1 .INTRODUCTION

1.1 OVERVIEW:-

In the early days of the Internet, the World Wide Web consisted only of web sites. These were essentially information repositories containing static documents. Web browsers were invented as a means of retrieving and displaying those documents. The flow of interesting information was one-way, from server to browser. Most sites did not authenticate users, because there was no need to. Any security threats arising from hosting a website were related largely to vulnerabilities in web server software (of which there were many). If an attacker compromised a web server, he usually would not gain access to any sensitive information, because the information held on the server was already open to public view. Rather, an attacker typically would modify the files on the server to deface the web site's contents or use the server's storage. Today, the World Wide Web is almost unrecognizable from its earlier form. The majority of sites on the web are in fact applications (see Figure 1-2). They are highly functional and rely on two-way flow of information between the server and browser. They support registration and login, financial transactions, search, and the authoring of content by users. The content presented to users is generated dynamically on the fly and is often tailored to each specific user. Much of the information processed is private and highly sensitive. Security, therefore, is a big issue. No one wants to use a web application if he believes his information will be disclosed to unauthorized parties.

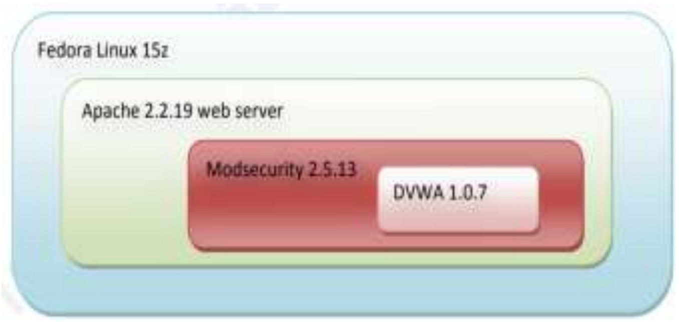


Fig:-Modsecurity

A **web application firewall** is a special type of application firewall that applies specifically to web applications. It is deployed in front of web applications and analyzes bi-directional web-based (HTTP) traffic - detecting and blocking anything malicious. The OWASP provides a broad technical definition for a WAF as "a security solution on the web application level which - from a technical point of view - does not depend on the application itself. WAF can be a virtual or physical appliance that prevents vulnerabilities in web applications from being exploited by outside threats. These vulnerabilities may be because the application itself is a legacy type or it was insufficiently coded by design. Therefore, securing web applications is becoming one of the most important things you need to pay attention as an end-user or as a business user. It inspects the application layer so it usually comes as an appliance type or as a server module. It generally identifies and blocks common web attacks such as cross-site scripting (XSS) and SQL Injection by customizing the rules. Therefore, the customization of its rules is very significant and requires high maintenance. There are many ways to protect a web application, such as implementing a secure coding practice, managing secure configuration, performing vulnerability assessment and deploying a web application firewall, but there is no silver bullet that it will protect the application entirely. Using a web application firewall is just one method that helps to protect such an application. This technology is relatively new comparing with other technologies, but it can become a powerful solution when you configure and use it properly .In this paper, Modsecurity is going to be used to demonstrate how to secure a web application using a WAF. Modsecurity protects web applications from a range of web attacks and allows monitoring of HTTP traffic with not many interference in the existing infrastructure. It is an open source WAF module for the Apache web server and it has been maintained by SpiderLabs , Trustwave. Since this is an open source product, it comes with free license and many users contribute to the community to improve and maintain the product A WAF is not a tool that just blocks the malicious activity on the application layer. It can also be used to analyze and detect malicious traffic that attacks your critical application .Therefore, this paper will also show how to analyze common web attacks by using WAF's detection and logging ability along with the Apache server's logging ability.

1.2 Evolution of web applications:-

Web applications are important for the development of the Internet. They improve the quality of a browser's work and the entire Internet can function plainly. A web application is a client-software application run by the client in a browser. The main function of a browser is to show the information received from a server and send the user's data back. The main advantage of this approach is the fact that clients do not depend on the user' specific operating system; therefore, web applications are cross-platform services. Due to this universal feature, web apps became very popular in 1990s and 2000s. Developers do not need to prepare different versions of the same app for Microsoft Window, Mac OS, Linux, etc. An app is created only once for any platform and it can work on any operating system. However, the different practical realization of HTML, CSS, DOM and other interfaces in browsers can cause troubles during the web application development and their further support. Moreover, a web app can work incorrectly because of the user's opportunity to change browser's settings in the way he wants.

The year 1995 is a crucial year in the era of the Internet. Netscape Communications presented JavaScript, a client-side scripting language that enables programmers to improve the user interface with the dynamic elements. JavaScript made the Internet faster and more productive because the data was no longer sent to the server to generate the whole web page. The embedded scripts fulfill various tasks on the specific downloaded page 'right on the spot'. JavaScript is one of the three most notable technologies (with HTML and CSS) of content production for WWW. It has the application programming interface that enables experts to work with texts, dates and various regular expressions.

2.SYSTEM ANALYSIS

2.1EXISTING SYSTEM:-

- . Web application vulnerabilities can allow attackers to steal data, immobilise or hijack your application. With increasing reliance on third-party software and code, organisations can find themselves at risk if supply chain security policies don't meet their own policies and frameworks.
- The Web Application Firewall is a managed service that analyses Layer 7 application traffic. This platform is specifically tuned to monitor only the target applications, and to inspect all inbound and outbound traffic.
- Till now we saw a lot of firewalls which are both open-source and commercial. Commercial firewalls are too expensive for low and medium enterprises for configuring and protecting their applications. There are some open-source firewalls which protects web applications from only limited kind of web application attacks.
- Modsecurity firewall offers visibility, detection, and protection against application and OWASP Top 10 vulnerabilities. Which is the key feature missing with other open-source web application firewalls

2.2PROPOSED SYSTEM:-

ModSecurity is a toolkit for real-time web application monitoring, logging, and access control. , ModSecurity gives you access to the HTTP traffic stream, in real-time, along with the ability to inspect it. This is enough for real-time security monitoring. There's an added dimension of what's possible through ModSecurity's persistent storage mechanism, which enables you to track system elements over time and perform event correlation. You are able to reliably block, if you so wish, because ModSecurity uses full request and response buffering. Web servers traditionally do very little when it comes to logging for security purposes. They log very little by default, and even with a lot of tweaking you are not able to get everything that you need. I have yet to encounter a web server that is able to log full transaction data. ModSecurity gives you that ability to log anything you need, including raw transaction data, which is essential for forensics. In addition, you get to choose which transactions are logged, which parts of a transaction are logged, and which parts are sanitized. Security assessment is largely seen as an active scheduled event, in which an independent team is sourced to try to perform a simulated attack. Continuous passive security assessment is a variation of real-time monitoring, where, instead of focusing on the behavior of the external parties, you focus on the behaviour of the system itself. It's an early warning system of sorts that can detect traces of many abnormalities and security weaknesses before they are exploited.

ADVANTAGES OF PROPSED SYSTEM:-

- **Flexibility**
  - ModSecurity achieves flexibility by giving you a powerful rule language, which allows you to do exactly what you need to, in combination with the ability to apply rules only where you need to.

- **Passiveness**
  - ModSecurity will take great care to never interact with a transaction unless you tell it. ModSecurity will give you plenty of information , but ultimately leave the decisions to you.

- **Predictability**
  - There's no such thing as a perfect tool, but a predictable one is the next best thing. Armed with all the facts, you can understand ModSecurity's weak points and work around them.

- **Quality over quantity**
  - Over the course of six years spent working on ModSecurity, we came up with many ideas for what ModSecurity could do. We didn't act on most of them. We kept them for later. Why? Because we understood that we have limited resources available at our disposal and that our minds (ideas) are far faster than our implementation abilities.

2.3 SYSTEM REQUIREMENT SPECIFICATION:-

2.3.1 Hardware Requirements

- 2 GB RAM (system memory).
- 25 GB of hard-drive space (or USB stick, memory card or external drive but see Live CD for an alternative approach)
- Either a CD/DVD drive or a USB port for the installer media.

2.3.2 Software Requirements:-

Operating System	:	Ubuntu.
Tool	:	VMware WorkStation.
Database	:	Apache, My SQL.
Languages	:	ModSec.

2.4 Introduction to DVWA:-

DVWA is a DAMM VULNERABLE WEB APP coded in *PHP/MYSQL*. Seriously it is too vulnerable. In this app security professionals, ethical hackers test their skills and run this tools in a legal environment. It also helps web developer better understand the processes of securing web applications and teacher/students to teach/learn web application security in a safe environment. The aim of DVWA is to practice some of the most common web vulnerability, with various difficulties levels.

How to use DVWA?

You just have to go to this link http://www.dvwa.co.uk/ and download. Once you downloaded. Install it on the virtual machine (**VMWARE or VIRTUALBOX**) You will require **XAMPP** (for windows) Then DVWA gives your local

IP you can check this by typing in the virtual machine (**if confing**) Then you have to type this IP in the Browser. That's it now you in the DVWA Environment. **What are the Benefits of DVWA?** Hacking anything without the permission is a **Crime**. So as a student or beginners from where you got this permission so you can use this. For advanced users to sharpen their skill DVWA is the **best platform**.

- In DVWA you do not have to take permission from other. You can simply install this in a virtual environment and start using it.
- It is very simple to install.

- This is the best place to do **hacking**.
- In fact, this is running in your local environment and it is **totally legal**.

DIFFICULTIES LEVELS IN DVWA ?

As the name suggests DVWA has many web vulnerabilities. Every vulnerability has four different security levels, low, medium, high and impossible. The security levels give a challenge to the 'attacker' and also shows how each vulnerability can be counter measured by secure coding Security level.

Impossible: In this level, you will face challenges like CTF and it is harder than the other level. This level gives difficulties which we face in the real world.

High: This vulnerability level gives the user an example of how to secure the vulnerability via secure coding methods. It lets the user understand how the vulnerability can be counter measured. This level of security should be un-hackable however as we all know this is not always the case. So if you manage to bypass it, that you are doing right.

Medium: This security level's purpose is to give the 'attacker' a challenge in exploitation and also serve as an example of bad coding/security practices.
Low: This security level is meant to simulate a website with no security at all implemented in their coding. It gives the 'attacker' the chance to refine their exploitation skills.

In DVWA we can test various **different kinds of Vulnerabilities.**

1) BRUTE FORCE: In the brute force vulnerability we can test whether the login portal is vulnerable to the brute force or not. Here we can try almost all combination of words, number, special symbol**.** We can also use the dictionary file. The main goals are to crack the login name and password. Brute force can be applied in the different parameter. Here we have to brute force login screen.

2) COMMAND INJECT: In the command inject vulnerability the goal is an execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user-supplied data to a system shell. In this attack, the attacker sends operating system commands are usually executed with the privileges of the vulnerable application so here you have an empty field so you can execute os commands on that.

3) Cross-Site Request Forgery (CSRF) :- CSRF is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. CSRF attacks specifically target state-changing requests, not theft of data since the attacker has no way to see the response to the forged request. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state-changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application. So here we can create a fake page and send it to the victim to perform the necessary steps.

4) FILE UPLOAD:

The file upload is a very simple feature just upload the file to the server. To reduce the risk we may only accept certain file extensions, but attackers are able to encapsulate malicious code into inert file types. Testing for malicious files verifies that the application/system is able to correctly protect against attackers uploading malicious files. The application may allow the upload of malicious files that include exploits or shell code without submitting them to malicious file scanning here we have to upload shell into the server.

5) INSECURE CAPTCHA:

CAPTCHA **(***"Completely Automated Public Turing test to tell Computers and Humans Apart"***)** Captchas are usually used to prevent robots to make an action instead of humans. It should add an extra layer of security but badly configured it could lead to unauthorized access. Here what we have to do is to bypass the captcha security function by using the tamper data or other methods.

6) SQL INJECTION:

SQL Injection (SQLi) refers to an injection attack where an attacker can execute malicious SQL statements that control a web application's database server. Since an SQL Injection vulnerability could possibly affect any website or web application that makes use of a SQL-based database, the vulnerability is one of the oldest, most prevalent and most dangerous of web application vulnerabilities. An attacker can use it to **bypass a web** application's authentication and authorization mechanisms and retrieve the contents of an entire database. SQL Injection can also be used to add, modify and delete records in a database, affecting data integrity.

7) SQL INJECTION BLIND:

In the Blind SQL we cannot see the response at that time. So any type of query which we fire will not show any response at that time. It involves a lot of guesswork on the part of the attacker and takes time for them to gain an understanding of the structure of the data they're trying to get at, but with skill and perseverance, all the data is still at their fingertips. To make things worse, the guesswork can be easily automated, reducing the time it takes to steal your data. Two techniques that are commonly used to do this – **Content-based Blind SQL Injection and Time-based Blind SQL** Injection. Here we have to check whether it is vulnerable to SQL blind or not example by using the time delay.

8) WEAK SESSION IDs:

The session prediction attack focuses on predicting session ID values that permit an attacker to bypass the authentication schema of an application. By analyzing and understanding the session ID generation process, an attacker can predict a valid session ID value and get access to the application. In the first step, the attacker needs to collect some valid session ID values that are used to identify authenticated users. Then, he must understand the structure of session ID, the information that is used to create it, and the encryption or hash algorithm used by the application to protect it. Some bad implementations use sessions IDs composed by username or other predictable information, like timestamp or client IP address. In the worst case, this information is used in the clear text or coded using some weak algorithm like base64 encoding.

In addition, the attacker can **implement a brute force technique** to generate and test different values of session ID until he successfully gets access to the application. So here we have to find out whether this is producing the weak session id or not.

9) XSS (DOM):

Cross-site scripting is a vulnerability that allows an attacker to send malicious code (usually in the form of Javascript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context **allowing the attacker to access any cookies or session tokens** retained by the browser. While a traditional cross-site scripting vulnerability occurs on the server-side code, document object model based cross-site scripting is a type of vulnerability which affects the script code in the client's browser**.**

10)XSS (REFLECT):

Reflected XSS attacks, also known as non-persistent attacks, occur when a malicious script is reflected off of a web application to the victim's browser. This vulnerability is typically a result of incoming requests not being sufficiently sanitized, which allows for the manipulation of a web application's functions and the activation of malicious scripts. Here we can use a various javascript function to exploit this.

11)XSS (STORED):

The **persistent (or stored) XSS** vulnerability is a more dangerous version of a cross-site scripting flaw it occurs when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing, without proper HTML escaping. Here we also have to use javascript but here if you successfully embedded the JavaScript then this will result into the xss stored means that malicious content will be visible to anyone. There is also more vulnerable machine available like **WAPP, Mutillidae, Metasplotiable** you can also try these.

2.5 FEASIBILITY STUDY:-

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

2.5.1 Economical Feasibility:-

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.5.2 Technical Feasibility:-

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.5.3 Social Feasibility:-

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.LITERATURE SURVEY

3.1 HTTP and HTTPS:-

HTTP (Hypertext Transfer Protocol)

HTTP is an client-server protocol that allows clients to request web pages from web servers. It is an application level protocol widely used on the Internet. Clients are usually web browsers. When a user wants to access a web page, a browser sends an HTTP Request message to the web server. The server responds with the requested web page. By default, web servers use the TCP port 80.

Clients and web servers use request-response method to communicate with each other, with clients sending the HTTP Requests and servers responding with the HTTP Responses. Clients usually send their requests using GET or POST methods, for example **GET /homepage.html**. Web servers responds with a status message (200 if the request was successful) and sends the requested resource.

An example will clarify this process:

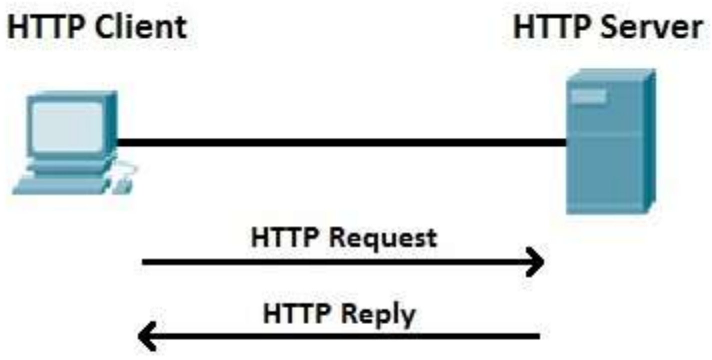


Fig:- Communication between Client and Server

The client wants to access **http://google.com** and points his browser to the URL **http://google.com** (this is an example of an HTTP Request message). The web server hosting **http://google.com** receives the request and responds with the content of the web page (the HTTP response message).

Web servers usually use a well-known TCP port 80. If the port is not specified in a URL, browsers will use this port when sending HTTP request. For example, you will get the same result when requesting http://google.com and http://google.com:80

HTTPS (Hypertext Transfer Protocol Secure)

Hypertext Transfer Protocol Secure is a secure version of HTTP. This protocol enables secure communication between a client (e.g. web browser) and a server (e.g. web server) by using encryption. HTTPS uses **Transport Layer Security (TLS)** protocol or its predecessor **Secure Sockets Layer (SSL)** for encryption.

HTTPS is commonly used to create a secure channel over some insecure network, e.g. Internet. A lot of traffic on the Internet is unencrypted and susceptible to sniffing attacks. HTTPS encrypts sensitive information, which makes a connection secure.

HTTPS URLs begin with **https** instead of **http**. In Internet Explorer, you can immediately recognize that a web site is using HTTPS because a lock appears to the right of the address bar:

HTTP Methods –

When you are attacking web applications, you will be dealing almost exclusively with the most commonly used methods: GET and POST. You need to be aware of some important differences between these methods, as they can affect an application's security if overlooked.

The GET method is designed to retrieve resources. It can be used to send parameters to the requested resource in the URL query string. This enables users to bookmark a URL for a dynamic resource that they can reuse. Or other users can retrieve the equivalent resource on a subsequent occasion (as in a bookmarked search query). URLs are displayed on-screen and are logged in various places, such as the browser history and the web server's access logs. They are also transmitted in the referrer header to other sites when external links are followed. For these reasons, the query string should not be used to transmit any sensitive information.

The POST method is designed to perform actions. With this method, request parameters can be sent both in the URL query string and in the body of the message. Although the URL can still be bookmarked, any parameters sent in the message body will be excluded from the bookmark. These parameters will also be excluded from the various locations in which logs of URLs are maintained and from the Referrer header. Because the POST method is designed for performing actions, if a user clicks the browser's Back button to return to a page that was accessed using this method, the browser does not automatically reissue the request. Instead, it warns the user of what it is about to do, as shown in Figure 3-1. This prevents users from unwittingly performing an action more than once. For this reason, POST requests should always be used when an action is being performed.

In addition to the GET and POST methods, the HTTP protocol supports numerous other methods that have been created for specific purposes. Here are the other ones you are most likely to require knowledge of:

HEAD functions in the same way as a GET request, except that the server should not return a message body in its response. The server should return the same headers that it would have returned to the corresponding GET request. Hence, this method can be used to check whether a resource is present before making a GET request for it.

3.2 <u>OWASP Top 10 Application Security Risks – 2017:</u>

A1 Injection

Injection flaws, such as SQL, OS, XXE, and LDAP injection occur when un trusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

A2-Broken Authentication and Session Management

Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities (temporarily or permanently).

A3-Cross-Site Scripting (XSS)

XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user supplied data using a browser API that can create JavaScript. XSS allows attackers to execute scripts in the victim’s browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

A4-Broken Access Control

Restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users’ data, change access rights, etc.

A5-Security Misconfiguration

Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, platform, etc. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

A6-Sensitive Data Exposure

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

A7-Insufficient Attack Protection

The majority of applications and APIs lack the basic ability to detect, prevent, and respond to both manual and automated attacks. Attack protection goes far beyond basic input validation and involves automatically detecting, logging, responding, and even blocking exploit attempts. Application owners also need to be able to deploy patches quickly to protect against attacks.

A8-Cross-Site Request Forgery (CSRF)

A CSRF attack forces a logged-on victim’s browser to send a forged HTTP request, including the victim’s session cookie and any other automatically included authentication information, to a vulnerable web application. Such an attack allows the attacker to force a victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

A9-Using Components with Known Vulnerabilities

Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defences and enable various attacks and impacts.

A10-Underprotected APIs

Modern applications often involve rich client applications and APIs, such as JavaScript in the browser and mobile apps that connect to an API of some kind (SOAP/XML, REST/JSON, RPC, GWT, etc.). These APIs are often unprotected and contain numerous vulnerabilities.

4.SYSTEM DESIGN

4.1 Dataflow Diagram

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

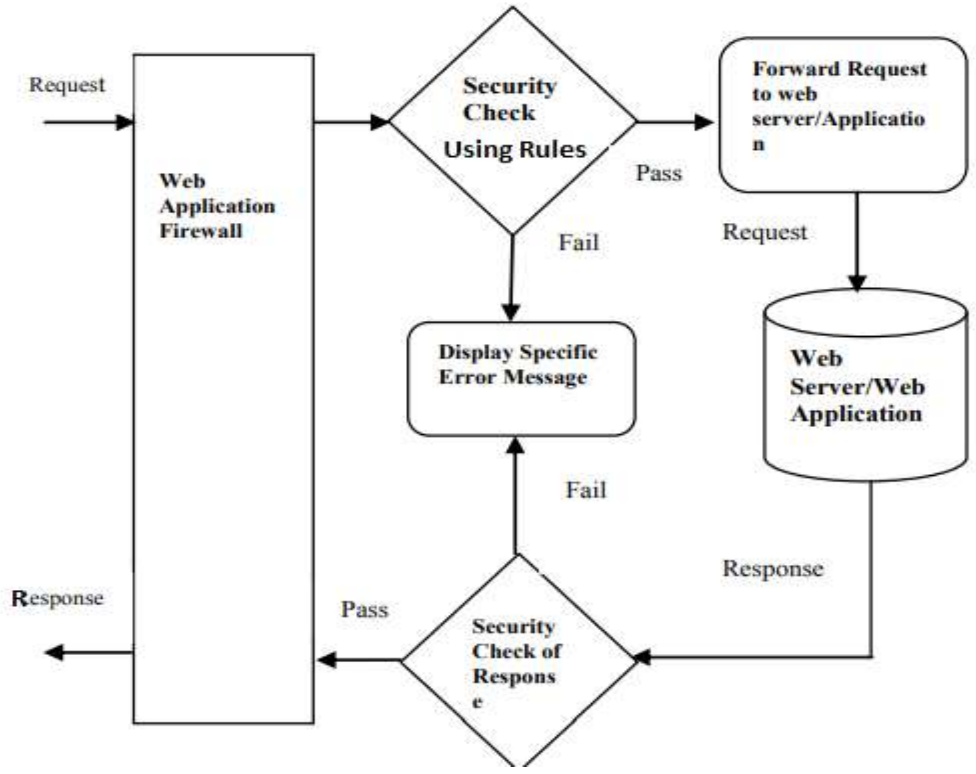

Request
Web Application Firewall
Security Check Using Rules
Pass
Forward Request to web server/Applicatio n
Request
Fail
Display Specific Error Message
Web Server/Web Application
Fail
Response
Response
Pass
Security Check of Respons e

Fig:-Class Diagram for ModSecurity Firewall

4.2 UML DIAGRAMS

Unified Modeling Language (UML) Diagram:

➔UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

➔The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to or associated with UML.

➔The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

➔The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

➔The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

<u>4.2.1 CLASS DIAGRAM</u>:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information

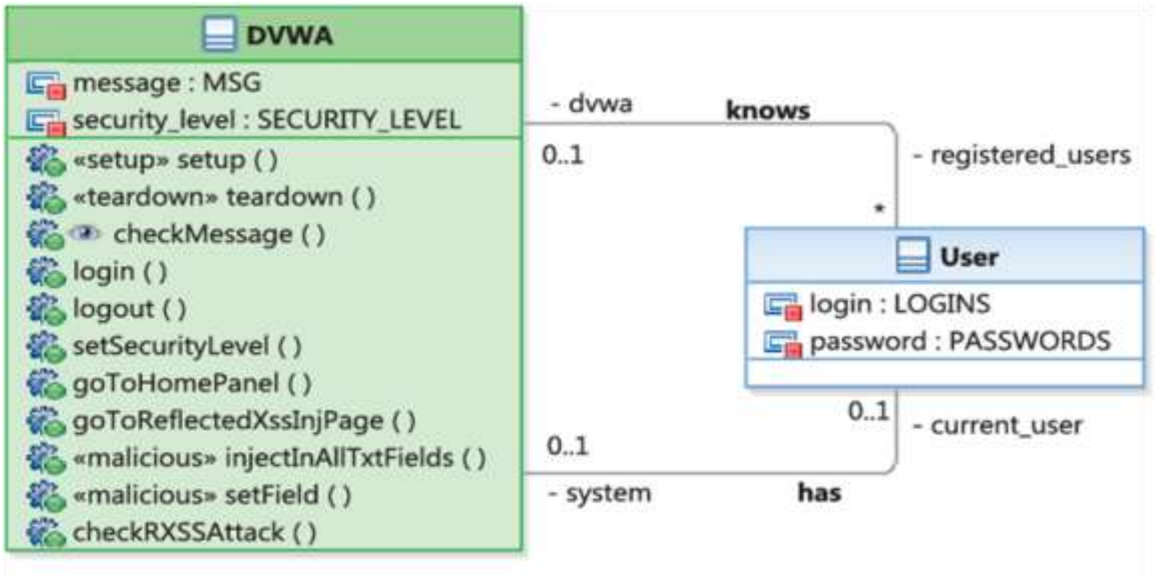


Fig:-Class Diagram for DVWA

<u>4.2.2 USE CASE DIAGRAM:</u>

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor.Roles of the actors in the system can be depicted

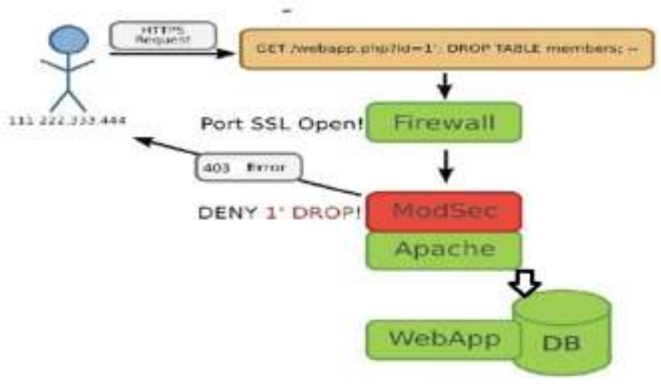


Fig: Use case diagram for ModSecurity Firewall in DVWA

4.3 MODULES DESCRIPTION

MODULES:-

1. Analysing the Attacks.
2. Blocking the Attacks.

In this section, the most common vulnerabilities such as XSS and SQL injection are exploited against DVWA. First, the attacks run without the Modsecurity blocking rules in place and the results are analyzed by correlating Wire shark traffic captures, Apache access/error logs and Modsecurity audit logs. Then, the appropriate blocking rule is placed in the Modsecurity configuration file based on the analysis. Lastly, the attack runs again to verify the rule by observing the response as well as by analyzing the logs to check if the attack is stopped.

4.3.1 Analyzing XSS attacks:-

"XSS vulnerability enables an attacker to target other users of the application, potentially gaining access to their data, performing unauthorized actions on their behalf, or carrying out other attacks against them" (Stuttered & Pinto, 2007). In this study, the XSSinjectionstring,"%3C%2FTITLE%3E%3CSCRIPT%3Ealert%28%22XSS%22%29%3B%3C%2FSCRIPT%3E#" is injected into the name parameter as shown below: *http://127.0.0.1/dvwa/vulnerabilities/xss_r/?name=%3C%2FTITLE%3E%3CSCRIPT%3Ealert%28%22XSS%22%29%3B%3C%2FSCRIPT%3E#*.

The Modsecurity audit log has certain formats. First, it starts with a unique identifying string such as "cb5e7204" to group the rest of the log entry. After this string, it has a capital letter, such as "A", which represents the audit log part. For example, "A" represents an audit log header, "B" represents a request header, "F" represents a response header, "H" represents an audit log trailer and "Z" represents the end of an audit log entry. There are additional parts you can add by adding "SecAudit Log Parts" line, but A, B, F, H and Z are the default settings. As shown in Figure 6, Modsecurity allowed the request because by default, it allows any requests that do not match with any SecRule.

<u>4.3.2 Blocking XSS attacks</u>:-

From the various logs in the above section, it is found that this XSS attack uses keywords such as "SCRIPT" and "alert" in the uniform resource identifier (URI). The easy and quick way to block this type of XSS attack is using a Target variable called "REQUEST_URI" which examines a text in URI. For example, the following line is added to the configuration:

```
SecRule   REQUEST_URI    "SCRIPT"|"alert"
```

This rule denies any requests that include the words "SCRIPT" or "alert" in their URI. Though this rule does not have a "deny" variable, it denies such requests because the "SecDefaultAction" is set to "deny" in Figure 2. However, this is a very simple rule that can block only simple XSS attacks. An attacker can bypass this type of filtering by encoding or by injecting the script into other places, such as a cookie field. To block more advanced XSS attacks, you can add common attack strings to the existing rules, such as the ones presented in Table1, or to include the corresponding XSS base rules from the CRS rule set.

```
SecRule ARGS
"(?i)(<script[^>]*>[\s\S]*?<\/script[^>]*>|<script[^>]*>[\s\S]*?<\/script[[\s\S]]*[\s\S]|<s
cript[^>]*>[\s\S]*?<\/script[\s]*[\s]|<script[^>]*>[\s\S]*?<\/script|<script[^>]*>[\s\S]*?
)" "id:'973336',phase:2,rev:'1',ver:'OWASP_CRS/2.2.8',accuracy:'8',msg:'XSS Filter -
Category 1: Script Tag
Vector',tag:'OWASP_CRS/WEB_ATTACK/XSS',tag:'OWASP_TOP_10/A2',severity:'2'
```

Analyzing SQL Injection Attacks:-

SQL Injection is another common web application attack method. This vulnerability allows attackers to inject malicious SQL statements to interact with the backend database. From this injection, the attacker may be able to obtain data as well as to execute malicious commands o the database (Stuttard & Pinto, 2007). In

this study, the SQL injection string (' union all select user, password from dvwa.users#) was injected into the “id” parameter, as shown below:
http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=%27+union+all+select+user%2C+password+from+dvwa.users%23&Submit=Submit#

Then, the browser responded with a list of users and the corresponding password hashes.

Blocking SQL Injection attack:-

From the various logs in the above section, it is found that this SQL Injection attack uses a keyword “union” in the argument. Similarly to the prevention of the XSS attack, you can add the following SecRule line to the configuration to block the SQL Injection attack. In this example, “msg” argument is added to the rule so it writes the

```
SecRuleARGS        "union”        “msg: ‘SQL Injection’”
```

log message that explains why it is blocked:

This rule denies any requests that include the keyword “union” as an argument. Again this is a very simple type of SQL injection attacks. To block more advanced SQL Injection attacks, you can add more common attack strings to the existing rule, such as the ones presented in Table 2, or the corresponding SQL Injection base rules from the CRS rule set should be included.

5.Implementation

5.1 Modsecurity Rule Writing:-

Modsecurity needs rules to operate. In Modsecurity, each one of these rules is called SecRule. SecRule has many features and functions. The basic syntax of a rule, which will be briefly explained later, is shown below:

```
SecRule Target Operation  [Actions]
```

The ‘Target’ variable is a part of the request or the response that is going to be examined. The ‘Operation’ is a part of the rule that is going to be compared with a matching variable. By default, it uses regular expressions if nothing is specified. The ‘Actions’ are optional variables that perform specific actions when there is a matching variable. For example, it can either allow or deny the web traffic by also returning the corresponding status codes. If no actions are specified, it will take the settings from the SecDefaultAction statement Modsecurity does not provide general protection against normal web attacks by default, unless you install the appropriate rules. Writing such rules can be a complex and a time consuming process. Therefore, Trustwave’s SpiderLabs founded the OWASP Modsecurity core rule set (CRS) project. While intrusion detection systems depend on vulnerability signatures, CRS helps Modsecurity to protect web applications from unknown vulnerabilities.

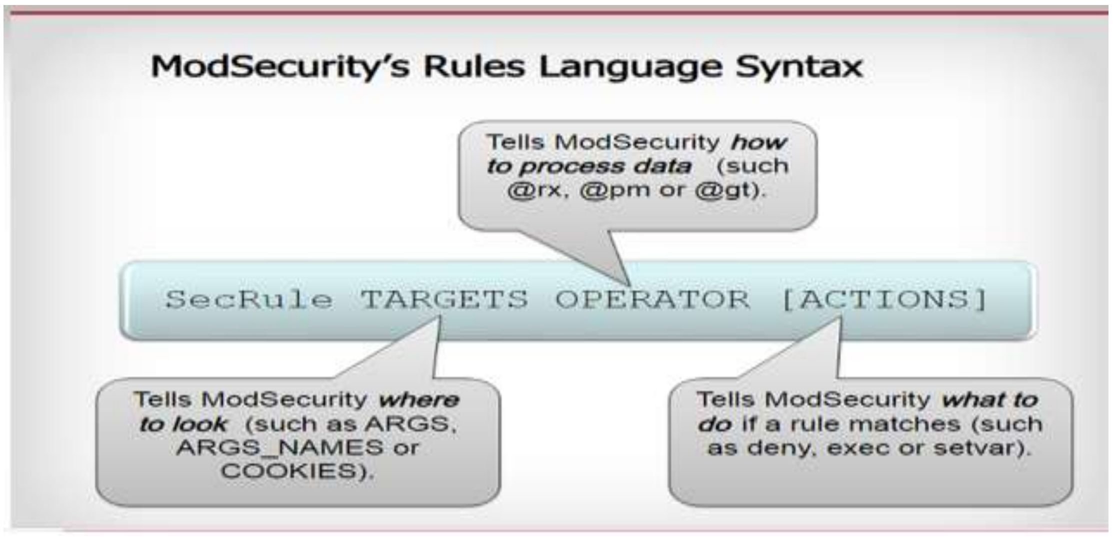

ModSecurity's Rules Language Syntax
Tells ModSecurity *how to process data* (such @rx, @pm or @gt).
SecRule TARGETS OPERATOR [ACTIONS]
Tells ModSecurity *where to look* (such as ARGS, ARGS_NAMES or COOKIES).
Tells ModSecurity *what to do* if a rule matches (such as deny, exec or setvar).

Fig:-ModSecurity Rule Syntax

The CRS rule set is very easy to follow and deploy since it provides excellent comments; the most updated rule sets can be downloaded from the OWASP Modsecurity CRS Project Site.

Here our task is to defend DVWA from the attacks even it is set to low security.

✓ First we need to download and configure DVWA.

✓ Need to install and configure ModSecurity.

5.2 Downloading and configuring DVWA in Ubuntu:

Step 1: Download DVWA bundle directly from dvwa website and extract it and rename as DVWA.

Step 2: Open terminal and Install MySQL server by entering the following command:

```
apt-get install mysql-server
```

When prompted, create a password for MySQL.(Don’t forget your MySQL password)

Step 3: Install unzip, Apache web server, PHP5, PEAR, and the PHP5 MySQL module by entering the following command:

```
root@ubuntu:~# apt-get install unzip apache2 php5 php5-mysql php-pear
```

Step 4: Move into the /var/www/html directory by entering the following command:

This is the directory that Apache serves by default.

```
root@ubuntu:~# cd /var/www/html
```

Step 5: Copy the extracted DVWA and past it in location described below

```
root@ubuntu:~#/var/www/html
```

Step 6: Open the DVWA database connection script by entering the following command:

```
root@ubuntu:~#/var/www/html# nanodvwa/config/config.inc.php
```

Step 7: Now we need to add our MySQL password to the DVWA database connection script. To do this, find the following line:

$_DVWA['db_password'] = 'p@ssw0rd';

and replace “p@ssw0rd” with your MySQL password.

Example:

$_DVWA['db_password'] = '1234';

Step 8: Open the Apache php.ini file by entering the following command:

Step 9: Find the following line:

allow_url_include = Off

and replace “Off” with “On”

Step 10: Change the permissions by entering the following command:

```
root@ubuntu:~# chmod -R 777 /var/www/html/dvwa
```

Step 11: Log into MySQL and create a database for DVWA by entering the following commands:

```
root@ubuntu:~# mysql -u root -p

Enter Password: [your mysql password]

root@ubuntu:~# create database dvwa;

root@ubuntu:~# exit
```

Step 12: Open the apache.conf file by entering the following command:

```
root@ubuntu:~# nano /etc/apache2/apache2.conf
```

Step 13: Move to the bottom of the file and add the following line:

```
ServerName localhost
```

Step 14: Start the Apache web server by entering the following command:

```
root@ubuntu:~# service apache2 start
```

Step 15: Open a web browser and navigate to:

```
Localhost/DVWA
```

This completes the installation and configuration of the DVWA.

Installing and configuring ModSecurity in Ubuntu:

Before configuring ModSecurity we need to install some modules which make it easy to configure.

Requirements:

- ✓ Apache
- ✓ MySQL
- ✓ PHP

Installing Apache: Apache is a free open source software. Which can be installed easily.

```
sudo apt-get update

sudo apt-get install apache2
```

To check if Apache is installed, direct your browser to your server IP address. To do that start the apache server.

```
sudo start service apache2
```

Installing MySQL: To install MySQL, open terminal and type in these commands:

```
sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql
```

During the installation, MySQL will ask you to set a root password. If you miss the chance to set the password while the program is installing, it is very easy to set the password later from within the MySQL shell.

Once you have installed MySQL, we should activate it with this command:

```
sudomysql_install_db
```

Installing PHP: To install PHP, open terminal and type in this command.

```
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

After you answer yes to the prompt twice, PHP will install itself.

Once the required fields are installed successfully, now lets move on the modsecurity installation.

Installing ModSecurity: Modsecurity is available in the Debian/Ubuntu repository:

```
apt-get install libapache2-modsecurity
```

```
apachectl -M | grep --color security
```

Verify if the mod_security module was loaded.

You should see a module named security2_module (shared) which indicates that the module was loaded.

Modsecurity's installation includes a recommended configuration file which has to be renamed:

```
mv /etc/modsecurity/modsecurity.conf{-recommended,}
```

Reload Apache

```
service apache2 reload
```

You'll find a new log file for mod_security in the Apache log directory:

```
root@droplet:~# ls -l /var/log/apache2/modsec_audit.log
```

```
-rw-r----- 1 root root 0 Oct 19 08:08 /var/log/apache2/modsec_audit.log
```

```
nano /etc/modsecurity/modsecurity.conf
```

Find this line:**Configuring modsecurity:**

Out of the box, modsecurity doesn't do anything as it needs rules to work. The default configuration file is set to Detection Only which logs requests according to rule matches and doesn't block anything. This can be changed by editing the modsecurity.conf file:

```
SecRuleEngineDetectionOnly
```

and change it to:

```
SecRuleEngine On
```

If you're trying this out on a production server, change this directive only after testing all your rules.

Another directive to modify is SecResponseBodyAccess. This configures whether response bodies are buffered (i.e. read by modsecurity). This is only neccessary if data leakage detection and protection is required. Therefore, leaving it On will use up droplet resources and also increase the logfile size.

Find this

```
SecResponseBodyAccess On
```

and change it to:

```
SecResponseBodyAccess Off
```

Now we'll limit the maximum data that can be posted to your web application. Two directives configure these:

```
SecRequestBodyLimit

SecRequestBodyNoFilesLimit
```

The SecRequestBodyLimit directive specifies the maximum POST data size. If anything larger is sent by a client the server will respond with a 413 Request Entity Too Large error. If your web application doesn't have any file uploads this value can be greatly reduced.

The value mentioned in the configuration file is

```
SecRequestBodyLimit 13107200
```

which is 12.5MB.

Similar to this is the SecRequestBodyNoFilesLimit directive. The only difference is that this directive limits the size of POST data minus file uploads-- this value should be "as low as practical."

The value in the configuration file is

```
SecRequestBodyNoFilesLimit 131072
```

which is 128KB.

Along the lines of these directives is another one which affects server performance: SecRequestBodyInMemoryLimit. This directive is pretty much self-explanatory; it specifies how much of "request body" data (POSTed data) should be kept in the memory (RAM), anything more will be placed in the hard disk (just like swapping). Since droplets use SSDs, this is not much of an issue; however, this can be set a decent value if you have RAM to spare.

```
SecRequestBodyInMemoryLimit 131072
```

This is the value (128KB) specified in the configuration file.

Setting Up Rules:

To make your life easier, there are a lot of rules which are already installed along with mod_security. These are called CRS (Core Rule Set) and are located in

```
root@droplet:~# ls -l /usr/share/modsecurity-crs/
```

The documentation is available at

```
root@droplet1:~# ls -l /usr/share/doc/modsecurity-crs/
```

To load these rules, we need to tell Apache to look into these directories. Edit the modsecurity.conf file.

```
nano /etc/apache2/mods-enabled/modsecurity.conf
```

Add the following directives inside <IfModule security2_module></IfModule>:

```
Include "/usr/share/modsecurity-crs/*.conf"

Include "/usr/share/modsecurity-crs/activated_rules/*.conf"
```

The activated_rules directory is similar to Apache's mods-enabled directory. The rules are available in directories:

```
/usr/share/modsecurity-crs/base_rules

/usr/share/modsecurity-crs/optional_rules

/usr/share/modsecurity-crs/experimental_rules
```

Symlinks must be created inside the activated_rules directory to activate these. Let us activate the SQL injection rules.

```
cd /usr/share/modsecurity-crs/activated_rules/ln -s /usr/share/modsecurity-
crs/base_rules/modsecurity_crs_41_sql_injection_attacks.conf
```

Apache has to be reloaded for the rules to take effect.

```
service apache2 reload
```

This completes the ModSecurity configuration.

Now lets test the ModSecurity WAF on DVWA as we have already installed it.

Lets test this with three popular kinds of attacks: SQL injection, Cross-Site Scripting and Brute Force.

6.SYSTEM TESTING

SYSTEM TESTING:-

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

6.1 Unit testing:-

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Unit testing 8r is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach: Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

6.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components

6.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is cantered on the following items:

➢	Valid Input	: identified classes of valid input must be accepted.
➢	Invalid Input	: identified classes of invalid input must be rejected.
➢	Functions	: identified functions must be exercised.
➢	Output	: identified classes of application outputs must be exercised.
➢	Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.4 **System Test**:-

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.5 White Box Testing:-

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level**.**

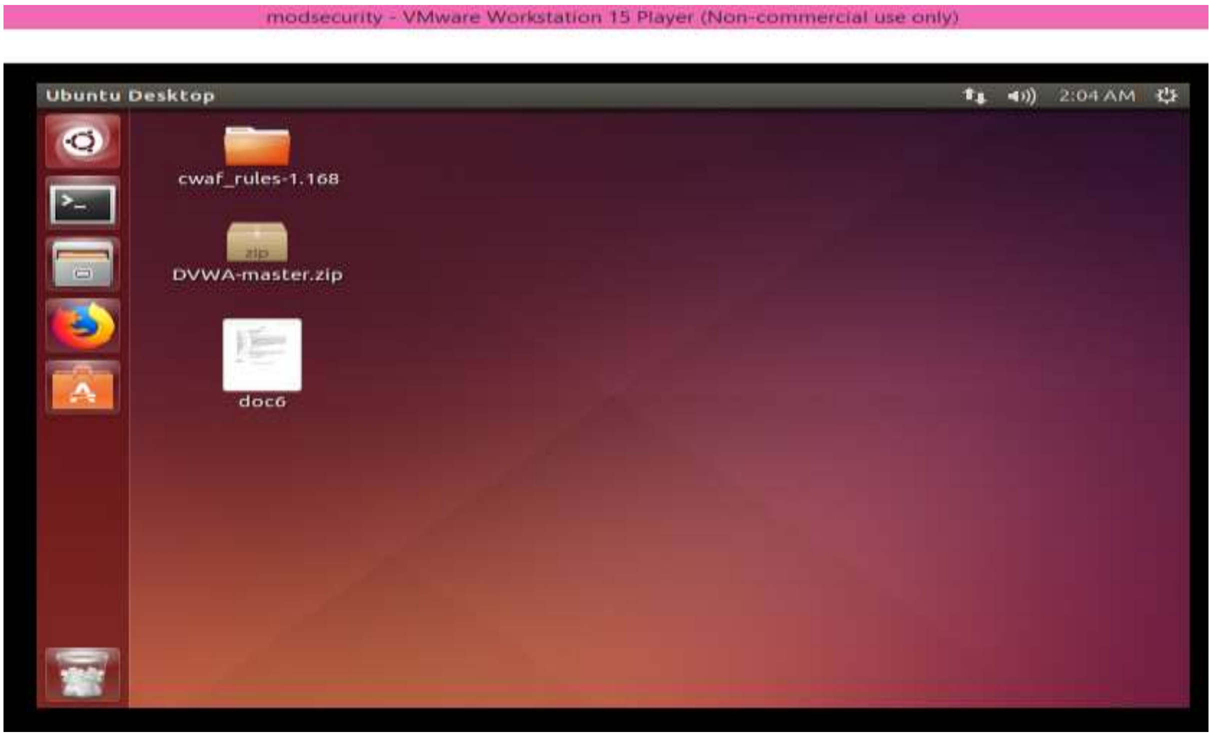
6.6 Black Box Testing:-

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

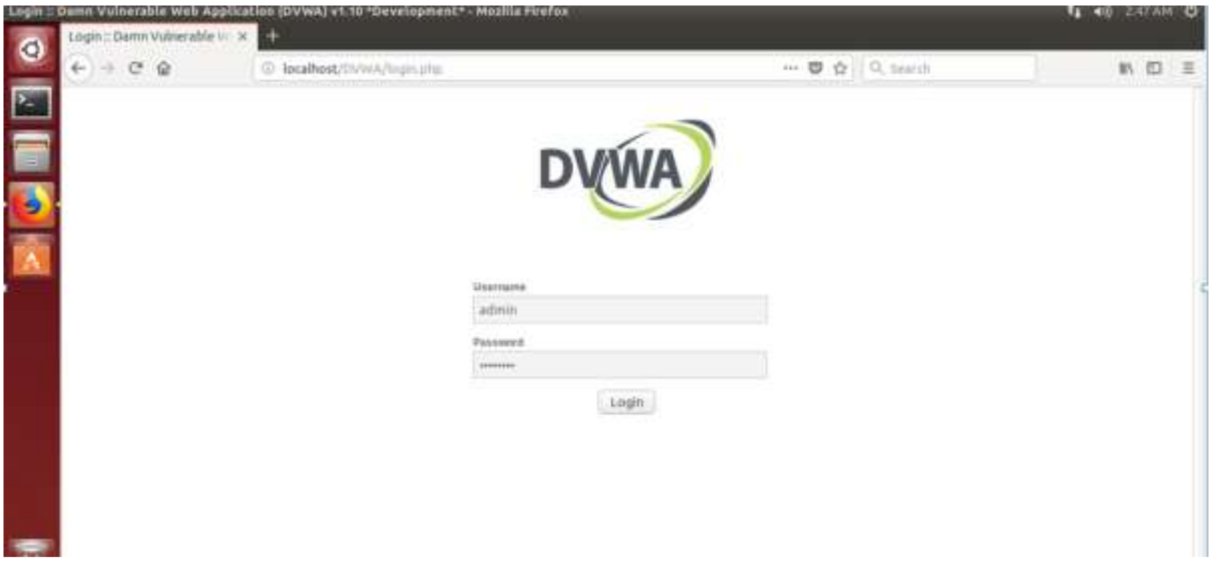
7.SCREENSHOTS

Ubuntu Operating System:-

Open the VMware WorkStation play the VMware workstation then the Ubuntu Operating System is displayed as shown in the following Screenshot.


modsecurity - VMware Workstation 15 Player (Non-commercial use only)
Ubuntu Desktop
2:04 AM
cwaf_rules-1.168
DVWA-master.zip
doc6

Open Mozilla Firefox to host the application type localhost/DVWA in address bar then the DVWA Login page will be displayed as shown enter username and password as shown in the below Screenshot


Login :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* - Mozilla Firefox
Login :: Damn Vulnerable W
DVWA
Login

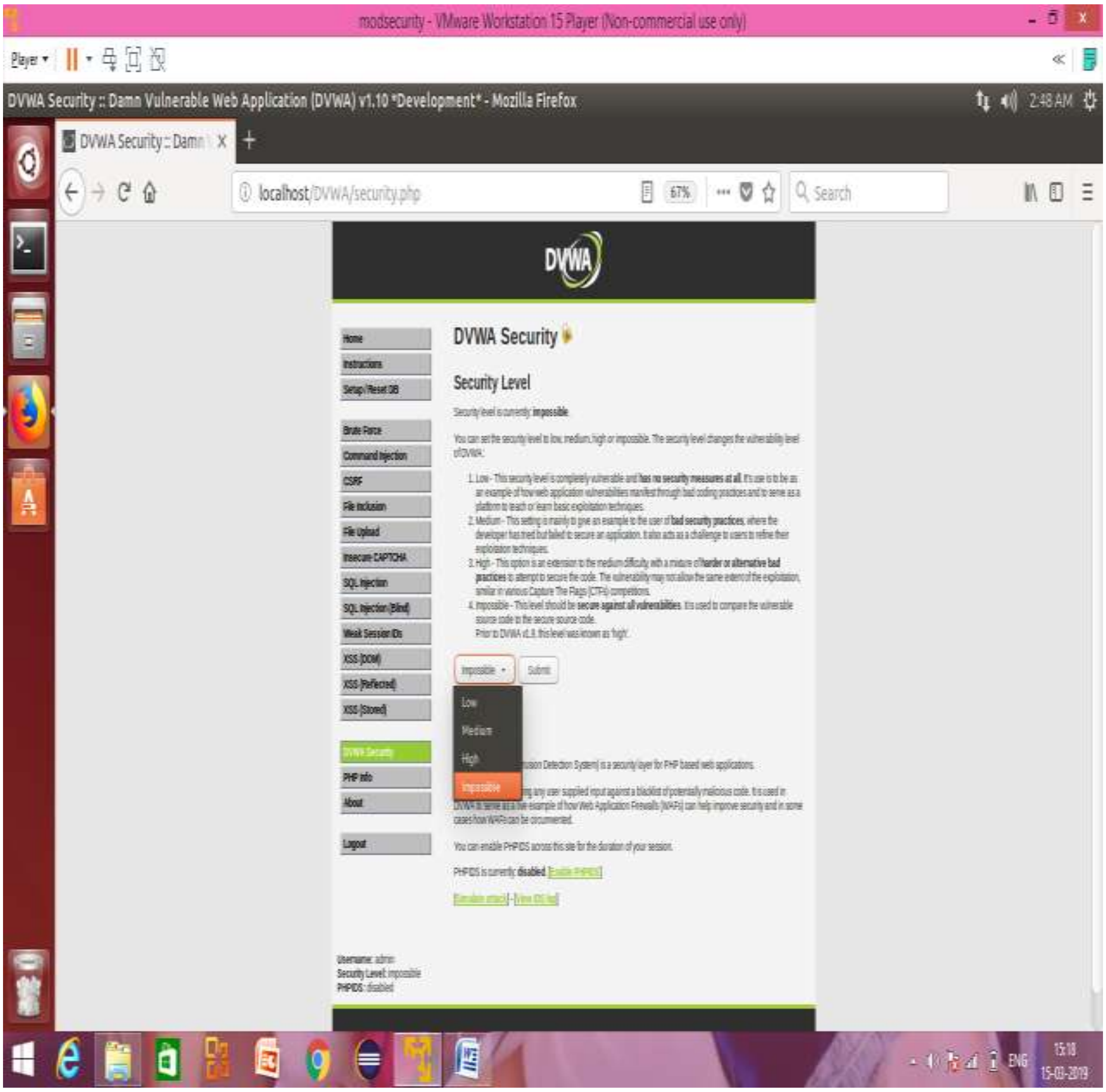
localhost/DVWA/login.php

Username
admin

Password
••••••••

Fig:Login Page

The Overall Application will be displayed as follows in the following Screenshot.Only Admin has the right to provide the security level to the application. Security can be provided to the application by selecting the module DVWA Security then the corresponding action takes place at right side. We have a chance to change the security level of DVWA by selecting options “Low”, “Medium”, ”High” displayed in below screenshot



DVWA Security

Security Level

Security level is currently: **impossible**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

[illegible] sion Detection System) is a security layer for PHP based web applications.

[illegible] ng any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently: **disabled**.

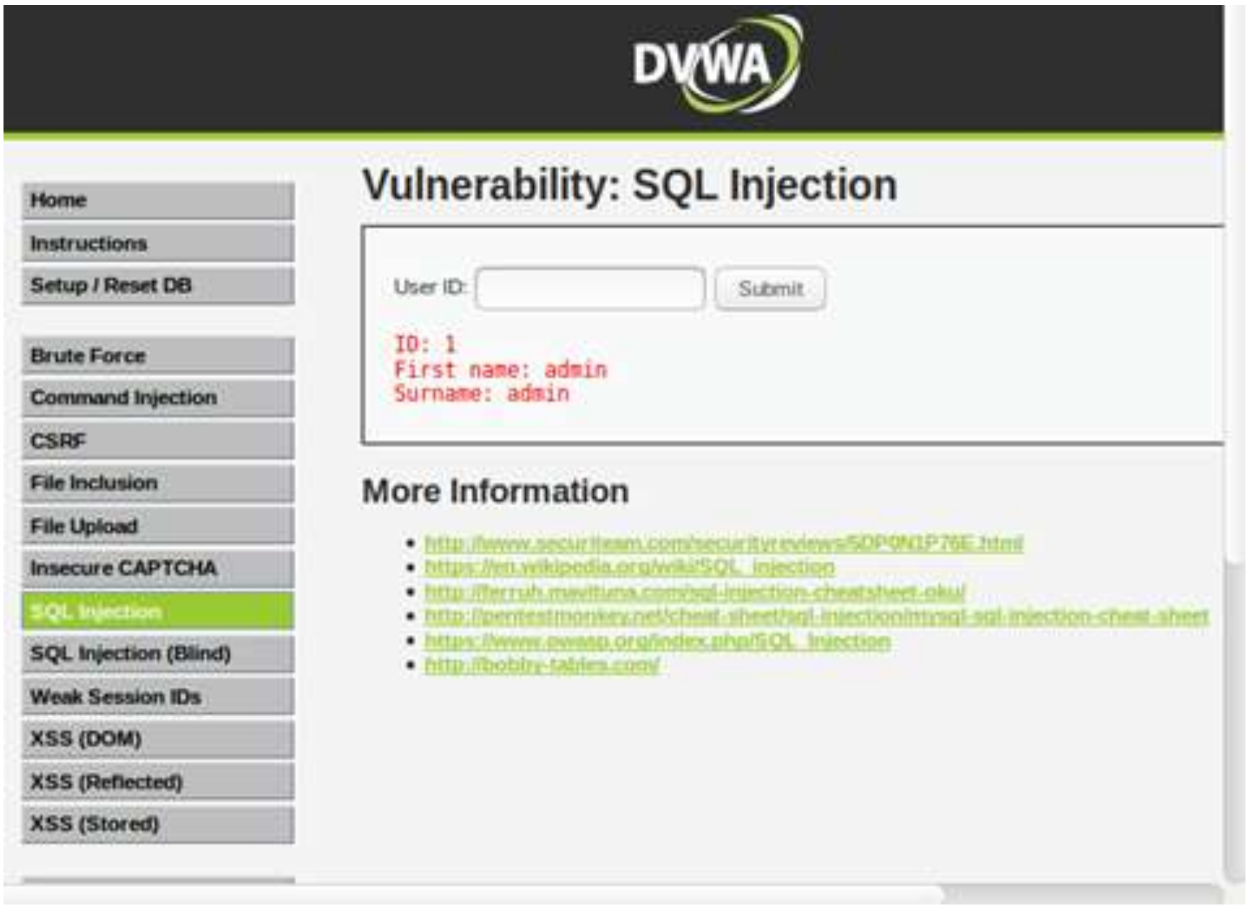
Username: admin
Security Level: impossible
PHPIDS: disabled

Fig:DVWA Application

SQL Injection:-

It might be possible that the web application server has more than one vulnerabilities, let assume if it is also having SQL injection vulnerability then it becomes very easy for an attacker to retrieve the data from its database using stolen cookies.

For example in DVWA I switch from XSS to SQL injection; now copy its URL with user ID=1



Vulnerability: SQL Injection

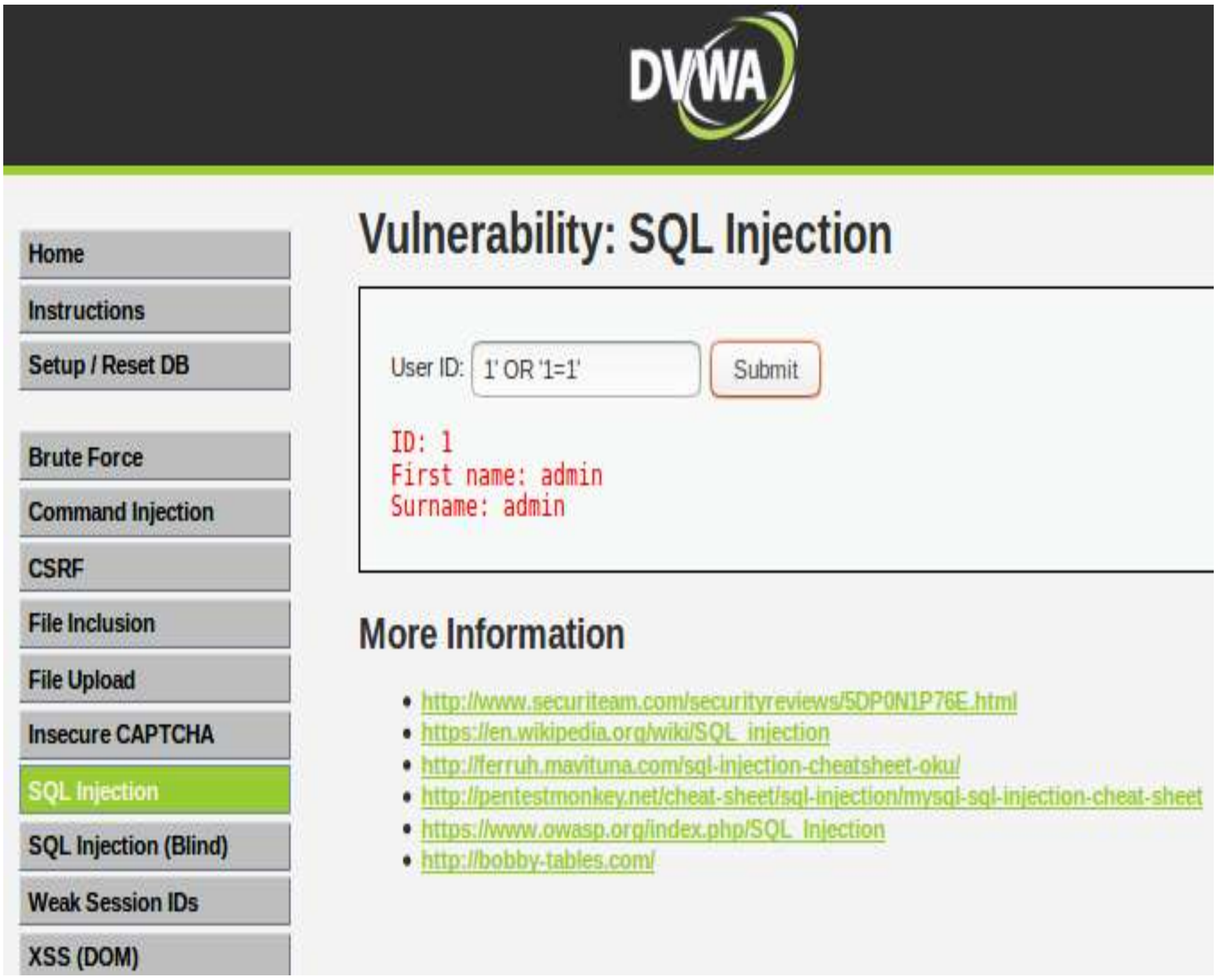
```
ID: 1
First name: admin
Surname: admin
```

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

Fig: Retrieving Information From Database

When the Attacker wants to create damage to the Organization he will drop some conditions to the SQL Query then he will enter into the database. How the Attacker will retrieve the Information by providing wrong script will be displayed in the below Screenshot.



Vulnerability: SQL Injection

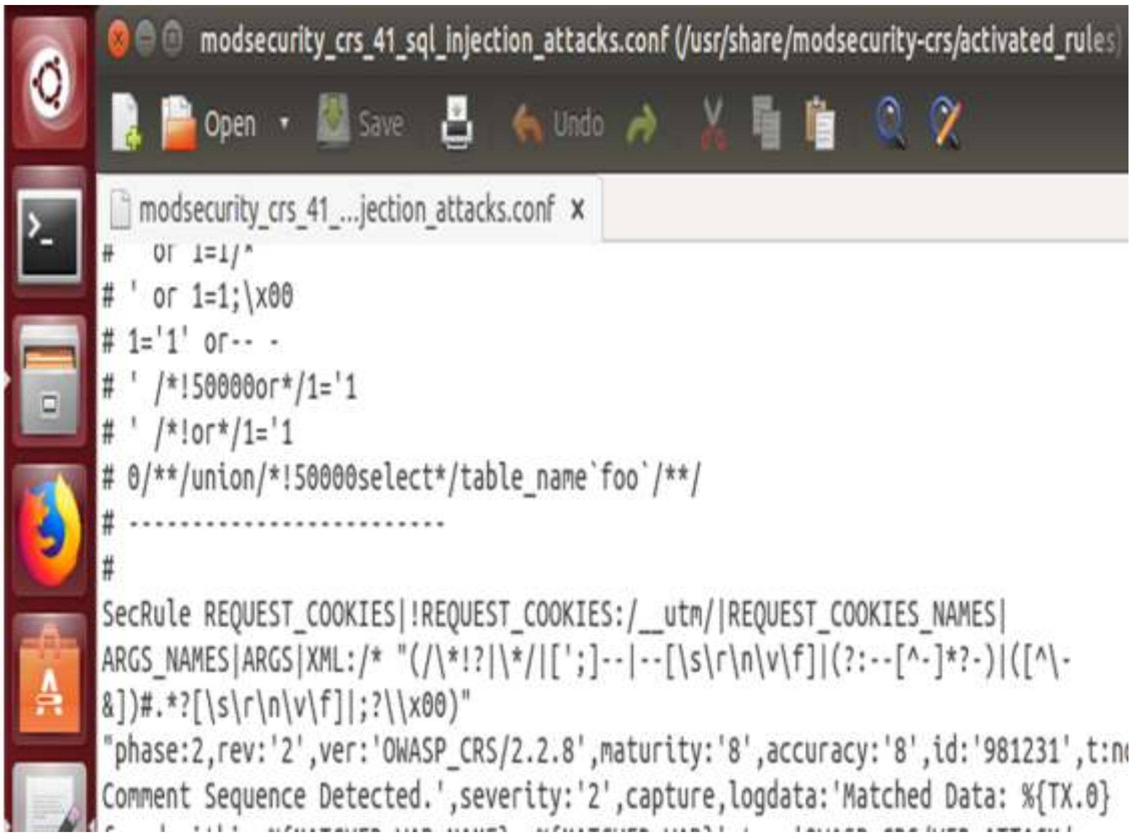
```
ID: 1
First name: admin
Surname: admin
```

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

Fig:-Finding Vulnerability In SQL Injection

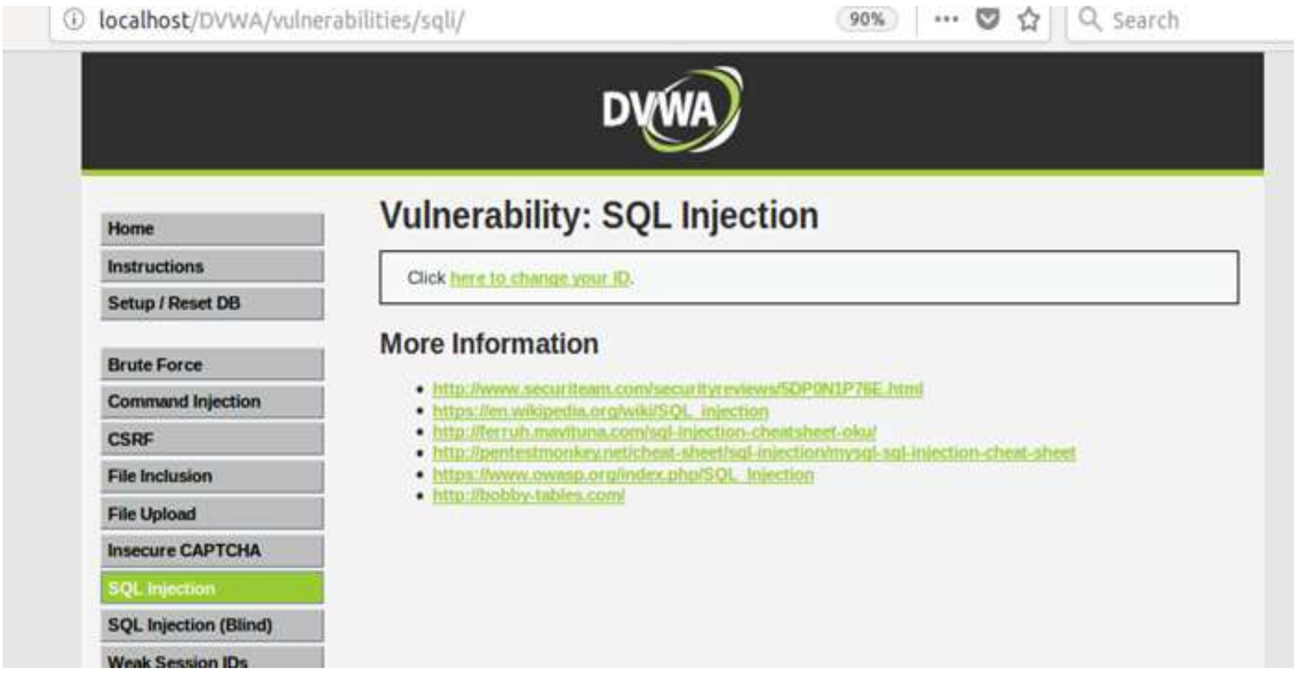
To Overcome this problem we will use Open Source Web Application Modesecurity. Using Modsecurity rule writing the developer has a chance to overcome this problem by providing rules to the web application.


modsecurity_crs_41_sql_injection_attacks.conf (/usr/share/modsecurity-crs/activated_rules)
Open
Save
Undo
modsecurity_crs_41_...jection_attacks.conf

```
#   or 1=1/*
# ' or 1=1;\x00
# 1='1' or-- -
# ' /*!50000or*/1='1
# ' /*!or*/1='1
# 0/**/union/*!50000select*/table_name`foo`/**/
# -----------------------
#
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|REQUEST_COOKIES_NAMES|
ARGS_NAMES|ARGS|XML:/* "(/\*!?|\*/|[';]--|--[\s\r\n\v\f]|(?:--[^-]*?-)|([^\-
&])#.*?[\s\r\n\v\f]|;?\\x00)"
"phase:2,rev:'2',ver:'OWASP_CRS/2.2.8',maturity:'8',accuracy:'8',id:'981231',t:n
Comment Sequence Detected.',severity:'2',capture,logdata:'Matched Data: %{TX.0}
```

Fig:-ModSecurity Rule for SQL Injection

When DVWA Security is set at "high" the corresponding rules will be executed.



Vulnerability: SQL Injection

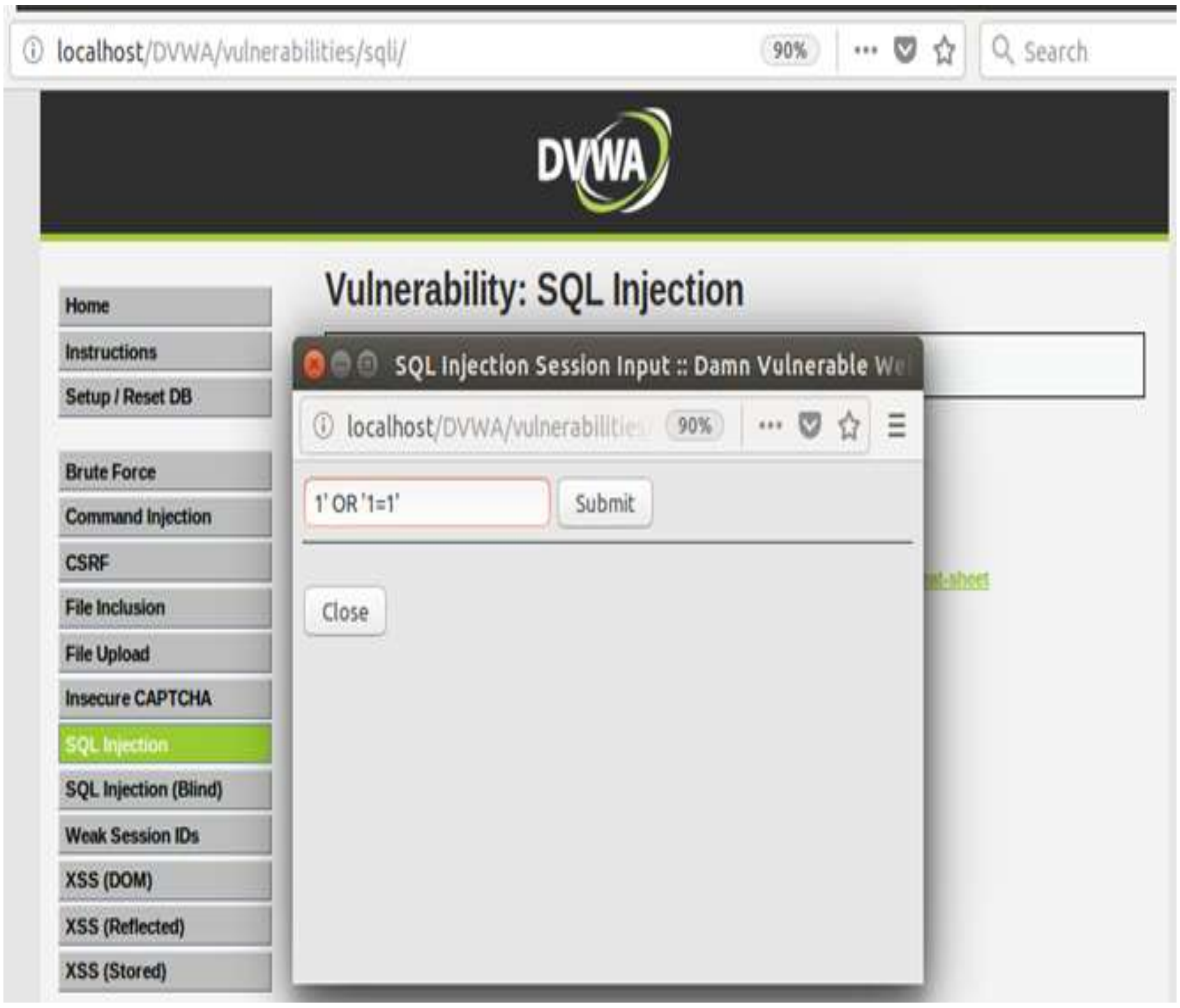
Click here to change your ID.

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

Fig:-DVWA Application when DVWA Security is at "High"

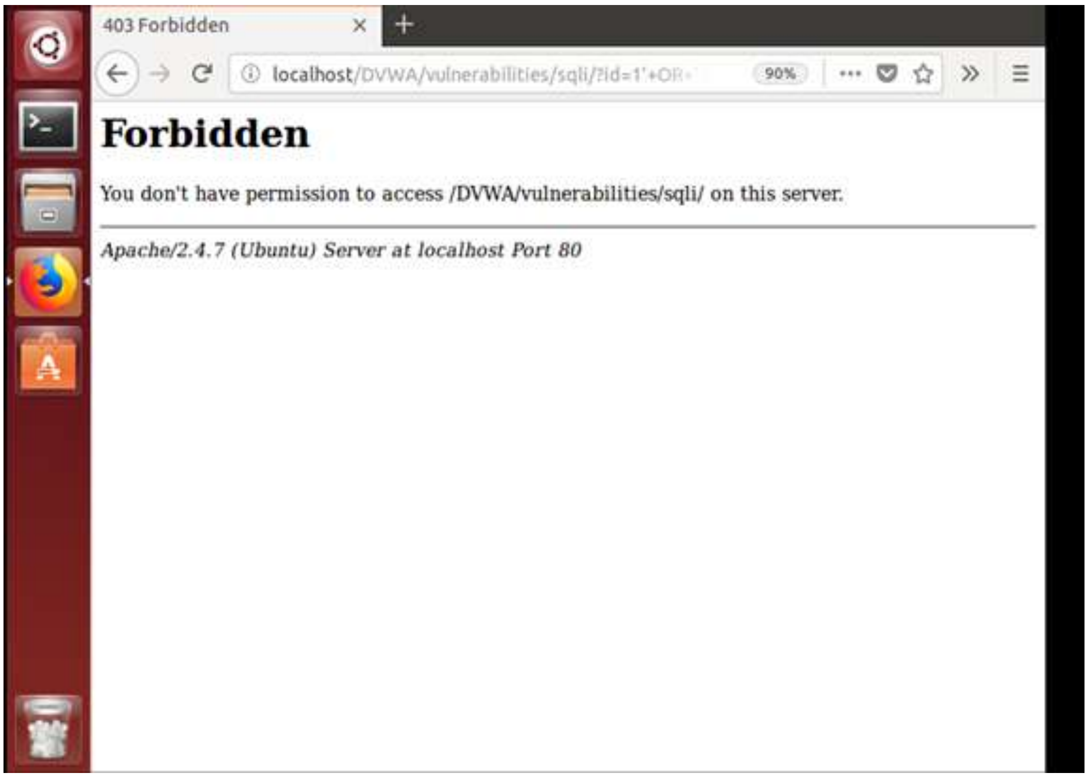
Enter the user id in the following manner as shown in the screenshot.



1' OR '1=1'

Fig:-Blocking The SQL Injection

After the submitting the text the following error will be displayed as follows



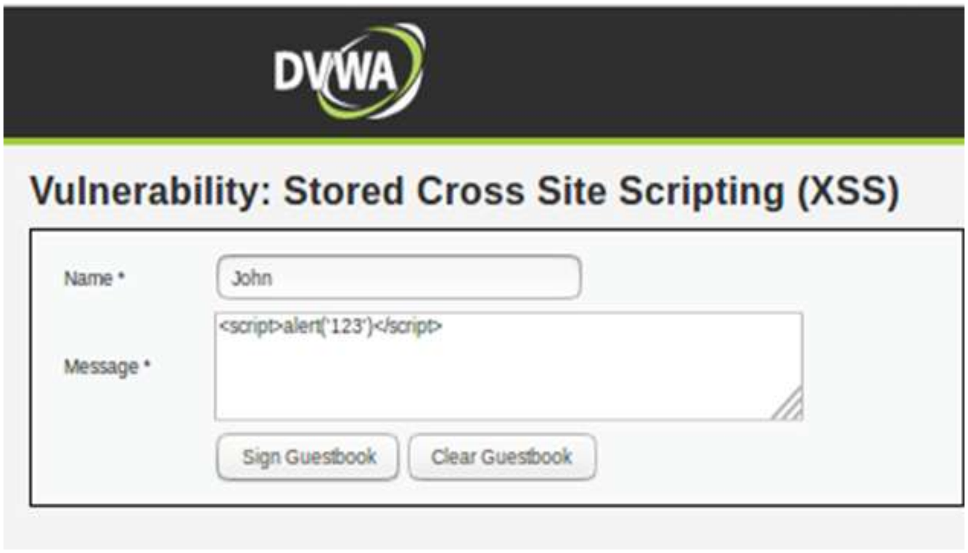
Forbidden

You don't have permission to access /DVWA/vulnerabilities/sqli/ on this server.

Apache/2.4.7 (Ubuntu) Server at localhost Port 80

Fig:-Blocking the Attacker From SQL Injection

Cross Site Scripting(XSS) Attack:- XSS generally occurs when user input is stored on the target server, such as in a database, in a message forum, visitor log, comment field, etc. And then a victim is able to retrieve the stored data from the web application without that data being made safe to render in the browser. With the advent of HTML5 and other browser technologies, we can envision the attack payload being permanently stored in the victim's browser, such as an HTML5 database, and never being sent to the server at all.



Vulnerability: Stored Cross Site Scripting (XSS)

Name * John

Message * <script>alert('123')</script>

Fig:-Cross Site Scripting(XSS)

Inject the script which gets stored in the server. Now when user will visit this page to read our message his browser will execute our script which generates an alert prompt as showing the following screenshot.

This was a small demo to show how to inject any script if the server is suffering from XSS and further you will learn what else an attacker can do to cause damage inside a web application server.

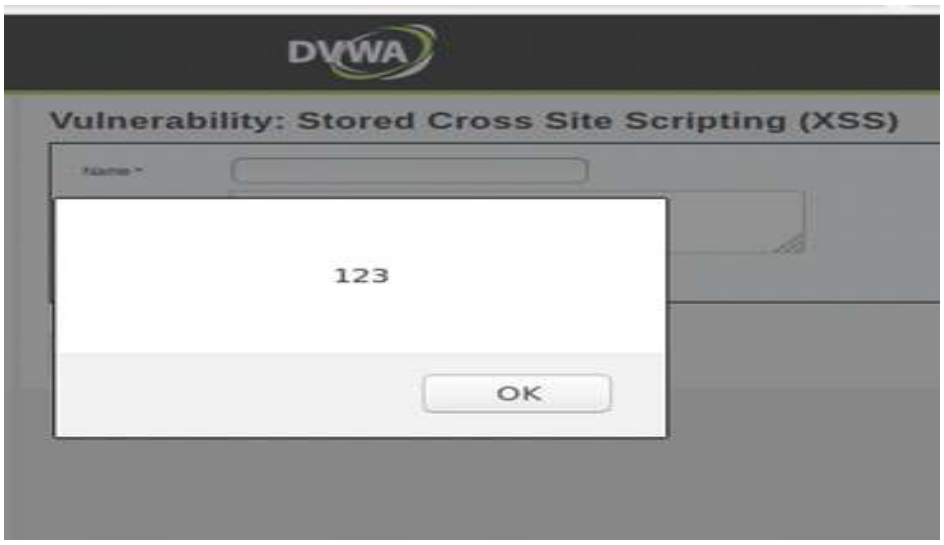


Fig:-Vulnerability in XSS Attack

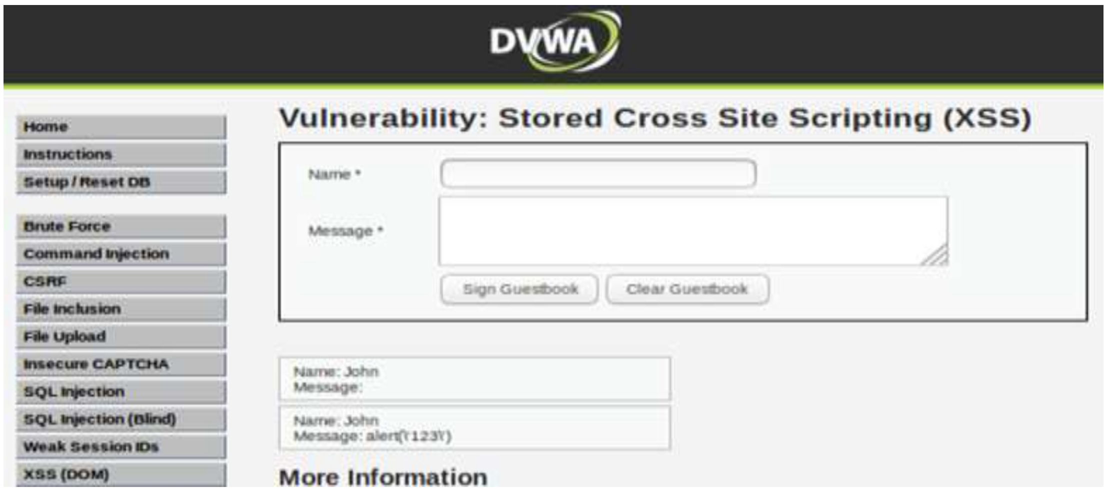
If the attack is aware that the web server is having XSS then he might think to

steal the web cookies which contain session Id, therefore, he will generate a script to fetch running cookies.Rule to block XSS Attack as shown in the below rectangular box.

```
SecRule ARGS
"(?i)(<script[^>]*>[\s\S]*?<\/script[^>]*>|<script[^>]*>[\s\S]*?<\/script[[\s\S]]*[\s\S]|<script
[^>]*>[\s\S]*?<\/script[\s]*[\s]|<script[^>]*>[\s\S]*?<\/script|<script[^>]*>[\s\S]*?)"
"id:'973336',phase:2,rev:'1',ver:'OWASP_CRS/2.2.8',accuracy:'8',msg:'XSS Filter - Category 1:
Script Tag Vector',tag:'OWASP_CRS/WEB_ATTACK/XSS',tag:'OWASP_TOP_10/A2',severity:'2'
```

Fig: ModSecurity Rule For XSS Attack

After Applying the rule the script tags are not executed only the text messages will send to the receiver.



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: John
Message:

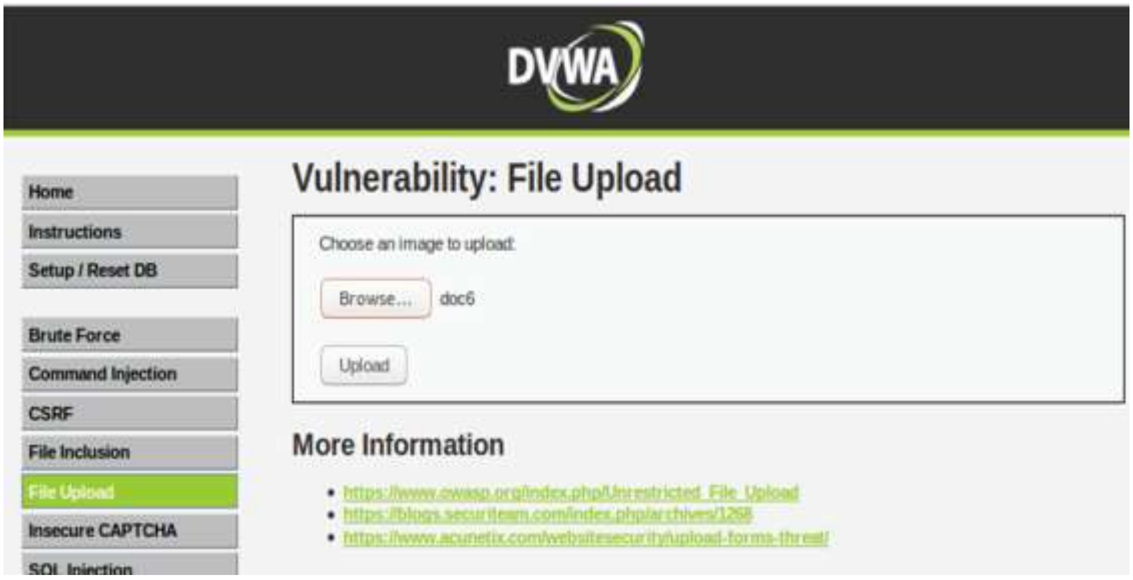
Name: John
Message: alert(\'123\')

More Information

Fig:-Blocking XSS Attack

File Uploading Attack:-

Now let assume if the server is suffering from XSS as well as file uploading both vulnerabilities; in this case how an attacker would be able to cause harm to the web application server.



Vulnerability: File Upload

Choose an image to upload:

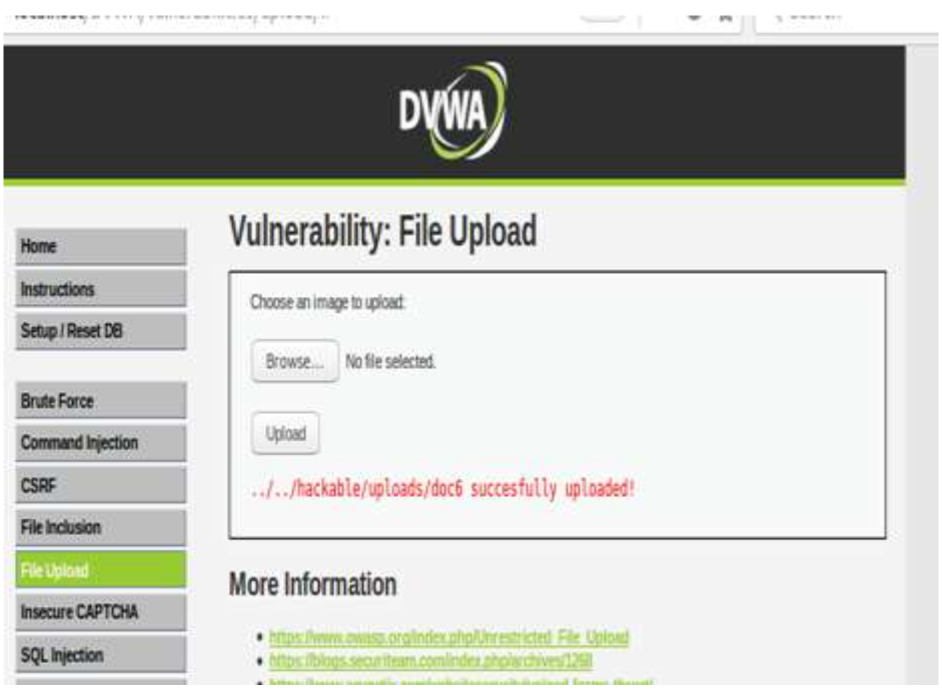
doc6

More Information

- https://www.owasp.org/index.php/Unrestricted_File_Upload
- https://blogs.securiteam.com/index.php/archives/1268
- https://www.acunetix.com/websitesecurity/upload-forms-threat/

Fig: File Uploading in DVWA

Again I switched to file uploading vulnerability in DVWA to **upload shell.php** and from the screenshot, you can see our shell.php file is successfully uploaded now **copy** the highlighted path



Vulnerability: File Upload

Choose an image to upload:

No file selected.

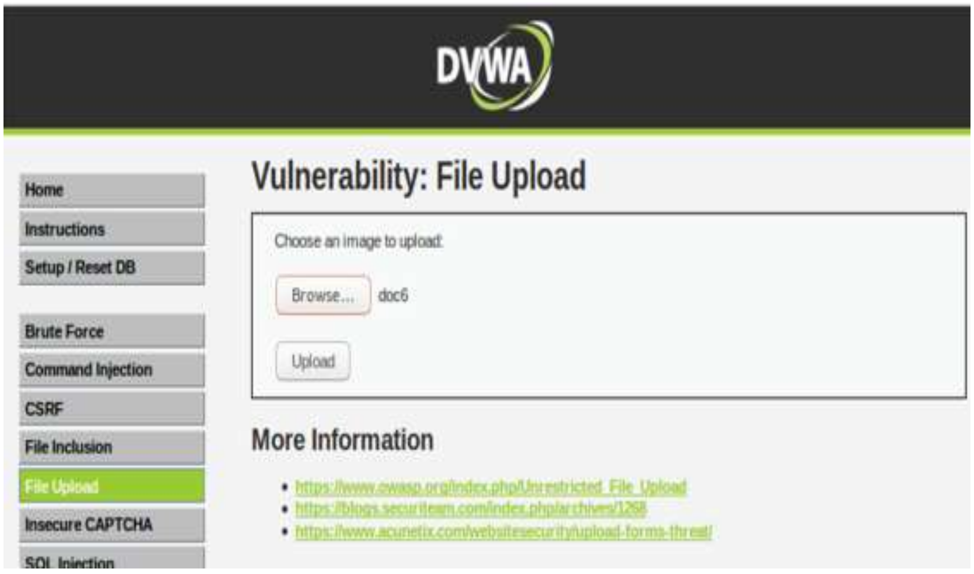
```
../../hackable/uploads/doc6 succesfully uploaded!
```

More Information

- https://www.owasp.org/index.php/Unrestricted_File_Upload
- https://blogs.securiteam.com/index.php/archives/1268

Fig:-Vulnerability in File Uploading

After placing the DVWA security at high the corresponding rule will be executed then the page will be displayed as follows



Vulnerability: File Upload

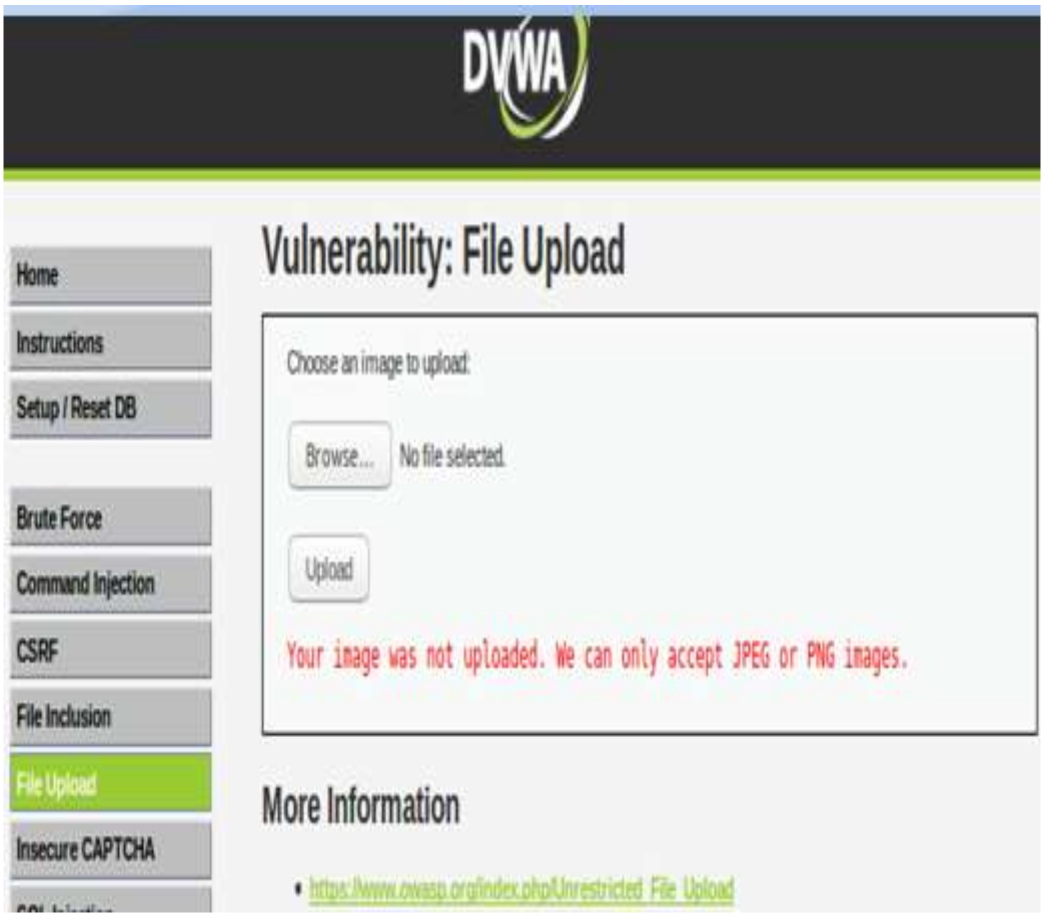
Choose an image to upload:

doc6

More Information

- https://www.owasp.org/index.php/Unrestricted_File_Upload
- https://blogs.securiteam.com/index.php/archives/1268
- https://www.acunetix.com/websitesecurity/upload-forms-threat/

Fig: File Uploading When DVWA Security is At High



Vulnerability: File Upload

Choose an image to upload:

No file selected.

Your image was not uploaded. We can only accept JPEG or PNG images.

More Information

- https://www.owasp.org/index.php/Unrestricted_File_Upload

Fig: Blocking File Uploading Attack

8.CONCLUSION

Web applications have been evolving so fast and they have become one of the most important things that we can’t live without, such as electricity and water. The use of web applications will not stop increasing but, on the other hand, attackers will not stop trying to penetrate your applications to. Implementing a web application firewall is a great method to protect your application from web attacks. However, the cost and the complexity of implementing a WAF are huge. If you are new to the WAF technology, you should start with an open source technology, such as Modsecurity, to learn the technology. Then, as a next step, you can test your small application. Once you are confident with the technology, you can start implementing it for your main application in order to protect it.

9.BIBLOGRAPHY

- https://www.sans.org/reading-room/whitepapers/application/web-application-firewalls-35817
- https://www.nginx.com/blog/modsecurity-waf-released/
- Damele, B., & Stampar, M. (2011). *Sqlmaps's user manual* . Retrieved from http://sqlmap.sourceforge.net/doc/README.pdf
- Hansen, R. (2008). Xss cheat sheet. Retrieved from http://ha.ckers.org/xss.html
- HP DVlabs, (2010). *2010 full year top cyber security risks report* .
- Retrieved from http://dvlabs.tippingpoint.com/img/FullYear2010 Ris k Report.pdf
- Ivey, T. (2010). *Damn vulnerable web application official documentation* .
- Retrieved from
- https://dvwa.svn.sourceforge.net/svnroot/dvwa/docs/DVWA_v1.3. pdf
- Mischel, M(2009). *Modsecurity 2.5* . (1st ed.). Birmingham, UK: Packt Publishing Ltd.
- Modsecurity, (2011). *Modsecurity reference manual* . Trustwave Holdings, Inc.Retrieved from http://sourceforge.net/apps/mediawiki/mod-
- security/index.php?title=Reference_Manual
- Owasp, (2011). *Web application firewall* . Retrieved from
- https://www.owasp.org/index.php/Web_Application_Firewall
- Phongthiproek, P. (2011). *Beyond sqli: Obfuscate and bypass* . Retrieved from http://www.exploit- db.com/papers/17934/
- Stuttard, D, & Pinto, M(2007). *The web application hacker's handbook* . (1 ed.). Indianapolis, IN: Wiley& Sons Publishing.
- Security Compass. (2010). Retrieved from https://addons.mozilla.org/en- US/firefox/addon/xss-me/

- Trustwave, (2011). *New modsecurity release includes key data protection advancements* . Retrieved from
- https://www.trustwave.com/pressReleases.php? n=new-
- modsecurity-release-includes-key-data-protection-advancements
-
- Vela, E., & Lindsay, D. (2009). *Our favorite xss filters/ids* . Retrieved from http://www.blackhat.com/presentations/bh-usa-
- 09/VELANAVA/BHUSA09-VelaNava-FavoriteXSS-SLIDES.pdf