Security Assignment: Web application security report

Please read these instructions carefully as they are quite specific.

Overview

This assignment requires you to prepare a report on web application security.

Except for the first section on threat modelling and misuse cases, this should be a **practical** exercise – i.e. report on your **personal** experience with the tools and technologies that you include.

There are no specific length requirements. **Be concise.** It is more important to present the topic clearly and in a practical way. Do not copy and paste anything from the web. Plagiarism is unacceptable and will result in a mark of zero. Cite references appropriately and use your own words if summarising the work of others.

Upload your report to Moodle in **pdf** format using the link provided. A separate dropbox is provided for associated code projects and other related items – please use a single **zip** file for this.

You can choose to use a single web application for all of this, or alternatively you can write small standalone "proof-of-concept" programs for each part of this report, or you can mix as you see fit. There are no restrictions on web server or framework, programming language, libraries, or operating system. Some of the suggestions below refer to the Hapi.js framework, but you may use an alternative (e.g. Play) if you prefer.

Contents of report

Your report should have the following main sections

1. **Threat modelling using misuse cases**

   This involves modelling threats using misuse cases and using this to add security requirements. More details are provided in section 1 below.

2. **Authentication and encryption**

   Explore web application authentication. Step-by-step instructions are provided in section 2 on the following pages.

3. **Penetration testing**

   Demonstrate how to exploit a selection of web application vulnerabilities and weaknesses. Step-by-step instructions are provided in section 3 on the following pages.

4. **Optional extras (for bonus marks)**

   You may wish to explore other aspects of securing your web application. For example you could look into how to encrypt stored data using the Node.js *crypto* module. This module is a wrapper for a selection of OpenSSL functions. This is entirely optional and not required for full marks.

5. **Conclusion**

   Briefly summarise what you see as the strengths and potential weaknesses of your web application from the perspective of security.

1. Threat modelling using misuse cases

Several software vulnerabilities are caused by poor specification of requirements, poor design and/or poor testing rather than specific programming flaws. It is important to consider security at all stages of the software development lifecycle.

For the purpose of this assignment, you are asked to specify and analyse requirements for a software system of your choice. Besides specifying basic functionality, you should concentrate on security requirements and use threat modelling to help put together those requirements. In particular you should specify what are known as **misuse cases** (also called abuse cases).

Steps for this section of your report

(a) Specify requirements for a software system of your choice. Make sure that it is simple enough to be specified fully in one or two pages of requirements, but sophisticated enough to be interesting – i.e. there should be at least two different types of user (actors); there should be a variety of functions, some of which are restricted to certain users; some of the functions should be multi-step (for example, a transaction might combine an order function with a payment function). You may think in terms of an application you have developed yourself (perhaps enhanced a bit) or something completely different.

(b) Define a selection of (normal) use cases[1]. This requires you to define the (benign) actors and how they interact with the system

(c) Define a selection of misuse cases. This will likely involve some new actors. You might wish to consider both external and internal attackers.

(d) Augment the requirements and use cases to mitigate the threats posed by the misuse cases

Your report should provide a listing and concise explanation of initial requirements (step 1) as well as augmented requirements with justification (step 4). The (mis)use cases in steps 2, 3 & 4 should be described both in text and using diagrams.

2. Authentication and encryption

Steps for this section of your report

(a) Write or adapt a small web application (e.g. using Hapi.js or other framework such as Play) that processes data from a form using a HTTP **POST** request. It can for example be a combination of a registration page for new user accounts and a login page for authentication. Include some persistence of data on the web server (e.g. using a database). You may reuse code from another module if you wish (e.g. the Donation sample app or Spacebook from last semester). Make sure that at least one form field has the `type="password"` attribute set to protect password entry.

(b) Test the web application in (a) above.

(c) Use a proxy such as that included with Burp Suite to intercept the request from the form to the web server and observe that the password is sent in the clear.

(d) Repeat part (a) using a HTTP **GET** request instead of POST. *Comment briefly in your report on what is different and suggest which is better from a security perspective, and why.*

(e) **TLS.** Deploy TLS on your web application based on a key pair that you generated yourself. Verify it in a browser and provide screenshots to show the certificate. Explain any browser warning that you get.

[1] For a good concise description of Use Cases, see http://www.computing.dcu.ie/~renaat/gdip/umldist-chap3.html (extract from *UML Distilled*, by Martin Fowler)

(f) **Password hashing & salting**. Recommended best practice for password storage requires passwords to be hashed and salted. You can implement this using a library such as **bcrypt**.

(g) Demonstrate use of **another authentication or authorisation technique** (e.g. OAuth).

3. Penetration Testing

Steps for this section of your report

(a) Demonstrate how a proxy (e.g. Burp Suite) can carry out a "Meet-in-the-middle attack" on a TLS connection. This can be based on connecting to any HTTPS website, not necessarily your own.

(b) Show how a stolen cookie can be used to bypass authentication and take over someone's identity.

(c) For demonstration purposes, make your web application vulnerable to **Cross Site Scripting** in two different ways (i) injection of a script fragment in form data; (ii) injection of a script fragment using the URL. Provide some input data to show how these vulnerabilities can be exploited.

(d) HTML forms offer some basic protections, but these are useless in thwarting a serious attacker. Some examples:

- You can force a user to enter a maximum of, say, 8 characters (e.g. for a student ID) by using `<input type="text" maxlength="8">`.
- You can make pre-filled data difficult for the user to change by using the *readonly* attribute; e.g. `<input type="text" name="country" value="Ireland" readonly="true">`.
- Another option for pre-filled data is to hide it altogether using the "hidden" input type `<input type="hidden" name="country" value="Ireland">`

Demonstrate use of some of these basic form restrictions and *show (e.g. using Burp Suite) why they offer no real protection against attack.*

(e) Show using examples how the framework/tools that you are using can provide filtering and sanitisation of input and output to enhance security. Use at least two different form fields with different and non-trivial requirements – e.g. email address, date of birth, 8-digit student ID, car registration number, …