Design of Software Systems

Project Assignment

1 Introduction

For the project of the *Design of Software Systems* course, you will be extending and refactoring OWASP ZAP[1], an open source security tool. ZAP can be used as an intercepting proxy, to study the contents of HTTP requests and responses. ZAP is also capable of automated scanning/spidering of a website.

You will be given 6 small assignments to complete within the codebase of the ZAP project. For each of these assignments, you are expected to apply the design principles and patterns that have been covered during the lectures.

DISCLAIMER *Running ZAP scans on a website without explicit permission can be considered to be malicious activity, for which you may be held responsible. If you want to use these automated spidering/scanning features, use them responsibly against training applications*[2]

2 Project Organization and Timeline

In order to allow you enough flexibility in planning the project, we have divided the entire project into 6 iterations. The assignments will gradually increase in difficulty, allowing you to apply what you have learned during the lectures. The first two iterations will ask you to extend the codebase of ZAP. The next two iterations will require you to refactor and improve existing code within the project. During the final two iterations, you will implement an extension within the ZAP project. Table 1 shows the start and end dates for each iteration.

We expect you to team up into groups of 4 or 5 students. The group composition is up to you, but you must let us know which group you are in before October 28, by sending an e-mail to **oss@cs.kuleuven.be**. You will remain in the same group for the entire

[1] `https://github.com/zaproxy/zaproxy`

[2] You can easily download and set up the Damn Vulnerable Web Application for this purpose (`http://www.dvwa.co.uk/`)

Date	Event
21 Oct 2016	Release of iteration 1 and 2
11 Nov 2016	Intermediate submission iteration 1 and 2
15 Nov 2016	Release of iteration 3 and 4
18 Nov 2016	Release of iteration 5
5 Dec 2016	Intermediate submission iteration 3 and 4
5 Dec 2016	Release of iteration 6
20 Dec 2016	Final submission (source and report)
During the exam	Questions about the project

Table 1: Project timeline

project. If you have trouble finding a group, let us know as soon as possible by mailing to the same address.

After you have sent your group composition, your group will be assigned a supervisor. This supervisor is part of the course staff and is there to help you with the project. You are allowed to schedule a 1 hour meeting with your supervisor at most **once per week**. When you schedule a meeting, make sure you prepare **concrete questions** and have **appropriate design artefacts** available to support your questions (e.g. you can show class diagrams for two alternative designs that you propose, and the supervisor can help you decide which design is more appropriate). Note that the supervisor will not do the work for you: if you give insufficient input, the supervisor will not be able to help you much.

The project timeline contains intermediate submission deadlines, on which you will submit a current snapshot of your source code (see the guidelines below). These moments are intended as intermediate feedback moments, not as formal evaluation points. We advise you to use these intermediate deadlines as a guideline regarding the division of the workload of the project, but you are free to complete your assignmetns at any time during the course of the project.

For the final evaluation of the project, you will have to submit your source code, as well as a report (see below for guidelines). The report should reflect the design decisions you made throughout the iterations, and should refer to the concepts covered during the lectures. Consider the report as the documentation of the extensions you made. You are writing the documentation for your future self, since you can use it at the exam, when we will ask you a few questions about your project.

Report Submission Guidelines

The report is should include the following sections, with their respective page limits:

- **Design decisions:** Explain the major decisions that have driven your design. Why did you use a certain design pattern. Which are potential alternatives? ... (max 2 pages)
- **Strenghts of the design:** Explain why your design is good. Argument your

decisions using the GRASP principles, like we have done throughout the course (max 1 page)

- **Weaknesses of the design:** Critically evaluate your own design. Where are the weaknesses? Argument your decisions using the GRASP principles, like we have done throughout the course (max 1 page)
- **Future improvements:** Describe how you would address the weaknesses in your design. For each of these improvements, sketch the solution and give an estimate of the amount of work (max 1 page)
- **Design Diagrams:** Class diagrams and interaction diagrams of the extensions you created. The minimum font size for the diagrams is 10pt, and they should include relevant methods, and can leave out irrelevant methods such as getters and setters (no limit)
- **Overview pyramid and test coverage:** An overview pyramid of the module that you refactored in iteration 5. Include a pyramid from before the refactoring, and one from after the refactoring. You should also provide an overview of the test coverage of your unit tests on your own code (max 1 page)

Code Submission Guidelines

Because of the size of the project, we only require you to submit a diff of your codebase against the original ZAP repository you cloned from GitHub. You submit your code by sending a mail to **oss@cs.kuleuven.be**. Please mention your group number in the subject line and attach your diff in a ZIP archive to this mail.

3 Getting Started

3.1 Introducing ZAP

As stated, this year's assignment will be centered around the *OWASP Zed Attack Proxy* ('ZAP'), one of the world's most popular free security tools. It is actively maintained by hundreds of international volunteers. ZAP can be used to automatically find security vulnerabilities in web applications during the development and testing phase. It is also a great tool for pentesters to use for manual security testing.

A good starting place for finding information on ZAP is its OWASP web page:

`https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project`

Here, you can download a copy of ZAP and find links to the *Getting Started* guide and various video tutorials. However, rather than downloading one of the installers –since you will be making changes to the source code– it is preferable to clone the ZAP *git* repository into a suitable working folder using:

```
git clone https://github.com/zaproxy/zaproxy.git
```

You can then open the project locally using *Eclipse* or any other integrated development environment of choice. Just make sure that you configure the Java Compiler Compliance level for the project to 1.7 or 1.8.

3.2 Running for the first time

Now try running the project: run as a Java Application, the main class for the project is `org.zaproxy.zap.ZAP`[3]. After agreeing to the ZAP Terms & Conditions (this needs to be done only once), an initial 'splash screen' appears and a dialog window asking if you want to persist the ZAP session; then the application's main window opens.

3.3 Intercepting web traffic using ZAP

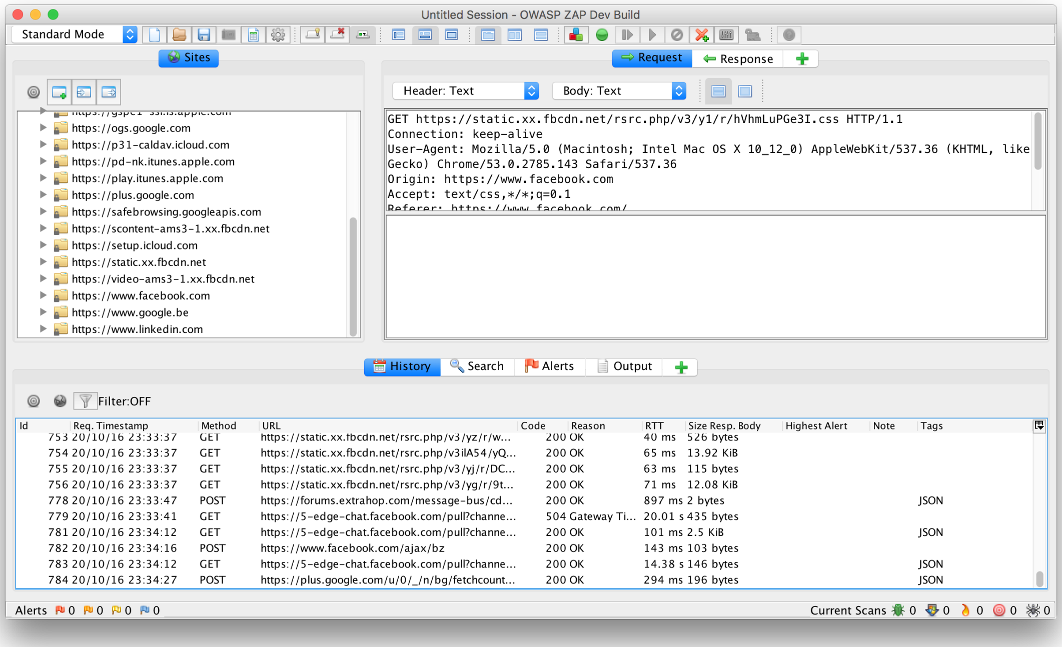
As mentioned, ZAP can function as an intercepting proxy server. To make use of this functionality, you need to configure your browser to connect to ZAP as a proxy server.[4] ZAP by default listens on port 8080, so the proxy server setting is `localhost:8080`, both for http and https traffic. If needed, the port ZAP listens on can be changed in *Options / Local Proxy.*

[3] If, at this time you are seeing an *Exception in thread "main" java.lang.UnsupportedClassVersionError*, you may need to either install a correct version of the JDK or switch the Java Compiler Compliance level from 1.7 to 1.8 or vice versa

[4] Depending on the operating system of your machine and the browser used, this may be a machine-wide setting, meaning that al HTTP and HTTPS traffic will be routed throug ZAP, once you configure ZAP as a proxy server. If you prefer to only route the traffic from one browser through the ZAP proxy, you should probably use FireFox as a browser, which allows configuring a proxy for this browser only.

If you want to intercept HTTPS traffic, ZAP will act as man-in-the-middle. It will generate certificates on-the-fly for any web site you want to visit. However, for your browser not to be alarmed and to block your requests, you will need to save ZAP's Root Certificate to a file and import it in your operating system's Root CA store. The Root Certificate can be found in *Options / Dynamic SSL Certificates*.

When you have successfully configured your operating system or your web browser (FireFox) to use ZAP as a proxy server and when you have successfully imported the ZAP root CA certificate as a trusted Root CA certificate in your machine's (or browser's) certificate store, you can start browsing the web. You will notice that the URLs visited are now recorded in the ZAP main window, as shown in figure 1. The *Request* and *Response* tabs allow you to inspect the HTTP request and response contents.



- https://gspe1-ssl.ls.apple.com
- https://ogs.google.com
- https://p31-caldav.icloud.com
- https://pd-nk.itunes.apple.com
- https://play.itunes.apple.com
- https://plus.google.com
- https://safebrowsing.googleapis.com
- https://scontent-ams3-1.xx.fbcdn.net
- https://setup.icloud.com
- https://static.xx.fbcdn.net
- https://video-ams3-1.xx.fbcdn.net
- https://www.facebook.com
- https://www.google.be
- https://www.linkedin.com

```
GET https://static.xx.fbcdn.net/rsrc.php/v3/y1/r/hVhmLuPGe3I.css HTTP/1.1
Connection: keep-alive
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_0) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/53.0.2785.143 Safari/537.36
Origin: https://www.facebook.com
Accept: text/css,*/*;q=0.1
Referer: https://www.facebook.com/
```

Id	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags
753	20/10/16 23:33:37	GET	https://static.xx.fbcdn.net/rsrc.php/v3/yz/r/w...	200	OK	40 ms	526 bytes			
754	20/10/16 23:33:37	GET	https://static.xx.fbcdn.net/rsrc.php/v3ilA54/yQ...	200	OK	65 ms	13.92 KiB			
755	20/10/16 23:33:37	GET	https://static.xx.fbcdn.net/rsrc.php/v3/yj/r/DC...	200	OK	63 ms	115 bytes			
756	20/10/16 23:33:37	GET	https://static.xx.fbcdn.net/rsrc.php/v3/yg/r/9t...	200	OK	71 ms	12.08 KiB			
778	20/10/16 23:33:47	POST	https://forums.extrahop.com/message-bus/cd...	200	OK	897 ms	2 bytes			JSON
779	20/10/16 23:33:41	GET	https://5-edge-chat.facebook.com/pull?channe...	504	Gateway Ti...	20.01 s	435 bytes			
781	20/10/16 23:34:12	GET	https://5-edge-chat.facebook.com/pull?channe...	200	OK	101 ms	2.5 KiB			JSON
782	20/10/16 23:34:16	POST	https://www.facebook.com/ajax/bz	200	OK	143 ms	103 bytes			
783	20/10/16 23:34:12	GET	https://5-edge-chat.facebook.com/pull?channe...	200	OK	14.38 s	146 bytes			JSON
784	20/10/16 23:34:27	POST	https://plus.google.com/u/0/_/n/bg/fetchcount...	200	OK	294 ms	196 bytes			JSON

Figure 1: ZAP successfully intercepting https browsing sessions

4 Iteration 1: Flipping images

In this first iteration, you will need to get familiar with the ZAP source code and the data flow through its software design, while ZAP is intercepting http(s) sessions. The goal of this iteration is to allow a user to browse any particular web site, but all of the images on the web pages should appear upside down[5] in the web browser.

You do not need to implement the rotation or mirroring of images yourself. It is fine to use whatever library you find convenient for this functionality.

[5] Putting images upside down, can be achieved either by rotating each image with 180 degrees, or by mirroring each image horizontally. Since some web sites (such as Google) compose their images of several sub-images, mirroring will give a more spectacular end result than rotating. However, for this assignment both approaches are considered fine.

5 Iteration 2 - Updated

In the second iteration of the project, you further extend the image filtering capabilities of ZAP. The first extension you created was the one capable of flipping the images sent to the user. In this iteration you will have to add more two capabilities:

- Image size based enhancing
- Water mark

We want to be able to easily add more capabilities to manipulate images. For that, you can define a configuration file, which specifies the order of each image capability and if the capability is active or not. If both capabilities are active, then, both have to be applied to the image.

Image size based enhancing In this filter you will change the images based on their size in kilobytes (kB). If the image is greater than 100 kB, you will transform the image to a greyscale image. On the other hand, if the image is smaller than 100 kB, you will simply keep the image as it is.

Water mark In this filter you will add a water mark to the images. You can choose wherever watermark you find interesting.

6 Iteration 3: Content Filtering

In this iteration, you are expected to design and implement a content filter feature for ZAP Proxy. A content filter inspects network traffic, and filters out inappropriate content. Such functionality is often used in corporate environments to prevent non-work related use of the Internet, or at home to keep children’s eyes away from unwanted content.

Concretely, your content filter needs to be able to process HTML pages, and determine if they are appropriate or not. Appropriate pages are allowed to pass through unmodified. Inappropriate pages should be replaced with a placeholder page. The placeholder page should tell the user that the page was blocked, along with a list of tags associated with the inappropriate content.

When starting, your content filter reads a list of inappropriate strings, along with their weights, and a set of tags that identify the type of content. An example of such a list is given below. Your solution should include a well-designed parser for this format. The parser should ignore malformed lines.

To identify content as inappropriate, you need to implement a simple classification algorithm, which uses the list of inappropriate strings. This algorithm looks for these strings, and simply calculates a sum of the weights of all strings that occur in the page. If that sum exceeds the pre-set threshold, the page is deemed inappropriate.

Take for example a news article about the presidential election in the US. It mentions the string *’Trump’* a lot, but none of the other inappropriate strings. Its score would be 4, which is below the threshold (10 in this case), so the content is considered appropriate.

Now take a page from a forum of Trump supporters, which contains the strings *’Trump’*, *’Nigger’*, *’deport Mexicans’*. Based on our common-sense content filter, this page scores a 24, which is well above our threshold of 10. This content is considered inappropriate, and the placeholder will tell the user it’s because of *racist* and *fearmongering* content.

```
# THIS IS A COMMENT AND CAN BE IGNORED
# FORMAT: threshold #VALUE#
threshold 10

# FORMAT: #inappropriate string#;#weight#;#TAGS SEPARATED BY COMMA#
trump;4;fearmongering,mysogenist,bigotry
nigger;10;racist
deport mexicans;10;racist,fearmongering
dick;2;sexual
```

Please do not over-design your solution. It is sufficient to support only the requested functionality. However, your design should be easily modifiable to support the following extensions:

- Other formats of the list of inappropriate strings
- Apply a similar filtering mechanism to other types of content

- Plug in other content classification algorithms

7 Iteration 4: Content Reporting

In this iteration you will create an extension that adds new types of reports to ZAP Proxy. To generate a report in ZAP Proxy, first you click on the **Report** menu item and select either **Generate HTML Report... Generate XML Report...**, **Generate Markdown Report...**. Your extension will add information to the report generated by ZAP Proxy, in any format (HTML, XML, Markdown). You should try to identify how ZAP Proxy is supposed to be extended, instead of simply trying to hack your solution into ZAP Proxy code.

We are interested in the distribution of images used on each website visited in a session. Your report will create statistics per website visited showing the following statistics:

- **Image type statistics**
  - percentage of images using a particular image type (jpeg, png, gif)
- **File size statistics**
  - minimum image size (in bytes) + URL of the smallest image
  - average image size (in bytes)
  - median image size (in bytes)
  - maximum image size (in bytes) + URL of the largest image
- **Image width statistics**
  - minimum image width (in pixels) + URL of the narrowest image
  - average image width (in pixels)
  - median image width (in pixels)
  - maximum image width (in pixels) + URL of the widest image
- **Image height statistics**
  - minimum image height (in pixels) + URL of the shortest image
  - average image height (in pixels)
  - median image height (in pixels)
  - maximum image height (in pixels) + URL of the longest image

You have to be careful with your data structures in order to guarantee they are not corrupted by any other threads from ZAP Proxy, both during data collection and when (concurrently) generating your report.

8 Iteration 5: Refactoring

In this iteration you will refactor some parts of ZAP Proxy. According to Martin Fowler, refactoring is:

“Refactoring (noun) a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior.”

“Refactoring (verb): to restructure software by applying a series of refactorings without changing its observable behavior.”

This extension is made with ZAP Proxy commit **cde0f32** in mind. A crucial aspect of ZAP Proxy is the the capacity to work with extensions. One of the main classes involved in handling extensions is the **ExtensionLoader** class.

The **ExtensionLoader** has many methods related to loading, unloading, and managing extensions. It is also responsible for notifying listeners about all types of events that may effect extensions, sessions, add-ons, etc, what can be seen in method as **ExtensionLoader:addOnInstalled(AddOn addOn)**. Because of this many responsibilities, the **ExtensionLoader** class is quite massive, containing more than 1300 lines of code.

8.1 Task 1

Your first task is to come up with a design and implementation that will simplify the **ExtensionLoader**, making it simpler and easier to maintain. You will have the GRASP patterns in mind when refactoring the **ExtensionLoader** class, and you will use refactoring techniques as the **Extract Class**, **Componentization**, etc.

In your design you will have to consider what are possible changes that will have the highest impact in reducing the complexity of the **ExtensionLoader** class. As an example, you could start by thinking about the **ExtensionLoader::startLifeCycle(Extension ext)** method. Do you think such responsibility belongs to the ExtensionLoader class?

8.2 Task 2

Your second task is to find another class in ZAP Proxy that is a good candidate class for refactoring. You will refactor the class you found, aiming at making it simpler while preserving its behaviour.