# Raswall Whitepaper

:

# INDEX

## Introduction

In this modern era where the whole world has become a giant and complex network providing end to end communication and services at any corner of the world.

The Network security has emerged as an alarming issue in all major aspects of businesses and other communication services due to rigorous increase in sophistication and abundance of security breaches. It is involved in various organizations, enterprises, and other types of institutions increasing its necessity to provide secure transmission of data. To prevent these security breaches we have come up with different ways to tackle these out of which the Firewall is the most popular and trusted one.

A Network Security Firewall or commonly known as Firewalls is a network security system which could be either hardware or software, presented at a network gateway server which monitors or controls the incoming and outgoing traffic of data of a private network. It creates a trusted wall of restrictions between the private network and the rest of the world. It checks each message coming or outgoing from the network and drop those messages which do not pass the access control tests.

## Objective

- Generally, most of the firewalls use proxy servers to check the validity of the application layer data. Doing so affects confidentiality of the data being transferred. Hence, our objective is to find the most efficient way to manage network access controls without interfering with the confidentiality of the data belonging to end users.

- To find the efficient way of identification of a node in a particular network upon which access controls will be implemented. Generally, most of the access control servers validate the identity of a node by checking their network layer addresses or IP addresses, and in these type of systems access controls can be easily bypassed. As DHCP server which provides a dynamic IP address and the server which check for access controls are not properly managed.

As a result, we will try to design a firewall for Raspberry pi by overcoming the above stated flaws and that will be first of its kind.

## Motivation

While studying the behavior of various firewalls we came across certain points as follows:

- In order to increase the security or to achieve better access controls over the transmitted data, firewalls have to compromise with the confidentiality of the end users. For e.g., a highly confidential https request must be intercepted and decrypted by proxy servers or firewall I order to validate and there on forwarded to actual server. Hence, transmitted data is no more secure or confidential.
- Most networks have different network infrastructure for DHCP server and access control servers ( firewalls ). So, IP addresses can be easily spoofed in these cases. Thus, the access control servers validating the node based on its IP address is no more useful.

## Implementation Details

- For comparison purposes we shall setup one conventional firewall using iptables and a proxy server(squid) on linux.
- We shall use iptables and layer7 filtering iptable module on linux as a separate firewall to calculate confidentiality and security.
- We shall setup a nodejs server on linux as captive portal web server.
- We shall setup a DHCP server and access control server on same linux machine such that DHCP server and access control server can be managed properly.

## Technologies Required:

1. Linux as a server operating system or routing platform.
2. IPtables as a packet filtering firewall.
3. Nodejs as a web server that will provide front end configurator for firewall, capative portal and DHCP server etc.
4. Raspberry pi for networking and server hardware.
5. Python for network programming.

## Implementation

1. **Using iptables (net filter under linux kernel) we can filter packets based on their destination addresses.**

*NOTE: Here privacy of the user data has higher preference over the security of user as well as the organization.*

<u>Packet Filter:</u>

By packet filtering we can analyze the network traffic using the IP and port addresses. Each IP network packet is examined to see if it matches one of a set of rules defining what data flows are allowed to pass through the firewall. The rules determine whether communication is allowed based upon the information contained within the Internet and transport layer headers and the direction that the packet is headed. These Packet filters enable the administrator to permit or prohibit the transfer of data based on the following controls: the physical network interface that the packet arrives on; the source IP address the data is coming from; the destination IP address the data is going to; the type of transport layer; the transport layer source port, and the transport layer destination port.

<u>Netfilter:</u>

It is a framework provided by the Linux Kernel that allows various networking-related operations to be implemented in the form of customized handlers. It offers various functions and operations for packet filtering, network address translation, and port translation, which provide the functionality required for directing packets through a network, as well as for providing ability to prohibit packets form reaching sensitive locations within a computer network.

**<u>IPtable script:</u>** (See Appendix)

**<u>Main.py</u>** (See Appendix)

**<u>Login Manager.py</u>** (See Appendix)

**Observation**:

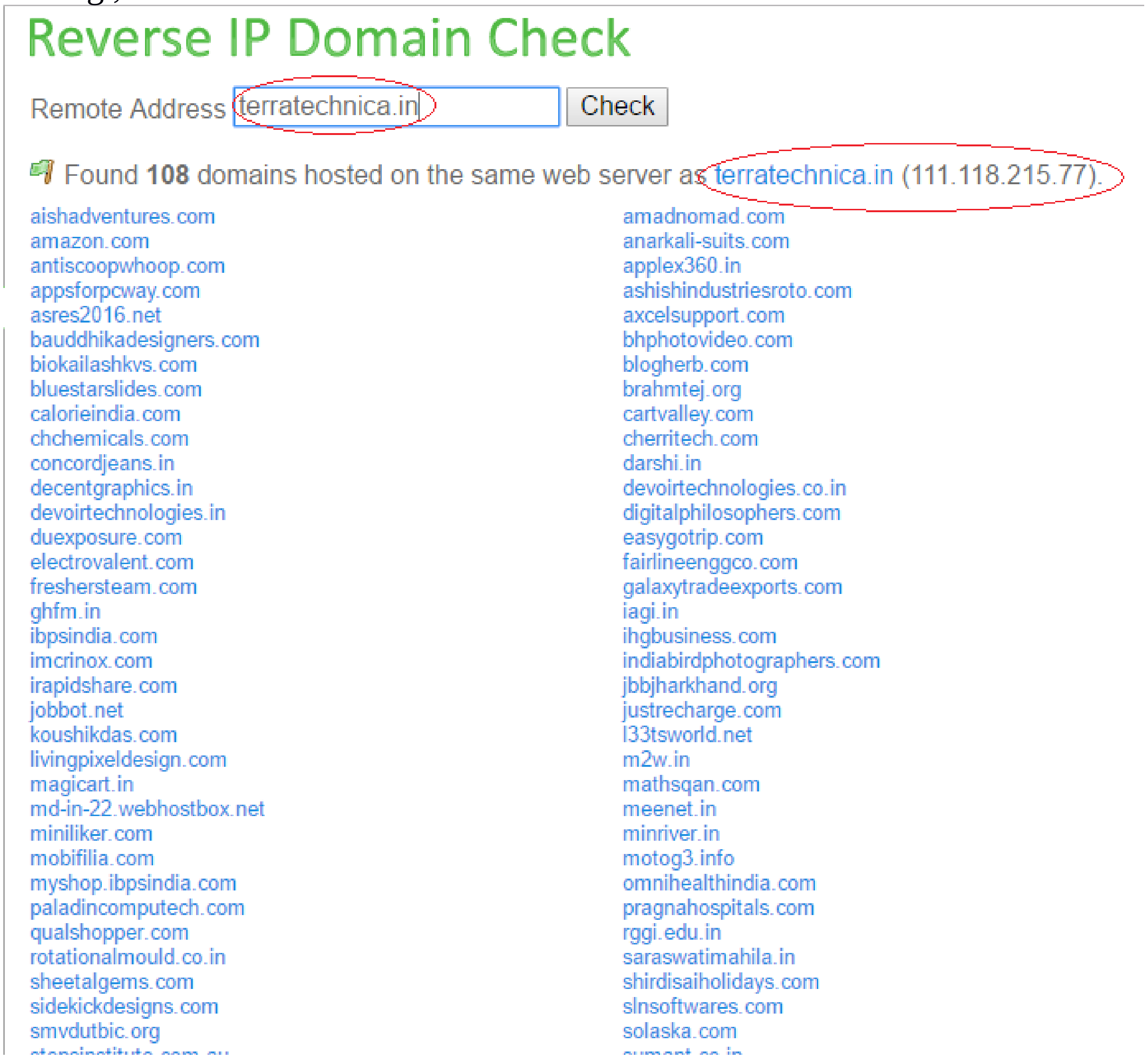
While performing the packet filtering using the netfilter on the incoming or outcoming packets between the user-end and the internet, we successfully examined the network and transport layer headers of these packets. The information stored in these headers such as the destination IP address and port address were checked against the rules for allowing them to pass through the firewall and those which were failed to do so were dropped successfully.

**Problems Encountered:**

Although we achieved full confidentiality of data and still managed to check the destination IP and port addresses but under some circumstances such packet filtering is not effective due to following reasons:

- As there could be many domains hosting at same time on a single IP address at the web server. Hence dropping the request for a particular domain could lead to the dropping of the all other domains curreny hosting on the same IP address( as we are checking the requested IP addresses to the allowed IP addresses in our iptables).

For e.g.,



Reverse IP Domain Check

Remote Address terratechnica.in Check

Found **108** domains hosted on the same web server as terratechnica.in (111.118.215.77).

| | |
|---|---|
| aishadventures.com | amadnomad.com |
| amazon.com | anarkali-suits.com |
| antiscoopwhoop.com | applex360.in |
| appsforpcway.com | ashishindustriesroto.com |
| asres2016.net | axcelsupport.com |
| bauddhikadesigners.com | bhphotovideo.com |
| biokailashkvs.com | blogherb.com |
| bluestarslides.com | brahmtej.org |
| calorieindia.com | cartvalley.com |
| chchemicals.com | cherritech.com |
| concordjeans.in | darshi.in |
| decentgraphics.in | devoirtechnologies.co.in |
| devoirtechnologies.in | digitalphilosophers.com |
| duexposure.com | easygotrip.com |
| electrovalent.com | fairlineenggco.com |
| freshersteam.com | galaxytradeexports.com |
| ghfm.in | iagi.in |
| ibpsindia.com | ihgbusiness.com |
| imcrinox.com | indiabirdphotographers.com |
| irapidshare.com | jbbjharkhand.org |
| jobbot.net | justrecharge.com |
| koushikdas.com | l33tsworld.net |
| livingpixeldesign.com | m2w.in |
| magicart.in | mathsqan.com |
| md-in-22.webhostbox.net | meenet.in |
| miniliker.com | minriver.in |
| mobifilia.com | motog3.info |
| myshop.ibpsindia.com | omnihealthindia.com |
| paladincomputech.com | pragnahospitals.com |
| qualshopper.com | rggi.edu.in |
| rotationalmould.co.in | saraswatimahila.in |
| sheetalgems.com | shirdisaiholidays.com |
| sidekickdesigns.com | slnsoftwares.com |
| smvdutbic.org | solaska.com |

- Some domains are configured for DDNS(dynamic DNS) i.e., there IP address is not static and changes often. Since, we are checking the IP address of requested headers against the IP addresses stored in iptables having permission to pass through the firewall. Hence, if the IP address of a domain changes then there will be no rule for the newly assigned IP address for that domain and thus we could not effectively block such IP addresses.

2. **Using transparent proxy(SQUID) and iptables.**

*To force network 0policies we must intercept all the traffic. Most of the traffic is usually http and ftp related. So, we must intercept all http traffic and should be able parse every request. For this purpose, we use squid proxy server. But we have to configure all our clients to use our proxy server. Turnaround for this is to use transparent proxy mode.*

## **Mongo DB :**

At the core of most large-scale web applications and services is a high-performance data storage solution. The backend data store is responsible for storing everything from user account information to shopping cart items to blog and comment data. Good web applications need to be able to store and retrieve data with accuracy, speed, and reliability. Therefore, the data storage mechanism we have chosen must be able to perform at a level to satisfy our server demand. Several different data storage solutions are available to store and retrieve data needed by web applications. The three most common are direct file system storage in files, relational databases, and NoSQL databases. We have focussed on the third type: a NoSQL database called MongoDB.

**NoSQL:** NoSQL (which is short for Not Only SQL) consists of technologies that provide storage and retrieval without the tightly constrained models of traditional SQL relational databases. The main motivations behind NoSQL are simplified designs, horizontal scaling, and finer control of the availability of data. The idea of NoSQL is to break away from the traditional structure of relational databases and allow developers to implement models in ways that more closely fit the data flow needs of their system. NoSQL databases can be implemented in ways that traditional relational databases could never be structured.

## NodeJs :

Node.js is a website/application framework designed with high scalability in mind. It takes advantage of the existing JavaScript technology in the browser and flows those same concepts all the way down through the webserver into the backend services. Node.js is a great technology that is easy to implement and yet extremely scalable. Node.js is a very modular platform, which means much of the functionality that we will use is provided by external modules rather than being inherently built into the platform. The Node.js culture is very active in creating and publishing modules for almost every imaginable need. Therefore, we have focussed on understanding and using the Node.js tools to build, publish, and use our own Node.js modules in this applications.

## ExpressJs :

Express provides a lightweight module that wraps the functionality of the Node.js http module in a simple-to-use interface. Express also extends the functionality of the http module by making it easy for you to handle server routes, responses, cookies, and HTTP request statuses.

**SQUID:** It is a caching proxy for the Web supporting HTTP, HTTPS, FTP, and more. It reduces bandwidth and improves response times by caching and reusing frequently-requested web pages. Squid has extensive access controls and makes a great server accelerator. Squid's access control scheme is relatively comprehensive and difficult for some people to understand. There are two different components: ACL elements, and access lists. An access list consists of an allow or deny action followed by a number of ACL elements. When loading the configuration file Squid processes all the ACL lines (directives) into memory as tests which can be performed against any request transaction. By themselves these tests do nothing. For example; the word "Sunday" matches a day of the week, but does not indicate which day of the week you are reading this. To process a transaction another type of line is used. As each processing action needs to take place a check in run to test what action or limitations are to occur for the transaction.

**Transparent proxy:**
A transparent proxy (also called inline proxy, intercepting proxy, or forced proxy) is a server that sits between your computer and the Internet and redirects your requests and responses without modifying them. A proxy server that does modify your requests and responses is defined as a non-transparent proxy. A transparent proxy can be used for various reasons, and as it does not need any configuration on the client side, it can be an easy-to-maintain alternative to other proxy types.

**NOTE :-  To intercept HTTPS traffic we must install a SSL certificate to all clients device.  We are generating a self-signed certificate for our setup .**

Interception caching is generally implemented by configuring a network device (router or switch) on our network perimeter to divert client requests to our Squid server. Other components that need to be configured include packet filtering software on the operating system running Squid, and finally Squid itself. First of all, let's see how the interception of requests occurs:

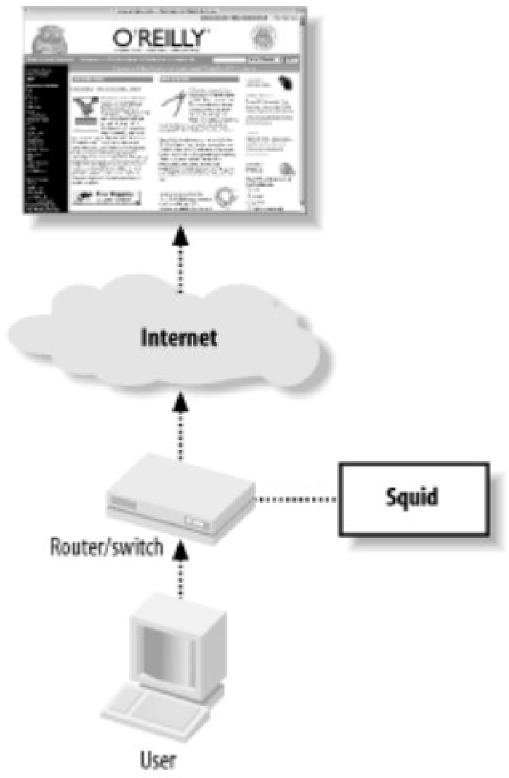
1. A client requests a web page http://www.example.com/index.html .
2. First of all, the client needs to resolve the domain name to determine the IP address, so that it can connect to the remote server. Next, the client contacts the DNS server and resolves the domain name www.example.com to 192.0.2.10 .
3. Now, the client initiates a TCP connection to 192.0.2.10 on port 80.
4. The connection request in the previous step is intercepted by the router/switch and is directed to the Squid proxy server instead of sending it directly to a remote server.
5. On the Squid proxy server, the packet is received by the packet filtering tool, which is configured to redirect all packets on port 80 to a port where Squid is listening.
6. Finally the packet reaches Squid, which then pretends its the remote server and establishes the TCP connection with the client.
7. At this point, the client is under the impression that it has established a connection

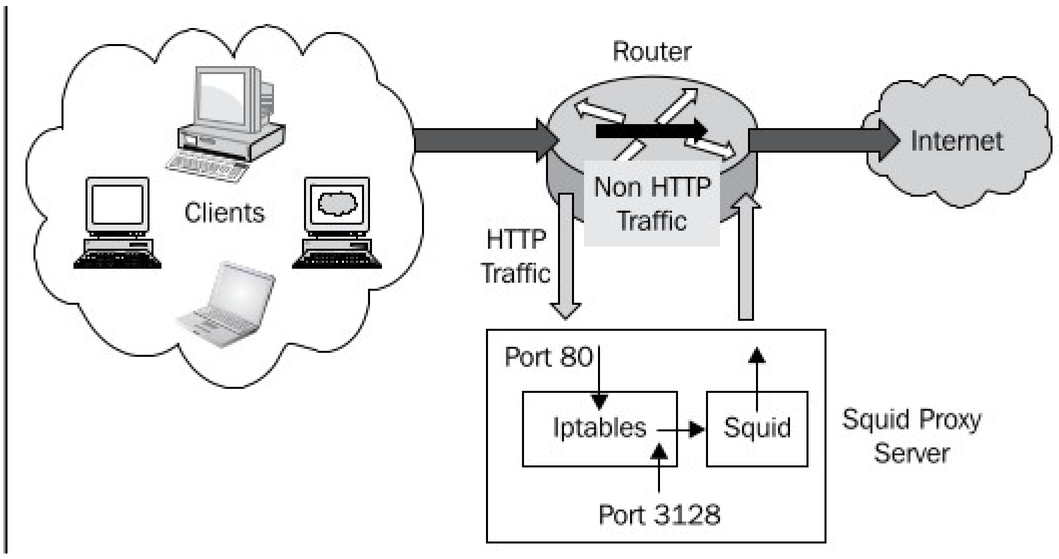
with the remote server when it's actually connected to the Squid server.

8. Once the connection is established, the client sends a HTTP request to the remote server asking for a specific URL ( /index.html in this case).
9. When Squid receives the request, it then pretends to be a client and establishes a connection to the remote server, if the client request can't be satisfied from the cache, and then fetches the content the client has requested.

**So, the idea is to redirect all our HTTP traffic to the Squid server using router/switch and host-based IP packet filtering tools.**

**Figure 9-1. How HTTP interception works**





## Core Components of Squid :-

### <u>Access Control List (ACL) :</u>

Access controls are the most important part of our Squid configuration file. We have used them to grant access to our authorized users and to keep out the bad guys. We have used them to restrict, or prevent access to, certain material; to control request rewriting; to route requests through a hierarchy; and to support different qualities of service.

Access controls are built from two different components. First, we have defined a number of access control list (ACL) elements. These elements refer to specific aspects of client requests, such as IP addresses, URL hostnames, request methods, and origin server port numbers. After defining the necessary elements, you combine them into a number of access list rules. These rules apply to particular services or operations within Squid. For example, the http_access rules are applied to incoming HTTP requests. There are two different components: *ACL elements*, and *access lists*. An access list consists of an *allow* or *deny* action followed by a number of ACL elements. When loading the configuration file Squid processes all the acl lines (directives) into memory as tests which can be performed against any request transaction. Types of tests are outlined in the next section *ACL Elements*. By themselves these tests do nothing. For example; the word "Sunday" matches a day of the week, but does not indicate which day of the week you are reading this.

To process a transaction another type of line is used. As each processing action needs to take place a check in run to test what action or limitations are to occur for the transaction.

## ACL Elements:

- **dst**: destination (server) IP addresses
- **myip**: the local IP address of a client's connection
- **arp**: Ethernet (MAC) address matching
- **srcdomain**: source (client) domain name
- **dstdomain**: destination (server) domain name
- **time**: time of day, and day of week
- **url_regex**: URL regular expression pattern matching
- **port**: destination (server) port number
- **proto**: transfer protocol (http, ftp, etc)
- **method**: HTTP request method (get, post, etc)
- **http_status**: HTTP response status (200 302 404 etc.)
- **browser**: regular expression pattern matching on the request user-agent header

## Access Control Lists :

There are a number of different access lists:

- ***http_access***: Allows HTTP clients (browsers) to access the HTTP port. This is the primary access control list.
- ***http_reply_access***: Allows HTTP clients (browsers) to receive the reply to their request. This further restricts permissions given by http_access, and is primarily intended to be used together with rep_mime_type acl for blocking different content types.
- ***cache***: Defines responses that should not be cached.
- ***url_rewrite_access***: Controls which requests are sent through the redirector pool.

- ***ident_lookup_access***: Controls which requests need an Ident lookup.
- ***always_direct***: Controls which requests should always be forwarded directly to origin servers.
- ***never_direct***: Controls which requests should never be forwarded directly to origin servers.
- ***snmp_access***: Controls SNMP client access to the cache.
- ***cache_peer_access***: Controls which requests can be forwarded to a given neighbor (cache_peer).
- ***icap_access***: (replaced by adaptation_access in Squid-3.1) What requests may be sent to a particular ICAP server.
- ***adaptation_access***: What requests may be sent to a particular ICAP or eCAP filter service.
- ***log_access***: Controls which requests are logged. This is global and overrides specific file access lists appended to access_log directives.

**<u>Sample  ACL Rules :</u>**

- *acl localnet src 10.20.48.0/22 127.0.0.1*
- *acl SSL_ports port 443*
- *acl Safe_ports port 80        # http*
- *acl Safe_ports port 21        # ftp*
- *acl Safe_ports port 443       # https*
- *acl Safe_ports port 70        # gopher*
- *acl Safe_ports port 210       # wais*
- *acl Safe_ports port 1025-65535    # unregistered ports*
- *acl Safe_ports port 280       # http-mgmt*
- *acl Safe_ports port 488       # gss-http*
- *acl Safe_ports port 591       # filemaker*

- *acl Safe_ports port 777       # multiling http*
- *acl CONNECT method CONNECT*
- *http_access deny !Safe_ports*
- *http_access deny CONNECT !SSL_ports*

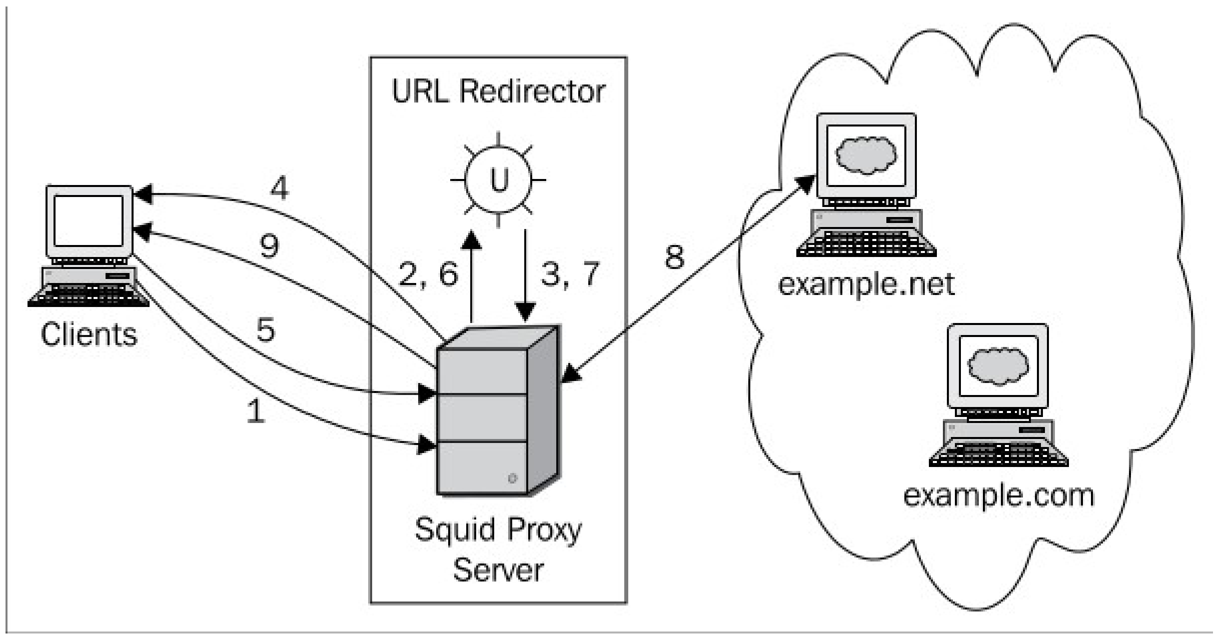
***Note:-*** *We will be using nodejs scripting language to control redirection of clients to implement access control policies .*

## <u>URL redirectors and rewriters</u>

*URL redirectors are external helper processes that can redirect the HTTP clients to alternate URLs using HTTP redirect messages. Similarly, URL rewriters are also external helper processes that can rewrite the URLs requested by the client with another URL. When a URL is rewritten by a helper process, Squid fetches the rewritten URL transparently and sends the response to the end client as if it was the one originally requested by the client.*

The URL redirectors can be used to send HTTP redirect messages like 301, 302, 303, 307, or 3xx, along with an alternate URL to the HTTP clients. When a HTTP client receives a redirect message, the client will request the new URL. So, the major difference between URL redirectors and URL rewriters is that the client is aware of a URL redirect, while rewritten URLs are fetched transparently by Squid, and the client remains unaware of a rewritten URL .

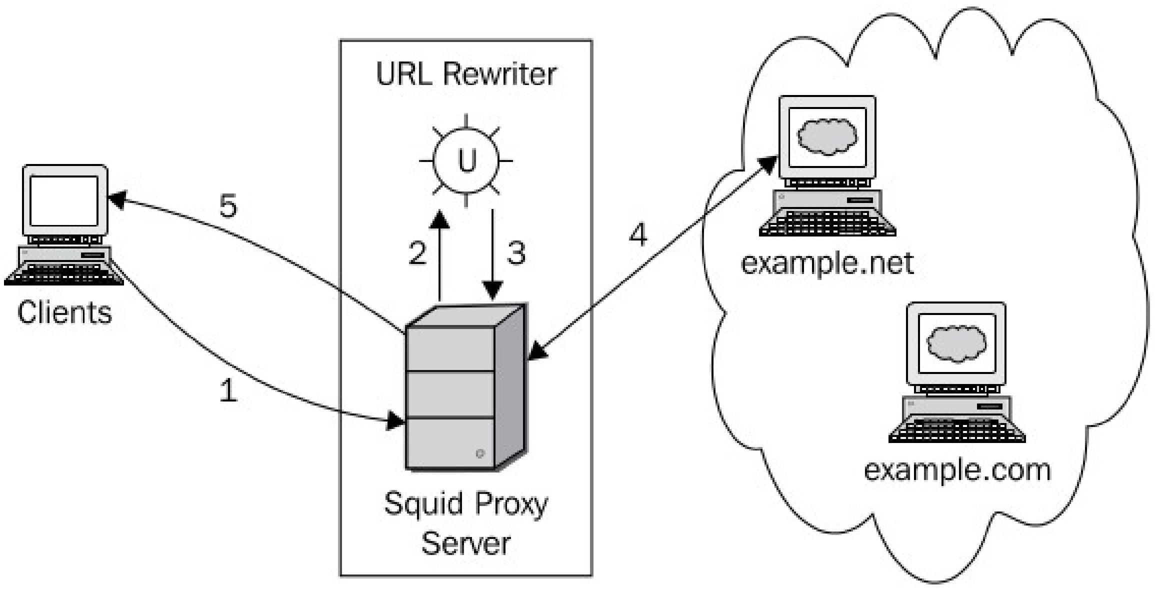
## Understanding URL redirectors



1. The Client requests the webpage http://example.com/index.html .
2. The Squid Proxy Server receives the requests and forwards the essential details related to the request to the URL redirector helper program.
3. The URL redirector helper program processes the details and issues a 303 HTTP redirect with an alternate URL http://example.net/index.html . In other words, the URL redirector program suggests to Squid that the client should be redirected to a different URL.
4. Squid, as suggested by the URL redirector helper, sends the redirect message to the client with the alternate URL.
5. The client, on receiving the redirect message, initiates another request for the new URL http://example.net/index.html .
6. When Squid receives the new request, it is again sent to the URL redirector helper program.
7. The URL redirector program processes the request and suggests to Squid that this URL can be fetched and we don't need to redirect the client to an alternate URL.
8. Squid fetches the URL http://example.net/index.html .

9. The response received by Squid from the origin server at example.net is delivered to the client.

## Understanding URL rewriters



1. The client requests a URL http://example.com/index.html using our proxy server.
2. Squid receives the request and forwards the essential details about the request to the URL rewriter helper program.
3. The URL rewriter helper program processes the details received from Squid and suggests to Squid that it should fetch http://example.net/index.html instead of http://example.com/index.html . In other words, the rewriter program has rewritten the URL with a new URL.
4. Squid receives the rewritten URL ( http://example.net/index.html ) from

the rewriter program and contacts the origin server at example.net instead of contacting example.com .

5. Squid delivers the response returned by the origin server at example.net to the client.

## Sample acl for redirector and rewriter: -

```
redirect_program /usr/local/bin/node /var/spool/squid/sample.js

redirect_children 30

redirect_rewrites_host_header off

redirector_access allow all
```

## Why we used url Rewriter and Redirector: -

- Network access controls are usually based on the IP address allotted the clients
- So, traffic policy for clients should be based upon their IP addresses
- Redirector are used to control access for all http/https/ftp requests
- IP address of every request is being checked by the redirectors and then action is taken accordingly.

## Internet Content Adaptation Protocol :-

ICAP, the Internet Content Adaption Protocol, is a protocol aimed at providing simple object-based content vectoring for HTTP services. ICAP is, in essence, a lightweight protocol for executing a "remote procedure call" on HTTP messages. It allows ICAP clients to pass HTTP messages to ICAP servers for some sort of transformation or other processing ("adaptation"). The server executes its transformation service on messages and sends back responses to the client, usually with modified messages. The adapted messages may be either HTTP requests or HTTP responses.

## Response Modification in ICAP :-

In the "response modification" (respmod) mode, an ICAP client sends an HTTP response to an ICAP server. (The response sent by the ICAP client typically has been generated by an origin server.) The ICAP server may then:

1) Send back a modified version of the response.

2) Return an error.

The response modification method is intended for post-processing performed on an HTTP response before it is delivered to a client. Examples include formatting HTML for display on special devices, human language translation, **virus checking**, and so forth.

```
Typical data flow:

   origin-server
       | /|\
       |  |
    3  |  |  2
       |  |
     \|/ |           4
   ICAP-client    -------------->   ICAP-resource
   (surrogate)    <--------------   on ICAP-server
       | /|\           5
       |  |
    6  |  |  1
       |  |
     \|/ |
     client
```

1. A client makes a request to a ICAP-capable surrogate (ICAP client) for an object on an origin server.

2. The surrogate sends the request to the origin server.

3. The origin server responds to request.

4. The ICAP-capable surrogate sends the origin server's reply to the ICAP server.

5. The ICAP server executes the ICAP resource's service on the origin server's reply and sends the possibly modified reply back to the ICAP client.

6. The surrogate sends the reply, possibly modified from the original origin server's reply, to the client.

## C-ICAP Server :-

c-icap is an implementation of an ICAP server. It can be used with HTTP proxies that support the ICAP protocol to implement content adaptation and filtering services.

## SquidClamav :-

### Clamav ICAP service and redirector for Squid

SquidClamav is an antivirus for Squid proxy based on the Awards winnings ClamAv anti-virus toolkit. Using it will help us securing our home or enterprise network web traffic. SquidClamav is the most efficient Squid Redirector and ICAP service antivirus tool for HTTP traffic available for free, it is written in C and can handle thousand of connections. The way to add more securing on your network for free is here.

SquidClamav is built for speed and security in mind, it is first used and tested to secure a network with 2,500 and more users. It is also known to working fast with 15000+ users.

### How it works

With SquidClamav You have full control of what kind of HTTP stream must be scanned by Clamav antivirus, this control operate at 3 different levels:

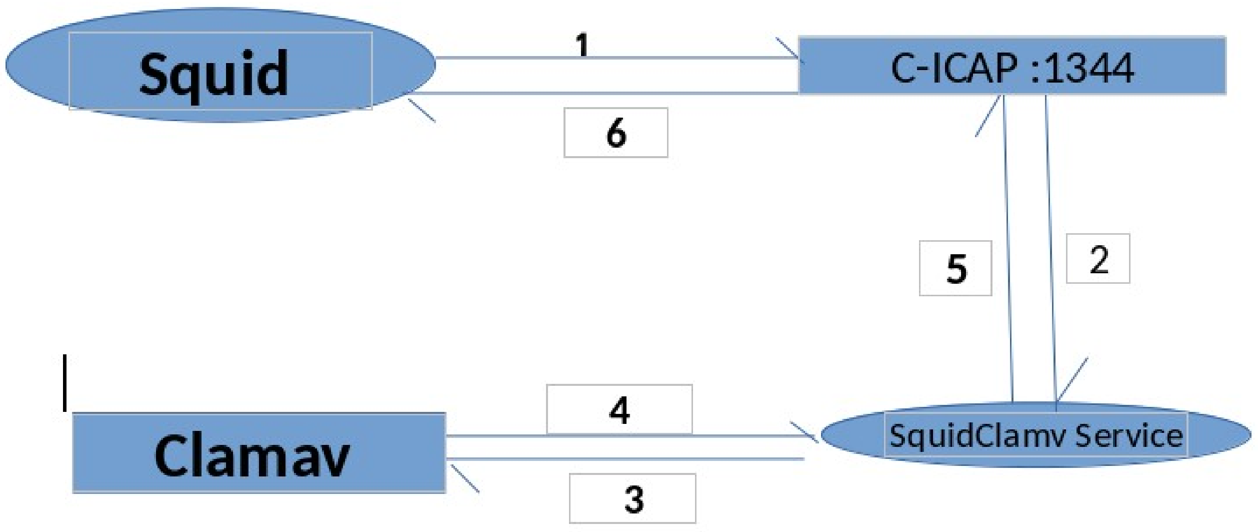
> At URL level, you can disable virus scanning for a set of web site, filename extension or anything that can be matched in an URL.
> At client side by disabling virus scan and other redirector call to a set of username, source Ip addresses or computer DNS name.
> At HTTP header level, where you can disable virus scanning following the content type or file size.

## Clamav :-

Clam AntiVirus is an open source (GPL) anti-virus toolkit for UNIX, designed especially for e-mail scanning on mail gateways. It provides a number of utilities including a flexible and scalable multi-threaded daemon, a command line scanner and advanced tool for automatic database updates. The core of the package is an anti-virus engine

available in a form of shared library.

## How C-ICAP + SquidClamav + Clamav + Squid works together :-



1. Squid sends a ICAP respmod request for squidclamav service to c-icap server running on localhost at port 1344 with the payload i.e. response from origin server .
2. C-icap server recieves the respmod request and forwards the request to squidclamav module/service ( which is running as library ) .
3. Squidclamav module now writes payload present in respmod request ( i.e. response from origin server ) to clamav socket for scanning of payload .
4. Clamav antivirus which is running as daemon will be listening on its socket and will reply whether the payload is malicious or not after scanning to the process asked for scanning .
5. Squidclamav will collect reply from clamav socket and now tell c-icap server what to do on when payload is malicious or not .

1. If reply from clamav says that payload was malicious then squidclamav will tell c-icap server to send a redirect HTTP response to squid in response of ICAP respmod request .
2. If reply from clamav says payload is not malicous then squidclamav will tell c-icap server to send an OK response to squid in response to ICAP respmod request .

Now Squid sends response to client depending on ICAP respmod response . If response says OK then squid will forward the original response from origin server to client after caching .

If response was a redirect HTTP response then squid will send that redirect HTTP response to the client . Now Client will be redirected to the malware warning page hosted on the our gateway .

## ISC DHCP Server :-

All IP devices need addresses, and ISC DHCP is the classic way to provide them. ISC DHCP is open source software that implements the Dynamic Host Configuration Protocol for connection to an IP network .

## Hostapd :-

**hostapd** is a user space daemon for wireless access point and authentication servers.

## So, What happens when a user connects to the network: -

1. Client will be given network configuration by the dhcp server i.e., gateway, ip address, nameserver addresses etc. The DHCP server will be hosted on raspberry pi wireless interface.
2. As gateway for the client is our firewall's ethernet interface ( 10.20.48.1 ) . So, all of client's traffic will be passing through our firewall network card.
3. Now all traffic but DNS traffic will be forwarded to the internet through another interface. Rest Traffic from client will be forwarded to the proxy server i.e. squid in our case, to implement access controls on client's requests. We are using Netfilter's implementation i.e. Iptables in our case to shape the data flow.

```
# Redirecting packets going to port 80 and 443 to our machine ports 3128 and 3129 respectively
iptables -t nat -I PREROUTING -p tcp --dport 80 -i vboxnet0 -j DNAT --to-destination 10.20.48.1:3128
iptables -t nat -I PREROUTING -p tcp --dport 443 -i vboxnet0 -j DNAT --to-destination 10.20.48.1:3129

# Masquerading client's dns requests
iptables -t nat -I POSTROUTING -p udp --dport 53 -o wlp6s0 -j MASQUERADE

# opening ports 3128 , 3129 for squid proxy server and port 10000 for our login manager.
iptables -t filter -I INPUT -p tcp --dport 10000 -j ACCEPT
iptables -t filter -I INPUT -p tcp --dport 3128 -j ACCEPT
iptables -t filter -I INPUT -p tcp --dport 3129 -j ACCEPT
```

4. All HTTP and HTTPS traffic is redirected to squid. Now squid check the client's request against access control lists. Sample Acls are defined earlier (in ACL section).
5. If client's request is OK according then the request is passed to the url redirector / rewriter. Sample Acl for redirector / rewriter is defined earlier.

6. Now the redirector process nodejs script in our case checks if the client's IP address is authenticated i.e. if client's IP address is present in authorised client's database.
7. Now client's IP address is checked if previously client has authenticated or not .

```
# authSessions client's collection
username: String,
magic:String,
ip: String,
category: String,
maxTime: Number,
req_url: String,
loginTime:Date
```

8. If Client ip is not authorised then a redirect message of redirection towards login manager is sent to client in reply to original request.

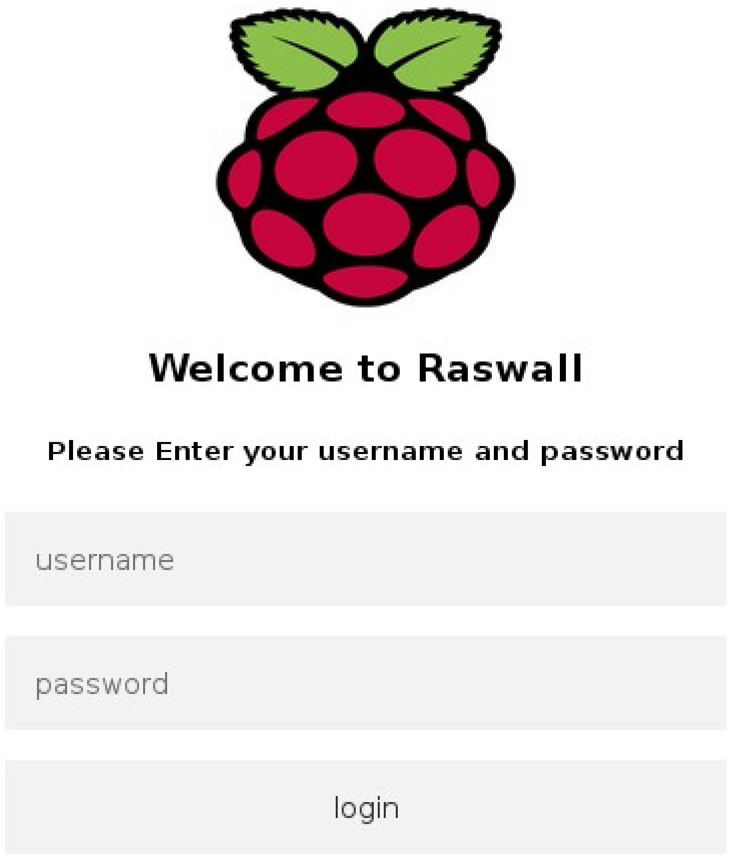
```
Client Request
http://detectportal.firefox.com/success.txt 10.20.48.2/10.20.48.2 - GET myip=-
myport=3128

Squid Response for redirection with code 302
302:http://10.20.48.1:10000/login?598f13866e04be29579332fc
```

9. Now Client's browser will request for login page to our login manager server and in response login form is sent to client.

10. Client now fills the login form and submits that form to login server.

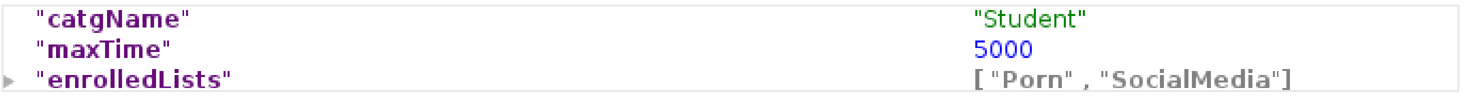
- If login credentials are OK then client's IP address and other details will be added to the authSessions collection.
- If Credentials are not OK then again client will be provided with a login page by login manager server.

11. Steps No. 8, 9 and 10 will be repeated until client's ip address is not added to the authSessions collection.

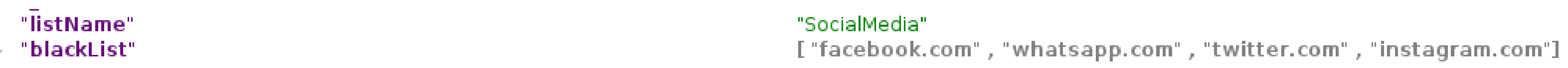


12. If client IP is authorised i.e. present in authSessions collection, then client's request is checked against data access policies like Social Media, Adult content etc .

```
// schema for category information
catgName:{type:String,unique:true},
maxTime:Number,
enrolledLists:[String]          // array of name of blacklists

// schema contains listname and blacklist of websites that it is intended to block
listName:{type:String,unique:true},
blackList:[String]
```



```
"catgName"          "Student"
"maxTime"           5000
"enrolledLists"     [ "Porn" , "SocialMedia"]
```



```
"listName"          "SocialMedia"
"blackList"         [ "facebook.com" , "whatsapp.com" , "twitter.com" , "instagram.com"]
```

Sample Documents of Data Policy in database

13. If Client's request is found OK then it is checked if some response is cached in the squid's cache unit.
14. If response was present in cache then it will be served from this cache.

Sample Squid Cache Configuration

```
# Uncomment and adjust the following to add a disk cache directory.

Cache_dir ufs /var/spool/squid 100 16 256


# Leave coredumps in the first cache dir

coredump_dir /var/spool/squid



# Add any of your own refresh_pattern entries above these.


refresh_pattern ^ftp:144020%10080

refresh_pattern ^gopher:14400%1440

refresh_pattern -i (/cgi-bin/|\?) 00%0

refresh_pattern .020%4320

cache_effective_user squid

cache_effective_group squid
```

15. If not present then request to the original origin server will be sent by squid.
16. Now response came from origin server is checked against icap access control list.
17. And if icap access control list allow response then it is sent to squidclamav an open source antivirus to check if malware exists or not through icap protocol.

```
icap_enable on

icap_send_client_ip on

icap_preview_enable on

icap_preview_size 1024
```

```
icap_service service_resp respmod_precache bypass=0
icap://127.0.0.1:1344/squidclamav


adaptation_access service_resp allow all
```

18. Response sent to c-icap server ( open source internet content adaptation server ) for accessing respmod service i.e. squidclamav will be then forwarded to clamd daemon ( open source antivirus ) .
19. If malware is found in the Response to the client request then client will be redirected to some other page showing info about malware found in the response and original response will be deleted.
20. If response is OK then response from origin server will be forwarded to the squid proxy and squid may cache that response and then pass that response to the client.

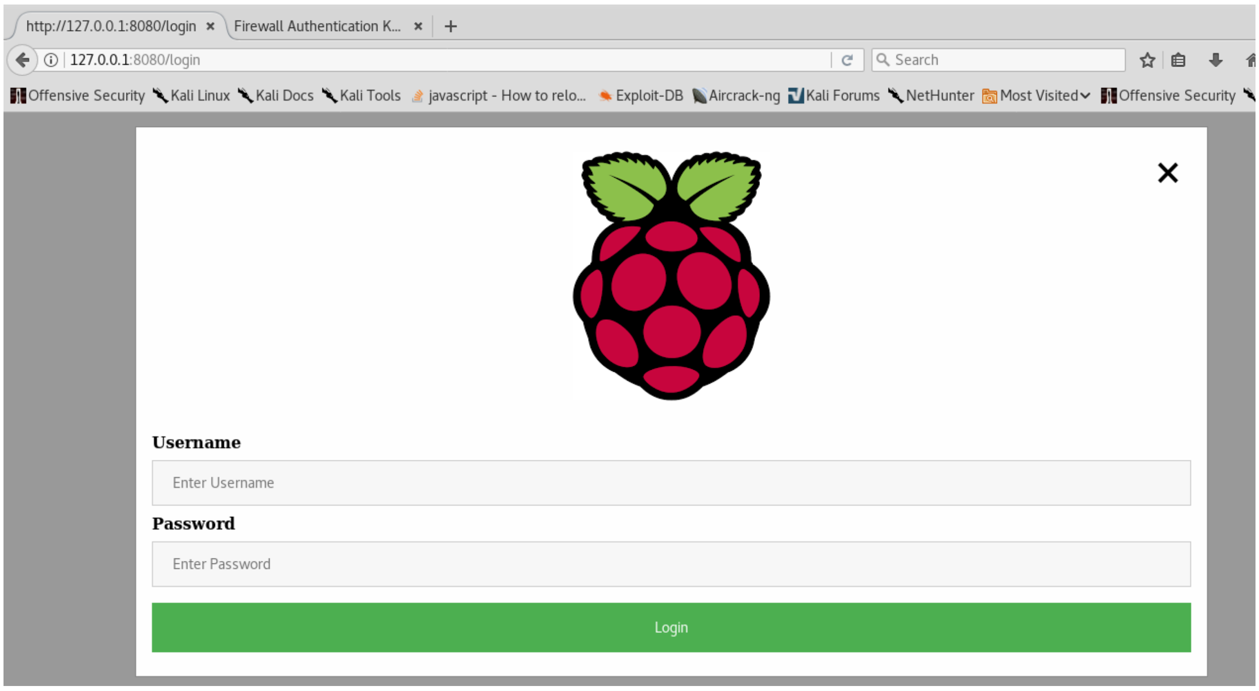
## Administrative Panel for Network administrator: -

This part of the firewall is for the organization implementing this application. It guides the person the who is responsible for the care-taking each and every function of this firewall. It provides all the functionality required to setup the different modules of this firewall.

Using this, the network administrator can have complete command over every aspect of the firewall. He/she can govern and configure different settings or functionalities required for the successful implementation of a firewall within an organization. It is a Graphical User Interface for shows the structure and schema of the data base along with other configurational settings involved.

## How does it work:

1. It runs on the port 8080 of our server furnished with a login/signup mechanism to provide access to legitimate and authenticate persons only.'



**Username**

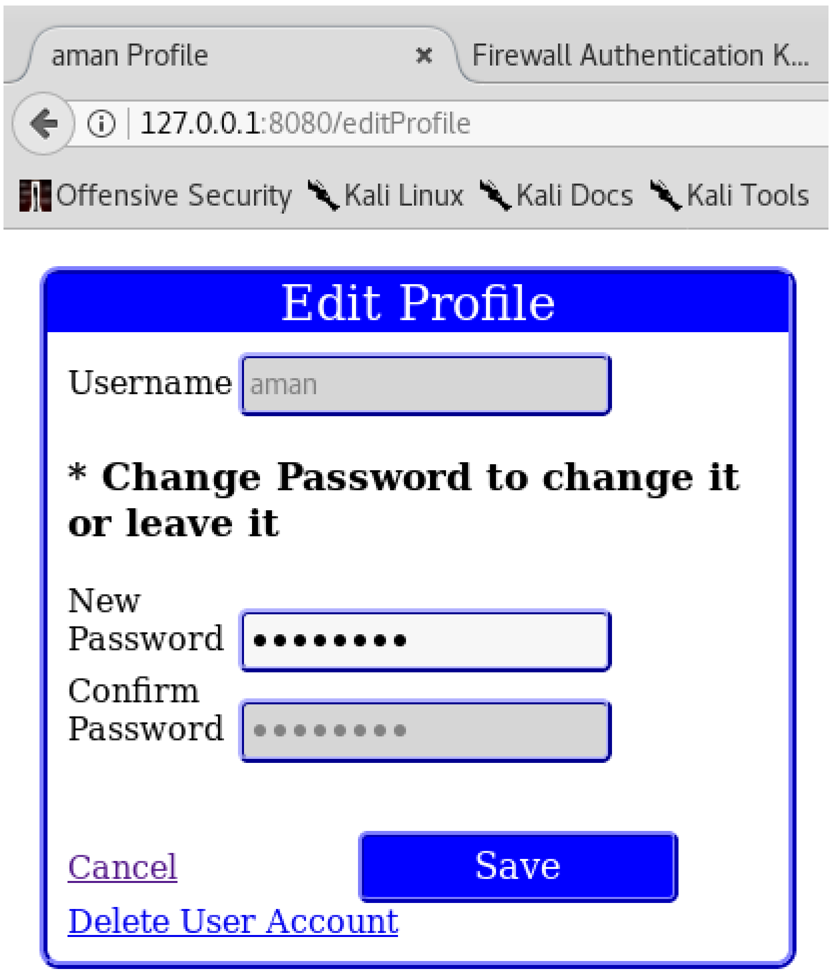
Enter Username

**Password**

Enter Password

Login

2. The admin needs to login if he/she is registered earlier as an admin, else signup as an admin.
3. After login admin can edit his/her profile.


aman Profile
Firewall Authentication K...
127.0.0.1:8080/editProfile
Offensive Security  Kali Linux  Kali Docs  Kali Tools

Edit Profile

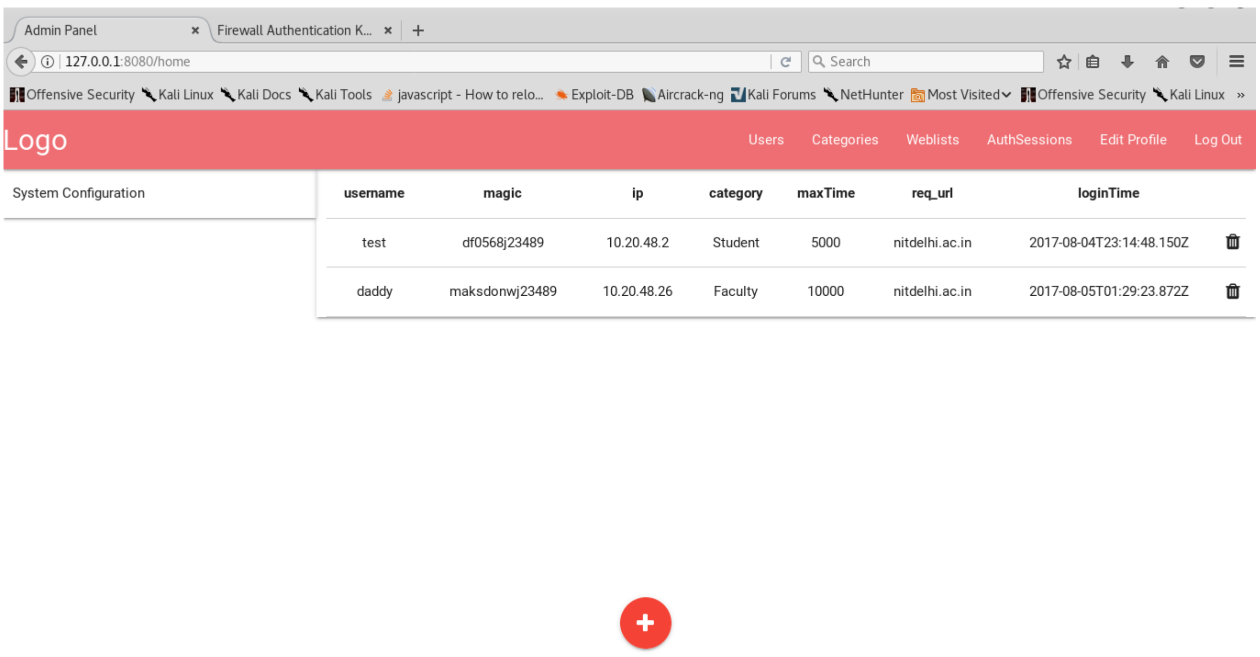
Username: aman

* Change Password to change it or leave it

New Password: ●●●●●●●●●

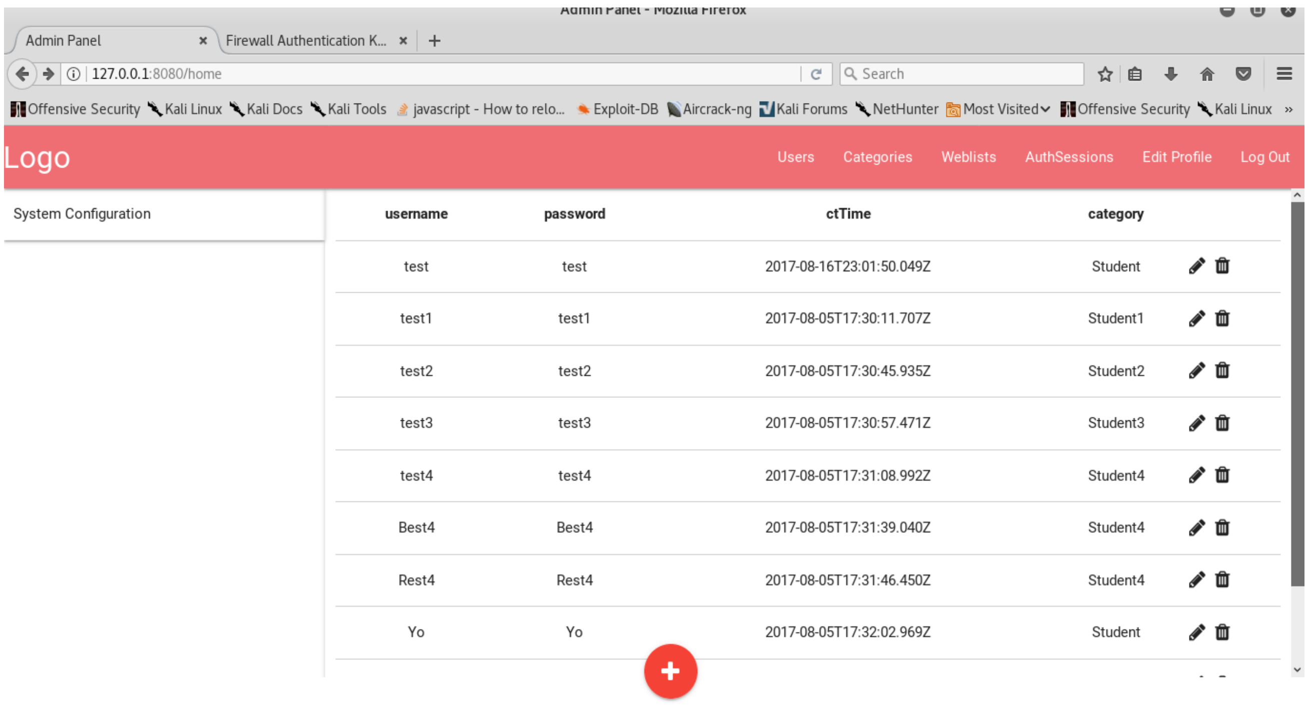
Confirm Password: ●●●●●●●●●

4. Admin can see all the currently active authSession running on the server. For each active user, admin can edit his/her details like category, username, password and the same will be updated in the database as well.



| username | magic | ip | category | maxTime | req_url | loginTime |
|---|---|---|---|---|---|---|
| test | df0568j23489 | 10.20.48.2 | Student | 5000 | nitdelhi.ac.in | 2017-08-04T23:14:48.150Z |
| daddy | maksdonwj23489 | 10.20.48.26 | Faculty | 10000 | nitdelhi.ac.in | 2017-08-05T01:29:23.872Z |

5. Admin can also see all the categories present in the database at present. He/she can edit the category details as required which includes changing category name, authentication time for that category, or any other changes associated with a particular category. He/she can also add a new category or delete a pre-existing one.
6. Admin can also see the weblists present in the database associated to each category name, with the rights of editing/adding/deleting of the enrolled lists in that category.
7. Admin can see the users of the organization and their all the credentials and can edit/add/delete anytime this collection of users.



| username | password | ctTime | category |
|---|---|---|---|
| test | test | 2017-08-16T23:01:50.049Z | Student |
| test1 | test1 | 2017-08-05T17:30:11.707Z | Student1 |
| test2 | test2 | 2017-08-05T17:30:45.935Z | Student2 |
| test3 | test3 | 2017-08-05T17:30:57.471Z | Student3 |
| test4 | test4 | 2017-08-05T17:31:08.992Z | Student4 |
| Best4 | Best4 | 2017-08-05T17:31:39.040Z | Student4 |
| Rest4 | Rest4 | 2017-08-05T17:31:46.450Z | Student4 |
| Yo | Yo | 2017-08-05T17:32:02.969Z | Student |

**Observation:**

Using SQUID proxy server the confidentiality of **http/https** requests can be breached by the firewall administrator and all the data can be logged.

## Appendix

***Please refer the attached folder with this report for all the codes.***

## References :

- **Clam AntiVirus 0.99.1 User Manual**
- **https://wiki.squid-cache.org/ConfigExamples/Intercept/SslBumpExplicit**
- **Squid: The Definitive Guide - By Duane Wessels**
- **https://wiki.squid-cache.org/Features/SslBump**
- **http://ubuntuserverguide.com/2012/06/how-to-setup-squid3-as-transparent-proxy-on-ubuntu-server-12-04.html**
- **icap_whitepaper_v1-01.pdf - By Network Appliance**
- **Iptables Tutorial 1.2.2 Manual - By Oskar Andreasson**

- **Squid Proxy Server 3.1 Beginner's Guide – By Kulbir Saini**
- **Phishing signatures creation HOWTO – By Török Edwin**
- **http://louwrentius.com/static/files/proxy-squid.txt**
- **Creating signatures for ClamAV – Clamav Manual**
- **http://freelinuxtutorials.com/tutorials/squid-proxy-server-tutorial/**
- **https://smoothnet.org/squid-v3-5-proxy-with-ssl-bump/**
- **http://c-icap.sourceforge.net/install.html**
- **http://www.icap-forum.org/documents/specifi cation/rfc3507.txt**