Sicurezza e Privatezza

LAB 6 – Web security

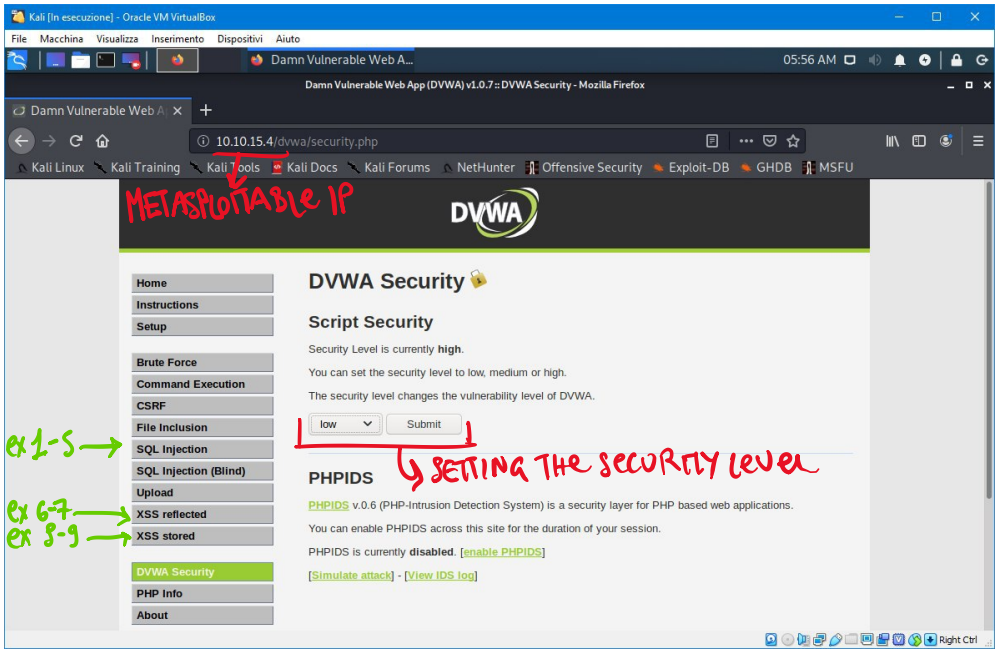
The learning objective of this lab is for students to get the basics of web security with cross-site scripting and SQL injection. After finishing the lab, students should be able to craft malicious payload to exploit vulnerable web pages and understanding how to protect a system against these types of attack.

Students are invited to solve the following exercises, writing a txt file that include payloads used for exploitation and a little description of what the payload do. The submission website is upload.di.unimi.it and the name of the txt file should be **`Lab6NameSurnameUniversityid`** (eg. Lab6MatteoZoia856378.txt). Files not conforming to these rules will be automatically rejected.

The first ten students who will respond correctly to the entire lab will receive a bonus of 0.5pt on the final grade. Exercises with the tag "extra" will not be considered for bonus.

Lab environment

In this lab we will use the metasploitable as victim machine and the Kali Linux as attacker machine. The metasploitable comes with several vulnerable services already included, in particular in this lab we try to attack (with SQL injection and XSS) a web site called DVWA placed in the metasploitable.
First we need to connect in the same virtual subnet the metasploitable (victim) and the Kali Linux (attacker), you can follow the steps in the video for more details
http://security.di.unimi.it/secpriv1920/video/SecLab4_MetasploitBof.mp4
After settings up the virtual network between metasploitable and Kali Linux you simply need to start the metasploitable, when it boot up, the vulnerable website starts automatically. For visiting the DVWA from the Kali Linux you need to type in the URL bar of the Kali browser the IP address of the metasploitable (you can find it with an ifconfig on you metasploitable).



In particular we try to attack the website section “SQL injection” in exercises 1 to 5, section “XSS reflected” in exercise 6 to 7 and sections “XSS stored” in exercises 8 to 9. Remember to set the DVWA security as said in exercises (and in the figure below).

Exercise #1: SQL injection warmup (security low)

SQL injection is a code injection technique that exploits the vulnerabilities in the interface between web applications and database servers. The vulnerability is present when user’s inputs are not correctly checked within the web applications before being sent to the back-end database servers. Many web applications take inputs from users, and then use these inputs to construct SQL queries, so the web applications can get information from the database. **The objective of this task is to get familiar with SQL injection by playing with the very first (and simple) injection to retrieve all the user in the database.**

Exercise #2: Information leak (security low)

SQL injection is basically a technique through which attackers can execute their own malicious SQL statements generally referred as malicious payload. Through the malicious SQL statements, attackers can steal information from the victim database. **The objective of this task is to get information about database tables and hosting server (eg. running OS or hostname machine).**

Exercise #3: Steal account credentials (security low)

Storing account credential in a database is always a crucial point, the passwords should always be hashed so if an hacker steal them through an SQL injection, they are still protected. With the last exercise you have discovered some tables name**, can you spot information about account passwords? What encryption algorithm is used?**

Extra 3.1: if you want you can try to login as an arbitrary user by update the password of a certain account with your favorite hash; but remember that in real world this makes irreversible changes in database, moreover your activity maybe logged somewhere.

Exercise #4: Read arbitrary file (security low)

In SQL language (with the right function) is possible to read a file and return it content as a string. As hacker we can try to read a file from the remote system. The file that we are always looking for is of course the passwd where older Linux systems were storing the passwords. **Can you print with an SQL injection the password file?**

Exercise #5: Countermeasure

The fundamental problem of the SQL injection vulnerability is the failure to separate code from data. When constructing a SQL statement, the program (e.g. PHP program) knows which part is data and which part is code. Unfortunately, when the SQL statement is sent to the database, the boundary has disappeared. On the bottom of the web page you can view the PHP source code, with that code in mind **can you describe in a few words a way to avoid the SQL injection problem?**

Exercise #6: XSS warmup (security low)

An XSS allows an attacker to inject a script into the content of a website or app. When a user visits the infected page, the script will execute in the victim's browser. This allows attackers to steal private information like cookies, account information, or to perform custom operations while impersonating the victim's identity.
Under XSS reflected you can find a simple webpage, **your goal for this section is to create a URL that when clicked, displays the cookie of the victim in an alert.**

Exercise #7: Advanced XSS reflected (security medium)

No more detail, **can you do the same as before with DVWA security level set to medium?**
To check for a possible XSS vulnerability, you need to test every point of user input to see if you can inject HTML and JavaScript code and whether it gets delivered to the output of the page.

Extra 7.1, can you do the same as before with DVWA security level set to high?

Exercise #8: Simple blog post (security low)

An attacker uses Stored XSS to inject malicious content (payload), most often JavaScript code, into the target application. If there is no input validation, this malicious code is permanently stored (persisted) by the target application. When a victim opens the affected web page in a browser, the XSS attack payload is served to the victim's browser as part of the HTML code. This means that victims will end up executing the malicious script once the page is viewed in their browser.
The objective of this task is to inject malicious code that pops up the cookie for every user that visiting the webpage.

Exercise #9: Avoiding sanitization (security medium)

Set the security level to medium and **try to do the same as before.** With security set to medium, some sanitization of the input fields is done after you send the post to server. Your goal is bypass the sanitization in order to store you XSS successfully on the database.