

5. SYSTEM DESIGN

5.1 SYSTEM APPLICATION DESIGN

5.1.1 Method Pseudo code

Registration

- 1 Include connection.php for database connection
- 2 Fill all the fields given in registration form.
- 3 Check for valid inputs
- 4 if inputs are valid send a confirmation email
- 5 show message-you are successfully registered.

Login

- 1 Include connection.php for database connection
- 2 Provide userid and password
- 3 Check for null inputs and invalid inputs
- 4 Match userid and password from database
- 5 If matched Login Successful
- 6 Else show error message and go to step 2

Customer Id Allocation

- 1 Include connection.php for database connection
- 2 Select the client
- 3 Enter a customer id
- 4 Select the user id from the drop down menu
- 5 show message-Customer Id successfully allocated.

Add Updates

- 1 Include connection.php for database connection
- 2 Fill all the fields given in client's details form.
- 3 Check for valid inputs
- 4 If inputs are valid add the details on the home page.
- 5 show message-Details successfully added.

Add Rules Signature

- 1 Include connection.php for database connection
- 2 Enter a customer id
- 3 Fill all the fields given in client's rule configuration form.
- 4 Check for valid rules signature
- 5 If inputs are valid add the configuration details on WAF box.
- 6 show message-Details successfully added.

5.2 DESIGN METHODOLOGY

For this project, I am using the function oriented design approach. The goal of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language. In technical terms, during the design phase the software architecture is derived from the SRS document. Design is an important and essential phase of software development and can be rightly considered as the backbone or base of the implementation phase. It rather acts as a raw material for the manufacturing of the implementation in a short way.

Today, two distinctly different approaches are available: the traditional design approach and the object-oriented design approach.

➤ **Traditional design approach**

Traditional design consists of two different activities; first a structured analysis of the requirements specification is carried out where the detailed structure of the problem is examined. This is followed by a structured design activity. During structured design, the results of structured analysis are transformed into the software design.

➤ **Object-oriented design approach**

In this technique, various objects that occur in the problem domain and the solution domain are first identified, and the different relationships that exist among these objects are identified. The object structure is further refined to obtain the detailed design.

In the real sense, Software design deals with transforming the customer requirements, as Described in the SRS document, into a form (a set of documents) that is suitable for implementation in a programming language. A good software design is seldom arrived by using a single step procedure but rather through several iterations through a series of steps. Design activities can be broadly classified into two important parts:

- Preliminary (or high-level) design and
- Detailed design

The meaning and scope of two design activities (i.e. high-level and detailed design) tend to vary considerably from one methodology to another. High-level design means identification of different modules and the control relationships among them and the definition of the interfaces among these modules. The outcome of high-level design is called the program structure or software architecture. Many different types of notations have been used to represent a high-level design. A popular way is to use a tree-like diagram called the structure chart to represent the control hierarchy in a high-level design. During detailed design, the data structure and the algorithms of the different modules are designed. The outcome of the detailed design stage is usually known as the module-specification document.

The end result of design is to produce a model that will provide a seamless transition to the coding phase, i.e. once the requirements are analyzed and found to be satisfactory, a design model is created which can be easily implemented.

In the function oriented design approach that is used for this project, the system is viewed as something that performs a set of functions. Starting at this high-level view of the system, each function is successively refined into more detailed functions.

Also, the system state is centralized and shared among different functions,

5.1.1 Function oriented design

Function oriented design involves the structured design of the system under consideration on the basis of structured analysis of requirements carried out in the system design earlier i.e. the creation of Data flow diagrams.

The aim of structured design is to transform the results of the structured analysis (i.e. a DFD representation) into a structure chart. Structured design provides two strategies to guide transformation of a DFD into a structure chart.

- Transform analysis
- Transaction analysis

Normally, the level 1 DFD is transformed into a module representation using either the transform or the transaction analysis and then further towards the lower-level DFDs. At each level of transformation, it is important to first determine whether the transform or the transaction analysis is applicable to a particular DFD.

5.1.2 Structure Chart

A structure chart represents the software architecture, i.e. the various modules making up the system, the dependency (which module calls which other modules), and the parameters that are passed among the different modules. Hence, the structure chart representation can be easily implemented using some programming language. Since the main focus in a structure chart representation is on the module structure of the software and the interactions among different modules, the procedural aspects (e.g. how a particular functionality is achieved) are not represented.

The basic building blocks which are used to design structure charts are the following:

- **Rectangular boxes:** Represents a module.
- **Module invocation arrows:** Control is passed from one module to another module in the direction of the connecting arrow.
- **Data flow arrows:** Arrows are annotated with data name; named data passes from one module to another module in the direction of the arrow.
- **Library modules:** Represented by a rectangle with double edges.
- **Selection:** Represented by a diamond symbol.
- **Repetition:** Represented by a loop around the control flow arrow.

For this project, it is essential to employ transform analysis as we have similar processing steps for the data items in consideration as emphasized by the functional requirements of the project.

5.1.3 Transform Analysis

Transform analysis identifies the primary functional components (modules) and the high level inputs and outputs for these components. The first step in transform analysis is to divide the DFD into 3 types of parts:

- Input
- Logical processing
- Output

The input portion of the DFD includes processes that transform input data from physical (e.g. character from terminal) to logical forms (e.g. internal tables, lists, etc.). Each input portion is called an afferent branch.

The output portion of a DFD transforms output data from logical to physical form. Each output portion is called an efferent branch. The remaining portion of a DFD is called the central transform.

In the next step of transform analysis, the structure chart is derived by drawing one functional component for the central transform, and the afferent and efferent branches.

5.3 STRUCTURE CHART

The structure charts using the transform analysis are shown below:

5.3.1 Login

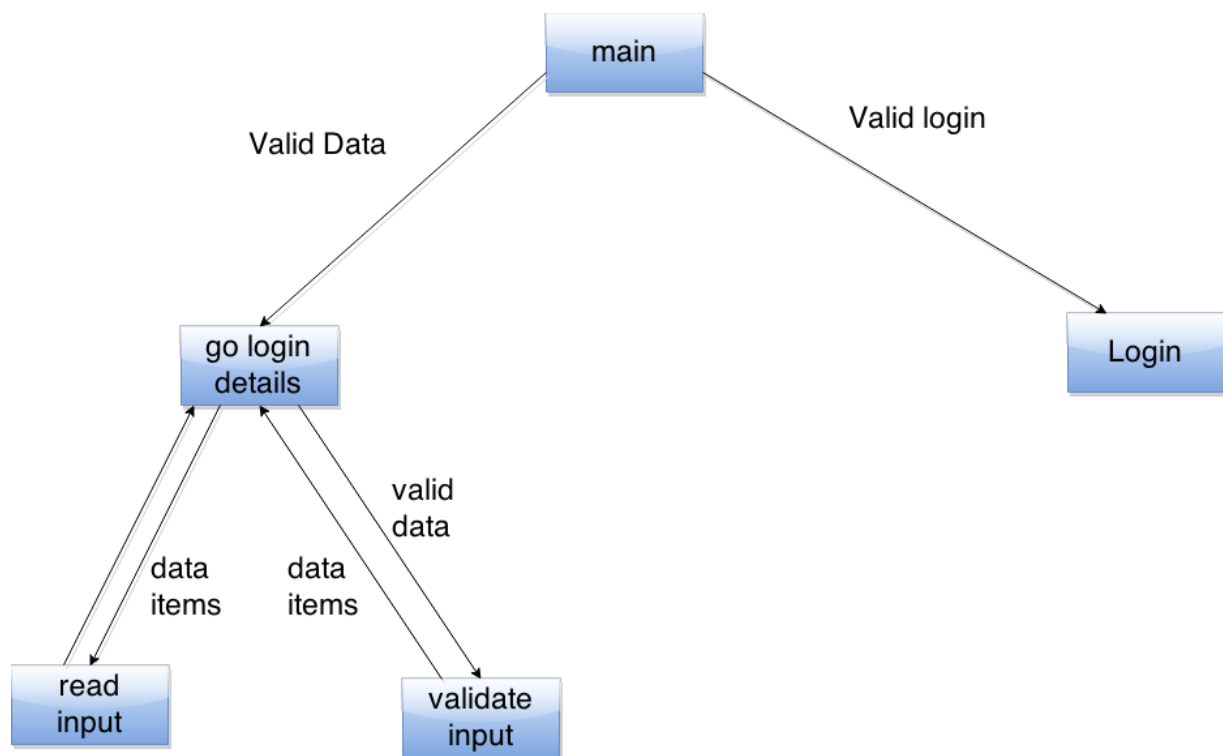


Fig 5.3.1 Login

5.3.2 Client Details Manipulation

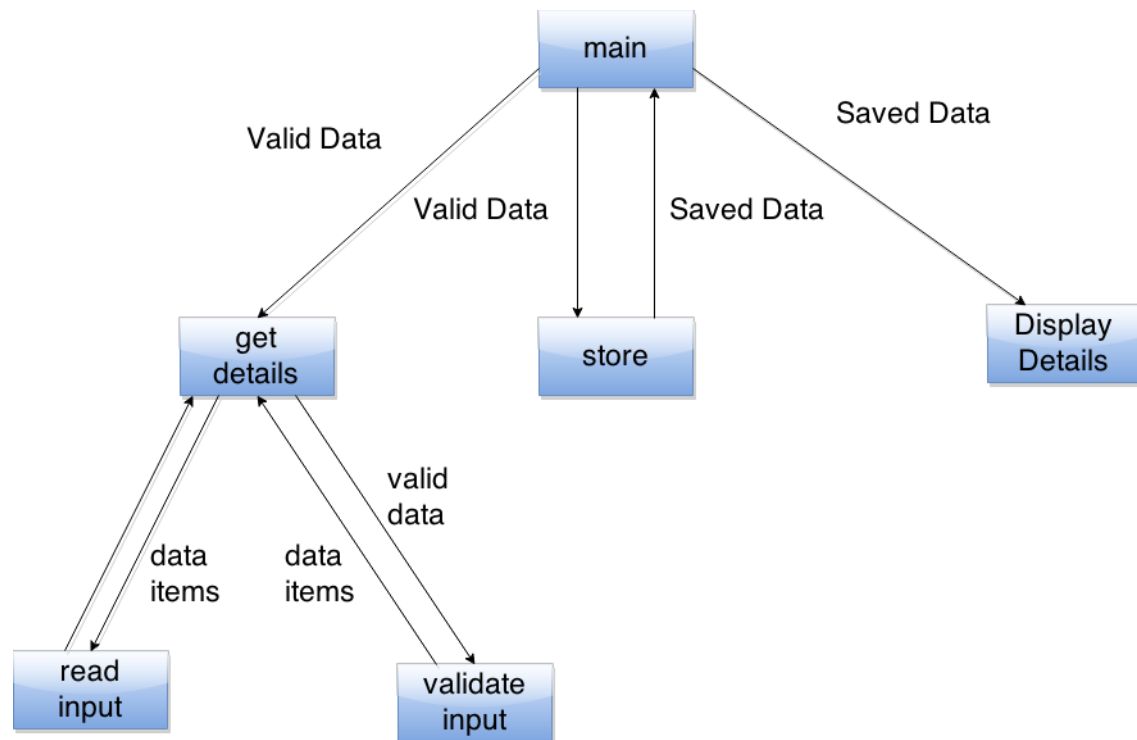


Fig 5.3.2 Client Details Manipulation

5.3.3 Scanning report of Clients URL

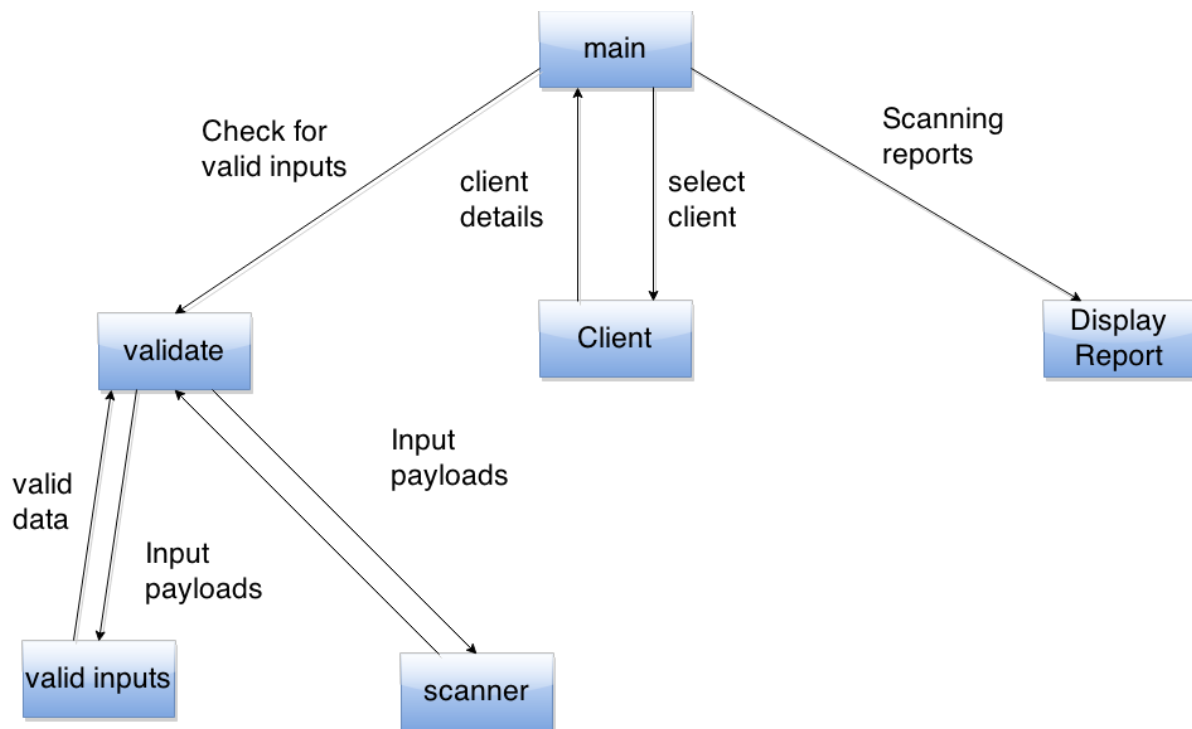
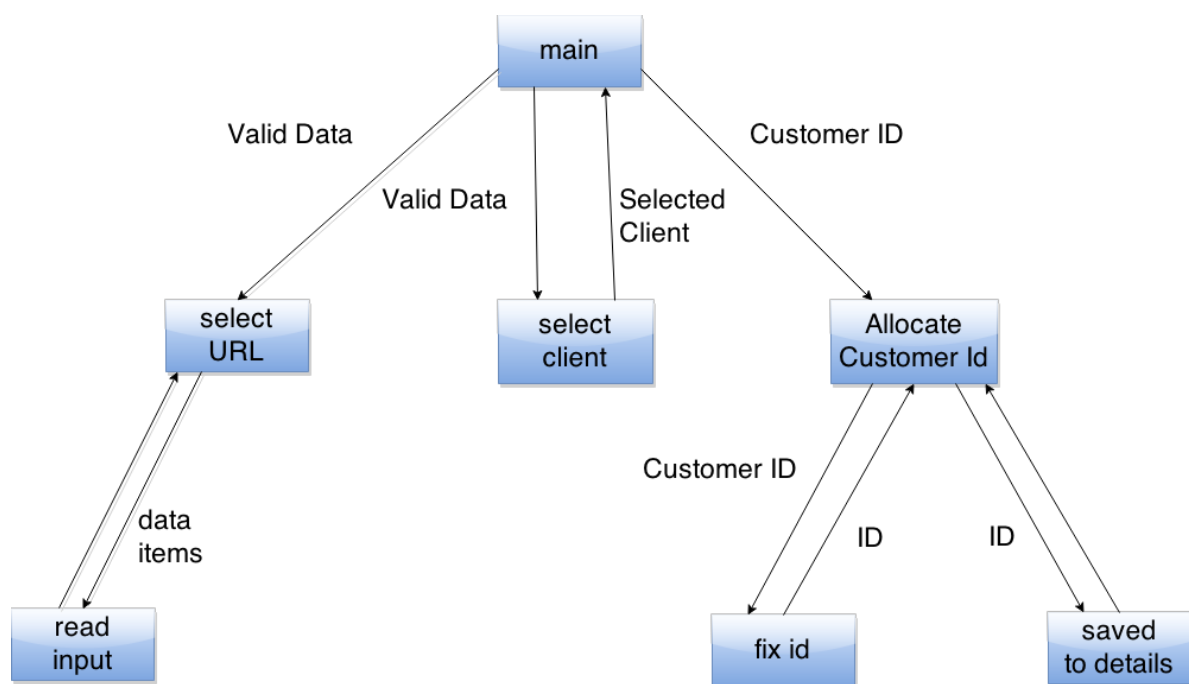


Fig5.3.3 Scanning report of Clients URL

5.3.4 Customer Id Allocation

Fig 5.3.4 Customer Id Allocation



5.4 INPUT/OUTPUT INTERFACE DESIGN

5.4.1 Home Page

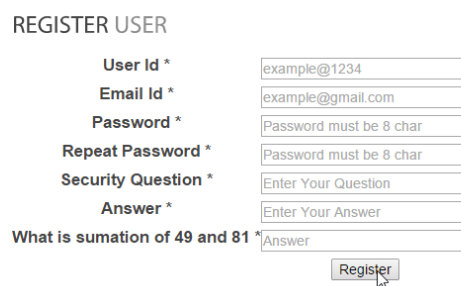
Fig 5.4.1 Home Page

5.4.2 Client Registration



The screenshot shows the priWAF Home Page. At the top center is a logo consisting of a red circle with a white 'C' and a black arrow pointing clockwise, with the text 'priWAF' in red and 'web application firewall' in small black text below it. Below the logo are two links: 'LOGIN' and 'REGISTER' in red. Under 'REGISTER' are two input fields: 'User Id *' with 'Username' as placeholder text, and 'Password *' with 'Password' as placeholder text. Below these fields is a red link 'Forgot Password' and a 'Login' button.

Fig 5.4.2 Client Registration



The screenshot shows the 'REGISTER USER' form. It contains the following fields and labels: 'User Id *' with placeholder 'example@1234', 'Email Id *' with placeholder 'example@gmail.com', 'Password *' with placeholder 'Password must be 8 char', 'Repeat Password *' with placeholder 'Password must be 8 char', 'Security Question *' with placeholder 'Enter Your Question', 'Answer *' with placeholder 'Enter Your Answer', and 'What is sumation of 49 and 81 *' with placeholder 'Answer'. A 'Register' button is at the bottom right.

5.4.3 Client Login



LOGIN REGISTER

User Id *

Password *

[Forgot Password](#)

Fig 5.4.3 Client login

5.4.4 Search Scanning Report

HOME LOGOUT

Welcome yash2554

Search By Customer ID Search By URL Search By Customer ID Search By URL

Search By Customer-ID:

Date:

Waiting for localhost...

Fig 5.4.4 Search Scanning Report

5.4.5 Scanning Report Download

HOME DASHBOARD LOGOUT

[Export to EXCEL](#)
[Export to CSV](#)
[Download](#)

INDUSFACE RECORDS

CustomerID	AlertID	FoundDate	Description	URL	Method	Para
3012	8482225	2015-04-13 00:00:33	<p>The Web application is vulnerable to cross-site scripting (XSS), which allows attackers to take advantage of Web server scripts to inject JavaScript or HTML code that is executed on the client-side browser. This vulnerability is often caused by server-side scripts	http://203.17.194.79/edelweiss/admin/js/tinymce/plugins/filemanager/dialog.php?type=0&lang=en_EN&popup=0&crossdomain=0&field_id=&relative_url=0&akey=key&fldr=&552a9668d8f2a	get	552a9668d8f2a

Fig 5.4.5 Scanning Report Download

5.4.6 WAF admin Login



LOGIN REGISTER

User Id *

Password *

[Forgot Password](#)

Fig 5.4.6 WAF admin Login

5.4.7 WAF admin dashboard



Fig 5.4.7 WAF admin dashboard

5.4.8 Customer Id Allocation

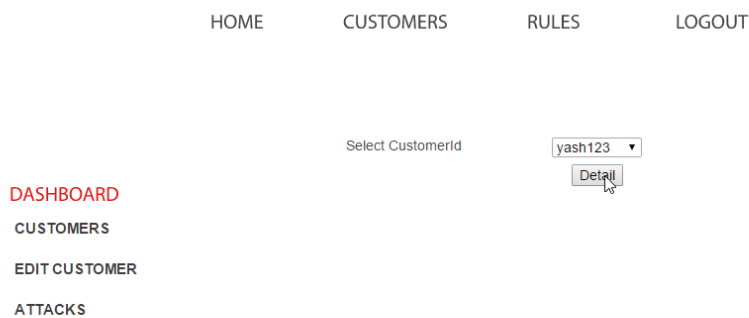


Fig 5.4.8 Customer Id Allocation

5.4.9 Edit Customer Details

HOME

CUSTOMERS

RULES

LOGOUT

DASHBOARD

CUSTOMERS

EDIT CUSTOMER

ATTACKS

EDIT DETAILS

UserID

yash123

CustomerID

2

Name

Yash Patel

Address

Ahm

Mobileno

9879316527

DATE(Service)

03-Apr-2015

IP

192.168.1.190

URL

webscantest.com

Submit

Reset

Fig 5.4.9 Edit Customer Details

5.4.10 Push Signature Rules of a specific Client

HOME

CUSTOMERS

RULES

LOGOUT

RULES DASHBOARD

SERVER RELOAD

CUSTOMERS

ATTACKS

Customer ID*:2

URL*:webscantest.com

IP-BlackList*:192.168.1.194

Rule Configuration*:<VirtualHost *:8800>
<!fModule mod_security2.c>

SecRuleEngine On
SecResponseBodyAccess On
SecRequestBodyAccess On
SecResponseBodyMimeType
text/plain text/html text/xml

Submit

Fig 5.4.10 Push Signature Rules of a specific Client

5.4.11 Server Apache Reload From Admin Panel



Fig 5.4.11 Server Apache Reload From Admin Panel

5.4.12 Client Attack Details Analysis/Monitoring

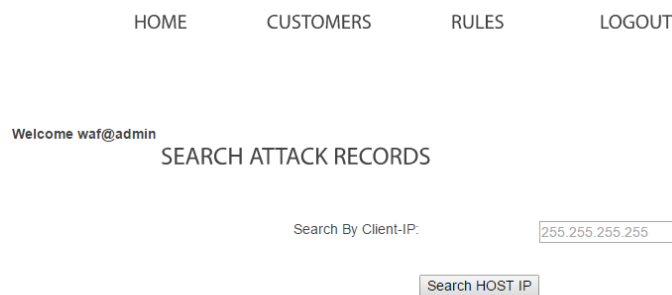


Fig 5.4.12 Client Attack Details Analysis/Monitoring

5.4.13 Attack Reports/Logs

[illegible]

Fig 5.4.13 Attack Reports/Logs

5.4.14 Customer Scanning Reports

HOME

CUSTOMERS

RULES

LOGOUT

Welcome waf@admin

Search By Customer ID

Search By URL

Search By Customer-ID:

Date:

007

1993-12-12

Search CustomerID

Fig 5.4.14 Customer Scanning Reports

Signature Development for Sec Rule (Core Rule set | Custom Rule Set)

Rule configuration (mod security)

[illegible]

Fig 5.4.15 Rule Configuration 1

```
##SQL INJECTION 3
SecRule ARGS:uid|ARGS:passwd|ARGS:textMail "(?!(?!([\\w,\\.\\'\\\\s]+char{?=\\(\\d\\)}))" "phase:2,t:lowercase,log,id:5,ddeny,msg:'NOT ALLOWED'"

##SQL INJECTION 3
SecRule REQUEST_URI|ARGS:uid|ARGS:passwd|ARGS:textMail "(?!([A-Z\\d]{6,})(?=\\d))" "deny,phase:2,id:6,log,msg:'SQL INJECTION'"
SecRule REQUEST_URI|ARGS:uid|ARGS:passwd|ARGS:textMail "(?!([\\'\\\\d][\\w\\-\\%]+)" "deny,phase:2,id:7,log,msg:'SQL INJECTION'"

##SQL INJECTION 4
SecRule ARGS:uid|ARGS:passwd|ARGS:textMail "(?!([\\'\\\\d]+(?=\\d)))" "deny,phase:2,t:lowercase,id:8,log,msg:'SQL INJECTION not Allowed'"

##SQL INJECTION 5
SecRule ARGS:uid|ARGS:passwd|ARGS:textMail "(?!([\\d](?=.*[a-z](2))))" "deny,phase:2,t:lowercase,id:9,log,msg:'SQL INJECTION not Allowed'"
# SecRule REQUEST_URI|ARGS|ARGS_NAMES "(!\\\"=%<.>/&:})"

##SQL INJECTION 6
SecRule ARGS:uid|ARGS:passwd|ARGS:textMail|ARGS_NAMES:btnSubmit "(?!([A-z\\'\\\\s]+(?(chr{([0-9]+\\)}) (?!\\)))" "deny,phase:2,t:lowercase,id:13,log,msg:'SQL INJECTION not Allowed'"

##SQL INJECTION 7
SecRule ARGS:uid|ARGS:passwd "[A-z\\'\\\\,]+(?(CHAR(=[\\'\\\\d]))+)" "deny,phase:2,t:lowercase,id:14,log,msg:'SQL INJECTION not Allowed'"
SecRule ARGS:uid|ARGS:passwd "\\+CHAR(=[\\'\\\\d])+)" "deny,phase:2,t:lowercase,id:14,log,msg:'SQL INJECTION not Allowed'"

##Directory Traversal
SecRule REQUEST_URI "(?!([\\..\\.])(2)(?!([\\w\\'\\\\s]+)" "deny,phase:1,t:lowercase,t:compressWhitespace,id:10,log,msg:'DIRECTORY NOT FOUND'"

##FILE INCLUSION
SecRule REQUEST_URI|REQUEST_HEADERS|ARGS "(!([\\..\\./])(3,))" "deny,phase:2,t:lowercase,t:compressWhitespace,log,id:11,msg:'FILE INCLUSION ATTACKS'"

##BRUTE FORCE PROTECTION
SecAction phase:1,nolog,pass,initcol:ip=%(REMOTE_ADDR),initcol:user=%(REMOTE_ADDR),id:'00113'
SecRule user:bf_block "%gt 0" "deny,status:403,log,id:'00114',msg:'IP address blocked for 10 seconds.',append:<script type=text/javascript>alert('IP address blocked f
or 10 seconds\");</script>"
SecRule REQUEST_METHOD ==POSTS" "phase:5,chain,t:none,nolog,pass,setvar:ip.bf_counter+=1,deprecatevar:ip.bf_counter=1/10,id:'00115'"
```

Fig 5.4.16 Rule Configuration 2

```

#SecAction phase:1,nolog,pass,initool:ip=$(REMOTE_ADDR),initool:user=$(REMOTE_ADDR),id:'00113'
#SecRule user:bf_block "%gt 0" "pass,status:403,log,id:'00114',msg:'IP address blocked for 10 seconds.'"
#SecRule REQUEST_METHOD "%POST%" "phase:5,chain,t:none,nolog,pass,setvar:ip.bf_counter+=1,deprecatevar:ip.bf_counter=1/10,id:'00115'"
#SecRule ip:bf_counter "%gt 3" "t:none,setvar:user.bf_block=1,expirevar:user.bf_block=10,setvar:ip.bf_counter=0"

#SecRule RESPONSE_STATUS "^403" "phase:3,pass,id:3130,msg:'Redirected',log,redirect:'http://192.168.1.184:8880'"

####Directory Listing

#SecRule RESPONSE_BODY "(?:<(?:TITLE>Index of.*?<H|title>Index of.*?<h)1>Index of|>\\[To Parent Directory\\]\\<\\[Aa]><br>)" "phase:4,t:none,capture,ctl:auditLogParts==E,deny,msg:'Directory Listing',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',id:'970013',severity:'3'"

####COOKIE-SESSION

#SecRule REQUEST_HEADERS|REQUEST_COOKIES|ASP.NET_Sessionid "!^[0-9A-z]{24}$" "phase:1,log,id:12,block,msg:'INVALID Session Token'"
#SecRule RESPONSE_HEADERS|Cookie:/ "(?:i(?:?sessionid|(php)?sessid|(asp|j|serv|jw)?session[_-]?(?:id)?(of(id)(token)|sid)=[{^\\s+}\\s?)" "chain,phase:3,t:none,id:13,pass,nolog,capture,setvar:session.sessionid=${TX.6},setvar:session.csrf_token=${TX.1}"
# SecRule SESSION:SESSIONID "(.*)" "t:none,t:sha1,t:hexEncode,capture,setvar:session.csrf_token=${TX.1}"

#####

ProxyRequests off

ProxyPass / http://demo.testfire.net/
ProxyPassReverse / http://demo.testfire.net/

ServerSignature Off

</IfModule>
</VirtualHost>

#SecRule REQUEST_URI "(\\/pr\\/)$" "phase:1,id:'55',allow,t:urlDecode,log,msg:'Potential intrusion detected',append:'<script type=text/javascript>alert(\"Stop trying to hack our site\\n\");</script>'"

```

Fig 5.4.17 Rule Configuration 3