Analysis Techniques to Identify Web Application Vulnerability that can Cause Security and Privacy Issues

Gabriele Qazolli, 001037593

Computer Science (Cyber Security), University of Greenwich

Faculty of Liberal Art and Sciences, Department of Computer Science

Email: qazolligabriele@gmail.com

Abstract

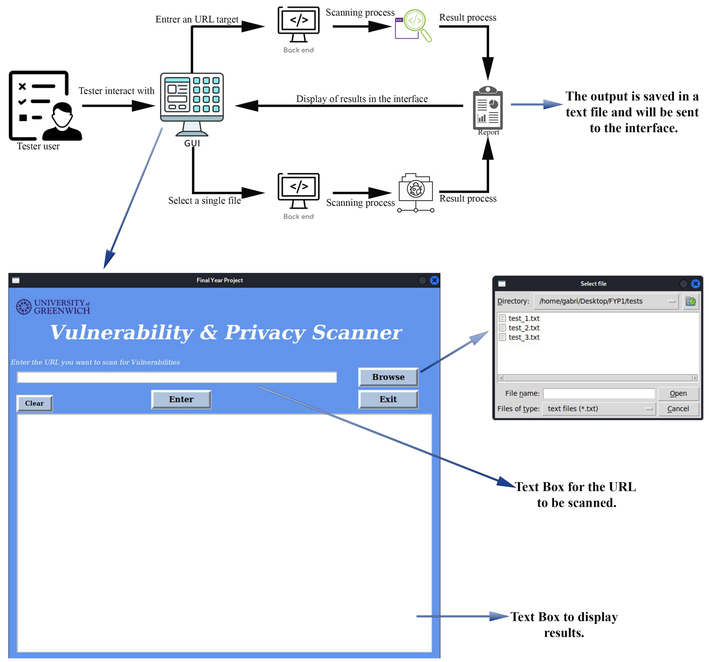
The problem of security of Web Applications is essentially technical and should be solved by spreading a culture of security of Web Applications, which is currently not satisfactory. In fact, many applications on the network are susceptible to attack by malicious individuals precisely because of their "careless" programming; these applications could become more secure with small measures. The objective of this work is to offer an overview of the most common vulnerabilities found in Web Applications and analyse and apply the possible methods and techniques used by scanners for these vulnerabilities that can cause security and privacy issues.

Introduction

Web Applications or web apps became quite popular at the end of the 90s due to the possibility for a client to access application functions using standard web browsers as terminals. In fact, the opportunity to update and evolve one's own application at a reduced cost without being forced to distribute numerous updates to one's customers via physical support has made this solution quite popular for many software producers. Unfortunately, however, recent history allows us to state with certainty that today's society has paid and continues to pay the heavy price of having long underestimated the intrinsic vulnerabilities of the Internet and its protocols.
Since the issue of security in web-based applications is a topic of great interest and impact, in recent years, we are witnessing the emergence and development of methodologies, techniques, and products to support developers and end-users in verifying their level of Web Application security. For this reason, Web Application Vulnerability Scanners were created.

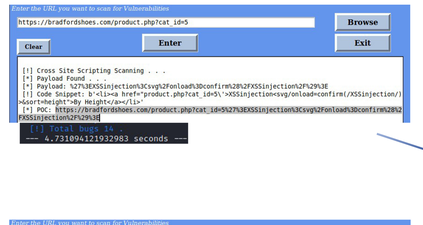
Problem Domain and Proposed Solution

These vulnerabilities aggravate various sections such as database security with personal information about the Client, such as a bank account or payment information stored. When attackers have access to this information, it is extremely difficult to defend it. Technologies such as network firewalls and anti-virus software, offer reasonably secure protection at the host and network levels but not at the application level. When network and host-level access points become relatively safe, public interfaces to Web Applications become the target of attacks. Therefore, the best approach to detect possible vulnerabilities and then fix them is through the use of Web Application Vulnerability Scanners.
There are already a lot of scanners out there, and each one has its strengths and weaknesses that need to be considered when trying to make a good product that can compete in this sector. Black-box vulnerability scanner such as Wapiti, Skipfish and W3af performs well when detecting difficult-to-find vulnerabilities such as stored XSS, blind SQL or XML injection, or blind shell injection. However, one point fundamental is the scanning time. All three scanners have an undetermined run-time that sometimes can take more than half an hour. If a developer needs to know if it is only vulnerable to Cross-Site Scripting, he, unfortunately, has to wait until the end of the process. Moreover, if the software offers a lot of information, it might overwhelm an end-user, even if all of the information is useful. The proposed solution is to create a black box scanner that will scan three specific vulnerabilities in Web Applications in a short time, and will also scan a file that may contain malicious code that can cause security or privacy issues. In addition, a nice and clean interface can help the user better understand the project. An overview of how the scanner works and a view of the interface as soon as it starts up can be found in the images below.

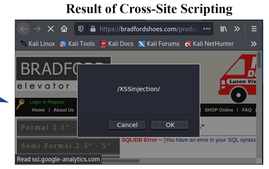


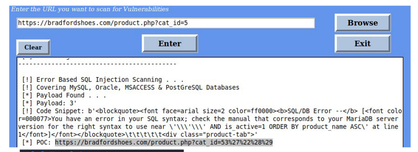
Result

Tests are carried out on URLs for the content of various vulnerabilities such as XXS and SQL injection and on files containing malicious code that may cause security and privacy issues. After input of the target URL, the program was able to identify two types of vulnerability: Cross-Site Scripting and error-based SQL injection. In addition, scanning the file *text3.txt* provided results regarding a possible web page redirection.

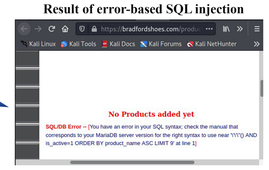


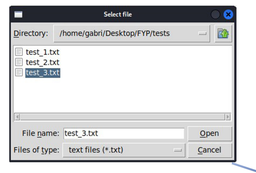
Result of Cross-Site Scripting





Result of error-based SQL injection





Result of file test_3.txt scanned

Discussion

From the results, we can see that the scanner was able to find vulnerabilities with XXS and error-based SQL injection. The scanner returns a POC (Proof of Content) which we can paste into the URL and display the result. In the case of XSS, we obtained an infobox showing us the injected text. And in the case of error-based SQL, we can see that it returns an error giving us essential information such as the column name (product name). Finally, scanning a file with malicious code is detected (redirect) by specifying the line where it is located and a description.