Web application security

1. Contents

2. Introduction

The aim of this exercise is to give you some exposure to web application security. You will work in groups of 4 and you will learn about:

1. The range of possible vulnerabilities that can arise when using web technologies,
2. How to develop websites that are more secure,
3. How to use tools and techniques to investigate and test common security weaknesses, this is known as penetration testing.

You will accomplish this by developing your own website. In fact, you will develop two versions of your website. The first version will be vulnerable to a series of attacks and the second version will not. You will create a short video to demonstrate both versions of your application, showing both the vulnerability and the fix. You will also write a report summarising your findings. There may be vulnerabilities that cannot be mitigated using development techniques alone, the solution might require platform configuration changes or software upgrades, for example. You should report on these in the same way making it clear what your recommendations are, even if you have not been able to implement a solution. If you can implement a solution then you should do so.

3. Building the website

Prerequisites: what you need to know

Most tutorials on website security assume that the student knows something about web development. This will not necessarily be the case for all of you, some of you will have had the opportunity to learn about building a website and some will not. The development knowledge you need for the type of website we're interested is not extensive. You will need to know:

- The 3-tier structure of a web application,
- How the HTTP protocol works,

- How to use HTML to create web page with a form,
- How a client, such as a browser, interacts with a web server,
- How to write a program on a server to process the information in the form,
- How to store information in, and retrieve information from, a database.

At the most basic level, these things are straightforward. You will be provided with a small website template that you can use as a starting point. Furthermore, there will be optional workshops over the first couple of days covering the fundamentals for those that would like them. More information about web development is provided in the references at the end of the document. Your websites will not be marked on the quality of the design (look and feel) and you should only need to implement a simple version of the features with limited business logic. There will be no need for extensive validation, for example, except where the validation relates to the security principle you wish to demonstrate. Likewise, there will be no need to have a complex database design, a small number of tables with just enough data to support your application will be sufficient.

Technology constraints

You should use HTML5 with JavaScript for the client-related features of your application. You should use PHP to implement the server-side logic coupled with a MySQL database at the back end. The choice of PHP is deliberate: on the one hand, it is probably the programming language most widely used in websites today[1] and on the other, it offers some interesting vulnerabilities to explore. Thus, if you are ever in a situation where you are required to assess the security of a web application, there will be a good chance that some or all of the server-side code is written in PHP and you will have some knowledge of the likely vulnerabilities. You may need to use older versions of PHP or MySQL in order to demonstrate certain vulnerabilities. There may be other complementary technologies that you wish to use in order to demonstrate vulnerabilities.

Website testing and deployment

You must deploy your secure website to the Azure platform. A special group account will be set up as soon as possible. Do not deploy the insecure version of your website to Azure or to any other hosting platform. You can limit the risk to your own computer by running vulnerable application and associated testing and scanning tools inside a virtual host. From the outset of this exercise, you will need to run one or more vulnerable virtual hosts. It's a good idea to set up your testing tools in another virtual host, for example: you might choose to test from a virtual machine running Kali Linux [1].

4. Specification for the website

The vulnerable and secure versions of your website should have as many vulnerabilities and fixes as you are able to demonstrate. Marks will be awarded for each feature achieved and will take account of the quality and completeness of the design, implementation and testing process.

[1] https://w3techs.com/technologies/overview/programming_language/all

It’s fairly unusual to write a specification for a website that includes information about how it should be implemented and a description of the vulnerabilities that must be present. Here are two suggestions, you may choose to either:

1. Use the design features described in Appendix A – Website Features as a starting point; these are taken from the Google Gruyere [2] which is a simple micro-blogging site. The most straightforward approach is probably to follow the Google Gruyere tutorial so that you become familiar with it and to study the feature list at the same time. Note: The Gruyere application does not use a database but you must.
2. Invent your own vulnerable app. You can find out more by reviewing other website security teaching applications. You could start with one of these: WebGoat [3], Damn Vulnerable Web Application [4] and the HacMe Casino [5]. There are also relevant courses on lynda.com [6], [7] which include information about testing toolsets and websites to practice on.

5. Penetration Testing

Penetration testing is a type of offensive security testing used to validate and verify the effectiveness of the application's security controls and to identify potential weaknesses. Software tools are used to automate common tests and to exploit weaknesses. You will need to use some of these tools from the outset of this exercise so you should aim to familiarise yourself with what’s available as soon as you can. As a starting point, there are courses available in the UCL Lynda catalogue [6], [7], and several books in the UCL Safari Catalogue, including [8], [9]. One of the first things you will learn to do is to intercept the messages or packets between a client and a web server. You might do this simply to monitor what is being passed to-and-fro or you may intercept the packet and change the content before it reaches the server, the latter is known as a man-in-the middle (MITM) attack. You may use whatever tools you believe to be appropriate but you are required to show that you have used some of the Python or Ruby penetration testing tools. The best projects will use these, will automate tests and may extend the tools if it is relevant.

6. Deliverables

The deliverables are a video of a demonstration of your working systems, a design report and your source code. The video demonstration should be about 8-10 minutes long with voice-over narration. You can use free screen recorder tools such as Jing or CamStudio to make your video. The video must show explicitly the each vulnerability listed in your report and a description of how it is mitigated in your system. You must go through each vulnerability in the order in which it is listed in your report. The narration must refer to the vulnerability and preferably the video should flash back to the visual list. Videos should be uploaded to Youtube and made unlisted. You need to submit your design report (a single pdf file only) which should contain the link to your video.

The report should contain:

1. A URL for your Youtube video
2. A brief description of your application including its architecture, explaining how it aligns with the Model, View, Controller pattern.

3. A description, of your test environment with a diagram. The software configuration of the clients and servers should be itemised. It should be clear what configuration contained the vulnerabilities and, if it was different, the configuration of the more secure server.
4. A review of your test plan explaining the strategy you used. Make it clear how the testing can be repeated, did you automate the testing?
5. A description of the vulnerabilities in your insecure server and associated fixes in the secure web application.
6. A list, with brief description, of the tools you used. Make it very clear if you have written your own tools for, say, testing or log analysis.

As well as the report, you should submit an archive containing your source code files (HTML, CSS, JavaScript, PHP etc.). **Please only submit source code**, no binary data files or database dumps. Please put your group number in the zip file name. Alternatively, you may submit a link to a Github repository.

Assessment

Your project work will be marked against the rubric visible at the upload link. Marks for your project work will be awarded first for the vulnerabilities explored and fixed, and second, for evidence of the process you followed. All members of a group will receive the same mark.

7. Going Further

If you would like to use these skills and learn more about security, you might like to join UCL's Computer Science Hacking Seminar. The UCL team competes regularly in Capture The Flag (CTF) competitions. Alternatively, consider becoming a member of UCLU's TechSoc; last year TechSoc hosted talks from companies like Netcraft, MWR Infosecurity and KPMG. TechSoc is organised and run by students like you.

8. References

[1] "Kali Linux: Our Most Advanced Penetration Testing Distribution, Ever.," 05-Feb-2015. [Online]. Available: https://www.kali.org/. [Accessed: 08-Jan-2017].

[2] "Web Application Exploits and Defenses." [Online]. Available: http://google-gruyere.appspot.com/. [Accessed: 08-Jan-2017].

[3] OWASP, "OWASP WebGoat Project - OWASP." [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project. [Accessed: 05-Jan-2017].

[4] "DVWA - Damn Vulnerable Web Application." [Online]. Available: http://www.dvwa.co.uk/. [Accessed: 05-Jan-2017].

[5] "Hacme Casino v1.0 | McAfee Free Tools." [Online]. Available: http://www.mcafee.com/us/downloads/free-tools/hacme-casino.aspx. [Accessed: 05-Jan-2017].

[6] C. Martorella, "Learning Python Web Penetration Testing," *Lynda.com - from LinkedIn*. [Online]. Available: https://www.lynda.com/Python-tutorials/Learning-Python-Web-Penetration-Testing/521198-2.html. [Accessed: 05-Jan-2017].

[7] "Ethical Hacking: Website and Web Application Testing," *Lynda.com - from LinkedIn*. [Online]. Available: https://www.lynda.com/Linux-tutorials/Ethical-Hacking-Website-Web-Application-Testing/512727-2.html. [Accessed: 08-Jan-2017].

[8] C. Duffy *et al.*, *Python: Penetration Testing for Developers*. Packt Publishing, 2016.

[9] C. Buchanan, T. Ip, A. Mabbitt, B. May, and D. Mound, *Python Web Penetration Testing Cookbook*. Packt Publishing, 2015.

Appendix A – Website Features

The application is modelled on Google Gruyere, you may design your own application if you wish. The set of features is not complete but represents a subset that you can start with. The Gruyere application refers to snippets. A snippet is simply text that may contain, in this case, some basic HTML formatting. Think of it as a small blog post.

Sign-in / sign-up requirements

1	A user must set a login name and password using an HTML form.
2	A user must sign-in using their login name and password.

Profile related requirements

3	A user may set or modify a user name. The user name will be shown in the menu bar of the application.
4	A user may set or modify the following attributes: password, URL for an icon, URL for a homepage, profile colour (text description), a private snippet.

Extended profile related requirements

5	An administrator can edit the profile attributes of any user, as listed above.
6	In addition, an administrator can edit the permissions associated with the user. These are: • Whether the user is an administrator or not. • Whether the user can create snippets or not.

Snippet management

7	A user can create a snippet.
8	A user can delete one of their own snippets.

Site Home page

9	If the user is not logged in, the Home page contains links to the sign-in and sign-up functions. The Home page also contains a list of all the users. A list containing information about all users. The following is displayed for each user: • The user’s user name, • The last snippet created by the user, • A link to a page displaying all the user’s snippets.
10	When a user has logged in, they can view the following additional items of information on the site home page: • A link to the user’s homepage. The default value for the user’s home page is a link to the website login page with the user’s username and password. For example: <website URL>/login?uid=benbear&pw=bear

File upload

11	A user may can upload a file to the website from the local filesystem.
12	A user may view files that have been uploaded. Assuming that the user uploads a file called theFile.html, the file can be viewed at the URL: <website>/userid/theFile.html