweb.com does some checking for this]

Not Tested 3.1. Verify that error messages do not reveal system internals

4. XSS and Injection testing (Multiple ASVS requirements) [Client]

Not Tested 4.1. Look for instances where user input is served to the client without input verification. For example, enter a short script to raise an alert in a user name field when setting up a test account.

5. APIs that can be used for unauthorized data access (more common in moble apps) [Client, Proxy]

Not Tested 5.1. Examine client requests, look for API calls that can be used with other parameters or without authentication to retrieve sensitive information.

6. Brute-force check for login [Proxy]

Present a sequence of invalid passwords for a user account

Not Tested 6.1. Check for rate limiting