Introduction to useful Web exploitation concepts for solving UniWA-CTF

Michael Kokkos, Alexander Soloviev

Aigaleo - Athens, Greece

Abstract

Today's society is heavily reliant on Web services and the Internet in general. As the importance of the Internet grows, malicious activity is growing too. That makes the field of Web application security more relevant than ever, as a seemingly small bug in a web service can cost a lot of resources or even lives. Penetration testing and code analysis are both viable techniques of securing web applications, with each of them taking a different approach on the problem. This paper introduces the reader into a plethora of important concepts to assist the participants in the UniWA-CTF to solve the Web exploitation challenges and develop a general understanding about the Web.

Keywords: Security, Web application, Malicious activity, Penetration testing, Code analysis

1. Introduction

In the recent years, following the trend of the vast expansion of the web, cybercrime has been a major issue and threatens financial institutions, governments, individuals and other stakeholders globally [1].

Almost 500 million consumers have ever been the victim of a cyber crime, with nearly 350 million of them in the 2019 alone. According to 2019 NortonLifeLock Cyber Safety Insights Report, and based on data collected from ten countries [2]. This report describes many types of cyber attacks that happened, along with many other useful analytics.

Mitigation of the various threats requires a certain level of familiarity with web security practices from the developers. But it is not enough, as to minimize the probability of a security breach, field experts are required too, so that they can perform tasks such as penetration testing and code auditing before a web application goes into production mode.

UniWA-CTF web challenges and boot-to-root boxes introduce the participants into some basic concepts of web exploitation, including SQL injections, static code analysis and provide introductory theoretical knowledge about the functionality of the search engines, specifically about the web crawlers.

2. Static Code Analysis

Figure 1: QR code that leads to the "What is Static Code Analysis?" video, from the channel "Simplified Coding"

Static analysis of source code is a scalable method for discovery of software faults and security vulnerabilities [3]. It is a "white-box" approach, which means that every information that the tester needs are available. The tester analyzes the source code, and tries to discover potential vulnerabilities.

This technique comes in contrast with another well-known technique used to secure a web application which is called "Penetration testing" and it's a "black-box" approach, which means that the tester does not have any information about the application and its function. The tester tries to discover vulnerabilities from the point of view of an attacker using various techniques and tools that the attacker would use in a similar situation.

An example of static code analysis tool is the "inspector", which is an integrated tool in the majority of Web browsers and allows the user to view the source code of a website or a web application (that can include HTML, JavaScript, CSS, etc) that is written with. Other functions include a JavaScript console, access and modification of the cookies, viewing the Web requests, and many other uses that are out of the scope of the paper.

In the example of Figure 2, it is presented the use of the inspector tool of the Firefox web browser. Its use is similar among the other web browsers that the tool is embedded too.

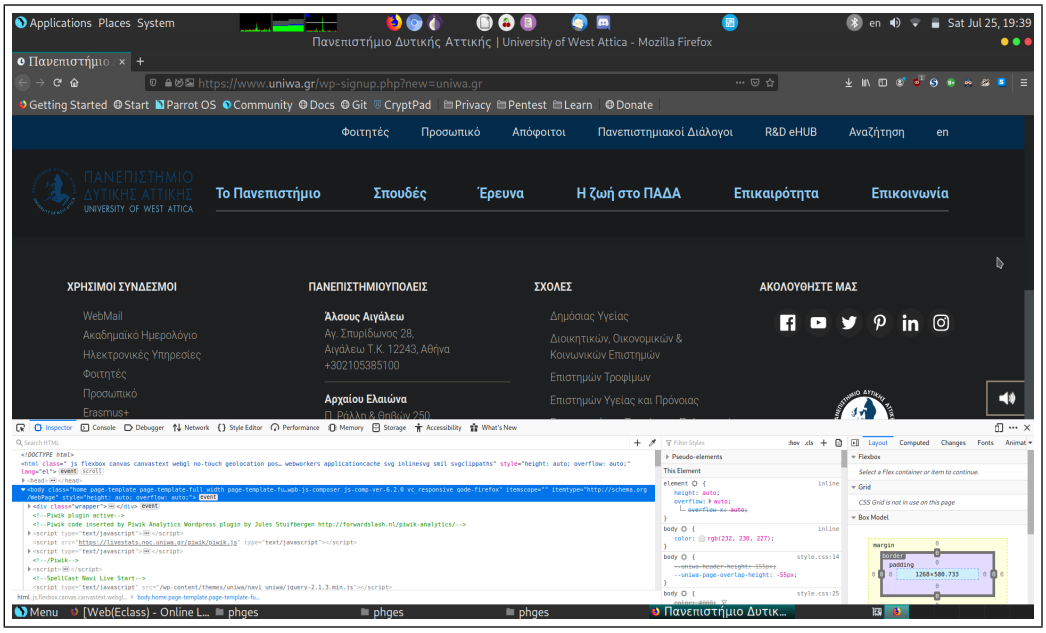

Figure 2: Inspector tool's User Interface in Mozilla Firefox web browser

3. Robots Exclusion Protocol

Figure 3: QR code that leads to the "What is a Robots.txt file?" video, from the channel "IgniteVisibility"

The robots exclusion protocol or simply robots.txt, is an unofficial standard which tells web crawlers and other web robots which pages or files the crawler can or can't request from a site [4].

3.1. Web crawlers

Web crawlers are software applications designed for exploring web applications automatically [5]. They are usually used by search engines so that they can keep their databases updated and help the end-users find their desired information with precision by querying these databases, but they can also be used with malicious intent, to steal data or staging DDOS attacks [6].

There are three main motivations of using the web crawlers for benevolent purposes [5]:

- Content indexing for search engines.
- Automated testing and model checking for web applications.
- Automated security testing and vulnerability assessment.

The process of crawling starts by giving the software a set of seed URLs as an input, then it downloads all the web pages associated with these URLs. After fetching a web page, the URL is removed from the working queue. Lastly, it parses the downloaded page, extracts the linked URLs from it, and adds new URLs to the list of seed URLs. This process, which is described above, continues until all of the contents reachable from seed URLs are reached [5].

To conclude, a web crawler is a piece of software that targets a particular topic and visits and gathers only relevant web pages associated with these URLs [7].

3.2. The Use of Standard

In 1993 and 1994 there have been occasions where crawlers have visited web servers where they weren't welcome for various reasons Vikas Rao Vadi [8]. These incidents indicated the need for established mechanisms for web servers to indicate to the crawlers which parts of the servers should not be accessed. This standard addresses this need with an operational solution [9].

The method that is used to exclude crawlers from a server is the creation of a text file on the server which specifies the files and directories that the administrator does not want the crawlers to access. This file must be accessible via HTTP on the local URL "/robots.txt".

This approach was chosen because of its easy implementation on web servers, and a crawler can find the access policy with only a single document retrieval. [9]

An example of a robots.txt is given in Figure 4. It is strongly suggested to the reader to study the standard's official page so that he or she can further understand how does the protocol functions. The link to the protocol's page is provided in [9].

```
1 User-agent: *
2 Disallow: /aFolder
3 Disallow: /aFile.txt
4 Disallow: /anotherFile.html
5 Disallow: /aThirdFile.bin
```

Figure 4: A robots.txt example

In the Figure 4 example, in the robots.txt file is used the "*" wildcard to apply its rules to all the crawlers, it is not mandatory to apply the rules to all the crawlers, different rules can apply to different crawlers and the name of the corresponding crawler can be used in the "User-agent" field.

The rules can be easily understood without an extended definition. In short, the example of Figure 4 shows that a directory and three files of different formats (one text file, one Hypertext Markup Language file and one binary file) are disallowed from access by the web crawlers. Disallowing web crawlers from accessing directories and files can be done by using the keyword "Dissallow:", followed by the path of the directory or the file on the web server.

4. SQL Injection Attacks

Figure 5: QR code that leads to the "Running an SQL Injection Attack" video, from the channel "Computerphile"

Among the most common types of cyber attacks is the SQL injection [10]. Although this is an old type of attack, with the first public discussions started appearing around 1998 [11], it is still an insisting threat.

SQL injection is a code injection technique. In its simplest form it is a relatively easy attack to perform, even for a person without an extended technical background. The process of the attack is that an input string is injected through the application to change or manipulate the SQL statement so that the attacker can take advantage. This attack can harm the database in several ways, including unauthorized access and manipulation of the database, and disclosure of sensitive data [12].

Different implementations of SQL injection attacks exist, some of them are so simple that they can be successful just by copying and pasting various available payloads that can easily be found on the Internet, more complicated SQL injection vulnerabilities exist too, these can require more in depth knowledge of the SQL language and databases in general so that they can be exploited.

```
or 1=1
```

Figure 6: An example of an SQL payload

In the example of Figure 6, is depicted a typical SQL payload in its simplest form. If an attacker fill a vulnerable web form with that, the attacker "tricks" the database software into giving him or her access. This happens because the attacker manipulates the query to always return "TRUE" making the intended first argument that requires the password to be "TRUE" irrelevant, because the second argument states that one is equal to one which is always "TRUE".

Explaining all the implementations of the attack and its countermeasures would be out of the scope of this paper, however there is a great paper that does so, it is provided in [13], and the participants are advised to study it if they want to deepen their knowledge about these kinds of attacks.

An SQL injection vulnerability can be discovered either by performing static or dynamic code analysis, either by trial and error by utilizing different payloads.

5. Conclusions

Web application penetration testing skills are useful not only to information security engineers or bounty hunters. A basic knowledge of web application security concepts, such as these that were described in this paper, can apply into fields such as these of software and web application development. The stand-alone and boot-to-root web challenges of UniWA-CTF have been designed with that in mind, so that the participation can have a positive effect to a wide range of students.

References

[1] F. Kamy, S. Conrad, V. Jay, Cybersecurity indices and cybercrime annual loss and economic impacts, Journal of Business and Behavioral Sciences 32 (2020) 63–71.

[2] NortonLifeLock Inc., 2020, 2019 Cyber safety insights report global results, URL: https://us.norton.com/nortonlifelock-cyber-safety-report.

[3] K. Goseva-Popstojanova, A. Perhinschi, On the capability of static code analysis to detect security vulnerabilities, Information and Software Technology 68 (2015) 18 – 33. URL: http://www.sciencedirect.com/science/article/pii/S0950584915001366. doi:https://doi.org/10.1016/j.infsof.2015.08.002.

[4] Google, 2020, Introduction to robots.txt, URL: https://support.google.com/webmasters/answer/6062608?hl=en.

[5] S. Mirtaheri, M. Dincturk, S. Hooshmand, G. Bochmann, G.-V. Jourdan, I.-V. Onut, A brief history of web crawlers, Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research (2013) 40–54.

[6] D. Yu, 2016, A hybrid approach to detect malicious web crawlers, URL: https://www.hillstonenet.com/blog/a-hybrid-approach-to-detect-malicious-web-crawlers/.

[7] P. Gupta, K. Johari, Implementation of web crawler, in: 2009 Second International Conference on Emerging Trends in Engineering Technology, 2009, pp. 838–843. doi:10.1109/ICETET.2009.124.

[8] A. T. Vikas Rao Vadi, Crawler and url retrieving queuing, International Journal of Computer Application 1 (2013) 14.

[9] M. Koster, 1994, A standard for robot exclusion, URL: https://www.robotstxt.org/orig.html.

[10] Rapid7, 2020, Common types of cybersecurity attacks, URL: https://www.rapid7.com/fundamentals/types-of-attacks/.

[11] S. M. Kerner, 2013, How was sql injection discovered?, URL: https://www.esecurityplanet.com/network-security/how-was-sql-injection-discovered.html.

[12] M. Humayun, M. Niazi, N. Zaman, M. Alshayeb, S. Mahmood, Cyber security threats and vulnerabilities: A systematic mapping study, Arabian Journal for Science and Engineering 45 (2020). doi:10.1007/s13369-019-04319-2.

[13] W. Halfond, J. Viegas, A. Orso, A classification of sql injection attacks and countermeasures (2006).