Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Webapp exploit detection through semi-automated black-box scanning

Author:
Abraão Pacheco Dos Santos Peres Mota

Supervisor:
Dr. Sergio Maffeis

Second Marker:
Dr. ??

January 10, 2018

Contents

Chapter 1

Introduction

1.1 Motivation

In recent years, use of internet applications has skyrocketed across the world. This is exarcebated by the ubiquity and sheer number of devices that are now connected to the internet. The users of these devices place a great deal of trust in the applications and websites they use to power the activities they engage in. These play an ever increasingly influential role in people's lives - as an example, in a not so distant past, people would have to travel to a physical bank branch to deal with account matters or execute transactions. Though physical presence can still be required nowadays, a majority of the population will now take advantage of online banking in their day to day lives, often even from the comfort of the phone in their pocket. This has obvious time and efficiency advantages, and benefit many who use this today. This was not an overnight phenomenon however; online banking initially faced heavy customer reluctance, enough to warrant studies on the cause of this [3]. This is understandable, given that everyone holds their financial situation as a very sensitive part of their lives. Banking is not a solitary example however; with the pervasive use of the internet, its users have gradually become less apprehensive about surrendering important details over a web connection, such as their phone numbers, addresses or medical history. All of this builds up to a massive responsibility placed on the shoulders of web application developers today; their products are expected to uphold a high standard of security, which oftentimes is hard to reach and maintain.

Despite this, it is all too common to hear about web applications that suffer from severe cyber attacks. In some of the most debilitating hacks we have heard of in recent years, it is often the case that they were a result of simple vulnerability mitigation measures not being taken. A vulnerability in this context can be illustrated as an unlocked window into a house - it may not be immediately clear that the house owner has overlooked this security aspect, but if a burglar manages to work out that this is the case, they can maliciously *exploit* this vulnerability in the house, and use that as an entry point to steal all the valuable contents within. Properly closing and locking all the windows in the house would be an obvious mitigation to this, but this is only one of the potential (ingenious) ways for a burglar to make his way into the house. The same principle can be applied to websites; it is important to cover as many bases as possible to prevent a potential information or content leak to malicious users. Some vulnerabilities may be harder to detect than others, and it is unlikely that *all* the possible vulnerabilities will be covered, but any efforts towards can only work in favour of the website developer.

Sadly, due to the immaturity of the web development industry and how quickly technologies emerge in the field, web security is an often overlooked aspect of development. The lack of security as a fundamental tenet for development is also due to a gap in developer education, and a high entry barrier to understand and mitigate potential security vulnerabilities. Though this view is slowly changing, web security is not treated as an important focus for new web developers to understand as part of many online and university courses, so many will get by, even work in professional development roles without having ascertained basic security principles. Common vulnerability mitigation measures are also hidden in the inner workings of many frameworks developers use and become accustomed to. For example, using *Anti CSRF* (Cross Site Request Forgery) tokens in web

forms to prevent action hijacking has become commonplace in web frameworks, and is a feature that is often activated by default [1, 2, 4]. It is very often the case that features like this will be used without knowledge of what they do and how they work (in my own experience, I had been venturing in web development for years before realising that feature existed). However, in the case that the developer changes to use another framework without *secure by default* features, or decides to write an application from scratch, the burden of creating a secure application lies with them even further. These default features become like 'training wheels' for some uneducated developers and without them, these creators will be left without a clue as to how to mitigate vulnerabilities, let alone know that these exist altogether. Experienced web developers are less likely to leave the 'windows' of their website unlocked, but it nevertheless makes sense to install security alarms to prevent both obvious and more subtle security risks. If these security systems can automatically work in the background it is ideal, but like a home security alarm, it is no good if no-one intervenes upon detection of a problem. This begs the question; what is the feasibility of creating a tool that can work as an aide to a web developer in detecting and preventing vulnerabilities in a website?

1.2 Objectives

The question raised above effectively highlights the overarching problem this project aims to tackle - improving security for web applications. The goal is to do this through a pragmatic approach; the final objective of this project is to construct a tool which can diagnose vulnerabilities on a target website and goes a step further and attempts to use said vulnerabilities to explore potential exploits on the target.

My initial proposal in solving this encompasses writing a browser extension to analyse the target website, and applying a wide variety of scans and techniques to detect potential security pitfalls the website may have. A browser extension is an appropriate approach to this as it is a lightweight application running with elevated privileges on the user's browser, giving it the appropriate clearance level to run a variety of automated scans on the user's behalf.

A tool of this kind has 2 immediately clear use cases. The first would be to give this tool to a website owner or developer and create it in such a way so that it gives clear and concise suggestions to quantifiably improve the security level of the target website - for example, if an SQL injection has been detected, show the user where on the website this vulnerability can be found, and make effective suggestions as to how to mitigate this (in an SQLi, it may be done by sanitizing user input). The tool could analyse a range of potential vulnerabilities and generate a quantifiable rating for the website in order to give feedback to the developer on where the website needs improving or immediate work. The slightly alternative use case of this tool provides a more in-depth scan per potential vulnerability, and would be better suited for use by a penetration tester, or an otherwise knowledgeable web security expert. In this use case, the tool would work in a similar fashion, albeit with a different final goal of going 'all the way' by helping the user to detect vulnerabilities and creating exploits using them on the target application. In both potential use cases, the benefits of the tool are twofold - it can be used as an educational stepping stone for developers to further their understanding of security in web applications. It also provides a pragmatic way to improve website security through alternate routes - that of the direct owner improving their website by given suggestions, or the penetration tester showing developers the dangers of ignoring security for their website by exploiting open vulnerabilities.

User driven approach

This goal poses several challenges, the biggest of which are:

1) Breadth of analysis 2) Depth of analysis 3) Accuracy of the scanner

This project aims to explore the latter of the two approaches mentioned. This is the more encompassing of the two possible use cases, with more information produced than just recommendations on how to improve a website. It also provides appropriate scope to allow for an in-depth analysis per potential vulnerability. Given that, this project aims to explore vulnerabilities in as much depth as possible, with less focus on attempting to detect as many vulnerabilities as possible. This choice is made with the evaluation of the tool in mind - as will be discussed later on. False positives (declaring that a website has a vulnerability when in fact it doesn't) are a major

evaluation metric for the project and minimising these through more accurate, in-depth scans will result in a quantifiably better, more usable tool.

1.3 Challenges

Chapter 2

Background

This should form the bulk of the interim report. You should consider that your objective here is to produce a near final version of the background section, as it will appear in your final report. All of this material should be re-usable, so it is worth getting it right at this stage of the project. The details of what to include can be found in the Project Report guidelines.

In writing the Background chapter you must demonstrate your capability of analysis, synthesis and critical judgement. Analysis is shown by explaining how the proposed solution operates in your own words as well as its benefits and consequences. Synthesis is shown through the organisation of your Related Work section and through identifying and generalising common aspects across different solutions. Critical judgement is shown by discussing the limitations of the solutions proposed both in terms of their disadvantages and limits of applicability.

2.1 Analysis

2.2 Related Work

2.3 Challenges

2.4 Limitations

Chapter 3

Evaluation

Project evaluation is very important, so it's important to think now about how you plan to measure success. For example, what functionality do you need to demonstrate? What experiments to you need to undertake and what outcome(s) would constitute success? What benchmarks should you use? How has your project extended the state of the art? How do you measure qualitative aspects, such as ease of use? These are the sort of questions that your project evaluation should address; this section should outline your plan.

Chapter 4

Project Plan

You should explain what needs to be done in order to complete the project and roughly what you expect the timetable to be. Don't forget to include the project write-up (the final report), as this is a major part of the exercise. It's important to identify key milestones and also fall-back positions, in case you run out of time. You should also identify what extensions could be added if time permits. The plan should be complete and should include those parts that you have already addressed (make it clear how far you have progressed at the time of writing). This material will not appear in the final report.

Bibliography

[1] Django Software Foundation. Django project documentation - cross site request forgery protection. https://docs.djangoproject.com/en/2.0/ref/csrf/, 2018.

[2] Rails Guides. Rails guides - csrf countermeasures. http://guides.rubyonrails.org/security.html#csrf-countermeasures, 2018.

[3] Tuire Kuisma, Tommi Laukkanen, and Mika Hiltunen. Mapping the reasons for resistance to internet banking: A means-end approach. *International Journal of Information Management*, 27(2):75 – 85, 2007.

[4] Taylor Otwell. Laravel docs - csrf protection. https://laravel.com/docs/5.5/csrf, 2018.