CLICKJACKING

Thesis Report

Submitted in fulfillment of the requirements

For the award of degree of

Master of Engineering

in

Information Security

Submitted By

HONEY PAL

(Roll No. 801333007)

Under the supervision of Dr. A. K. Verma

Associate Professor (CSED Department) Thapar University, Patiala.

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

June 2015

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "Clickjacking", in partial fulfillment of the requirements for the award of degree of Master of Engineering in Information Security submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Anil Kumar Verma and refers other researcher's work which are duly listed in the reference section. The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Honey Pal

Signature

(Honey Pal)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Anil Kumar Verma)

Associate Professor,CSED

Countersigned by

(Dr. Deepak Garg)

Head, CSED

Thapar University, Patiala

(Dr. S.S Bhatia)

DOAA,

Thapar University, Patiala

Abstract

This is an era of digital globalization. With the widening of IT, there is an enormous increase in the number of users on social media networks. The downside of this is that the integrity and confidentiality of the user data are at risk. Clickjacking is one such type of threat and is gaining popularity worldwide. Clickjacking is a widespread attack these days, in which the user click is hijacked in order to perform some action of hacker's interest. Click jacking is possible when multiple applications and websites share the same graphical display. The target page is constructed to lure the victim to click on an object. The click action is made to land on some other object and hence used to perform an action that the victim did not intended. In this work various attack methods such as compromising target display integrity, pointer integrity, temporal integrity and exploiting human perception system have been analyzed and addressed for overcoming clickjacking.

Keywords- Clickjacking, likejacking, Strokejacking, Cursorjacking, iframe.

TABLE OF CONTENTS

List of figures

List of tables

Sr. no.	Table Name	Attributes
1.	USERS_TABLE	USER NAME
		USER ID
		USER TYPE
		ALLOTTED BANDWIDTH
		USED BANDWIDTH
		PASSWORD
2.	BLACKLISTED EXTENSION	EXTENSION
3.	BLACKLISTED WEBSITES	WEBSITES
4.	BLACKLISTED KEYWORDS	KEYWORD
5.	WHITELISTED IP	IP_ADDRESS
6.	CLICKJACKED WEBSITES	WEBSITE
7.	CLICK TIME TO LIVE	IP
		REQUESTED URL
		CLICK TIME TO LIVE
		TIME OF CLICK
		STATUS
8.	RESULTS	WEBSITE
		IP ON WHICH WEBSITE IS HOSTED

Abbreviations

Serial. no	Acronym	Expansion
1	UI	User Interface
2	OWASP	Open Web Application Security Project
3	AppSec	Application Security
4	XFO	XMPC Feature Object
5	PHP	PHP Hypertext Preprocessor
6	HTML	Hyper Text Markup Language
7	CSS	cascading style sheets
8	HTTP	Hypertext transfer protocol
9	HTTPS	Hypertext transfer protocol secure
10	CSRF	Cross site request forgery
11	JSP	Java server pages
12	LAN	Local area network
13	URL	Uniform resource locator
14	IP	Internet Protocol

CHAPTER 1: INTRODUCTION

1.1. Overview

In the era of digital globalization each and every human on this planet is connected to others on a social media network through twitter, facebook, whatsApp, Google+ etc. The personal and private data of the users are available on the Internet. In such a scenario it is very important to maintain confidentiality and integrity of their data. The Dark side is that our networks are not completely secure. A small unawareness and carelessness can give the user data in the hands of the attacker and then the attacker is free to manipulate or misuse it in any form. Among many cyber threats Clickjacking is one such attack that is gaining popularity worldwide and has brought a great concern as far as the cyber security of the web clients is concerned.

In a dynamic web page the content is loaded in different frames that may or may not be of same origin. These contents are loaded into different iframes which share a common graphical display. In such a scenario, the webpage is vulnerable to clickjacking [1, 17] attack also known as UI redressing [2]. The term clickjacking was given first by Robert Hansen and Jeremiah Grossman at OWASP AppSec 2008[16].

The idea behind clickjacking is that the attacker creates a malicious page containing invisible iframes. The user is lured and tricked to click on it. The click is then hijacked and landed onto the target element and some action is performed which is of attacker's interest for example, in likejacking attack [3]. The attacker web page tricks the user to click on a facebook like button when the user clicks on the covering button, the click lands on the like button, then it appears in the user's friend's newsfeed that he has liked the attacker's website. According to the research the new attack variants can have a greater threat. In one case, the user's private data such as user name, password and emails can be stolen. In another case an attacker can spy on a user through their webcam and even turn on their microphone.

In clickjacking the attacker targets mainly the visual and temporal context of a web page. Visual context deals with the display that a user can see and intercept on a web page. Temporal context is concerned with the time at which a user's action is performed.

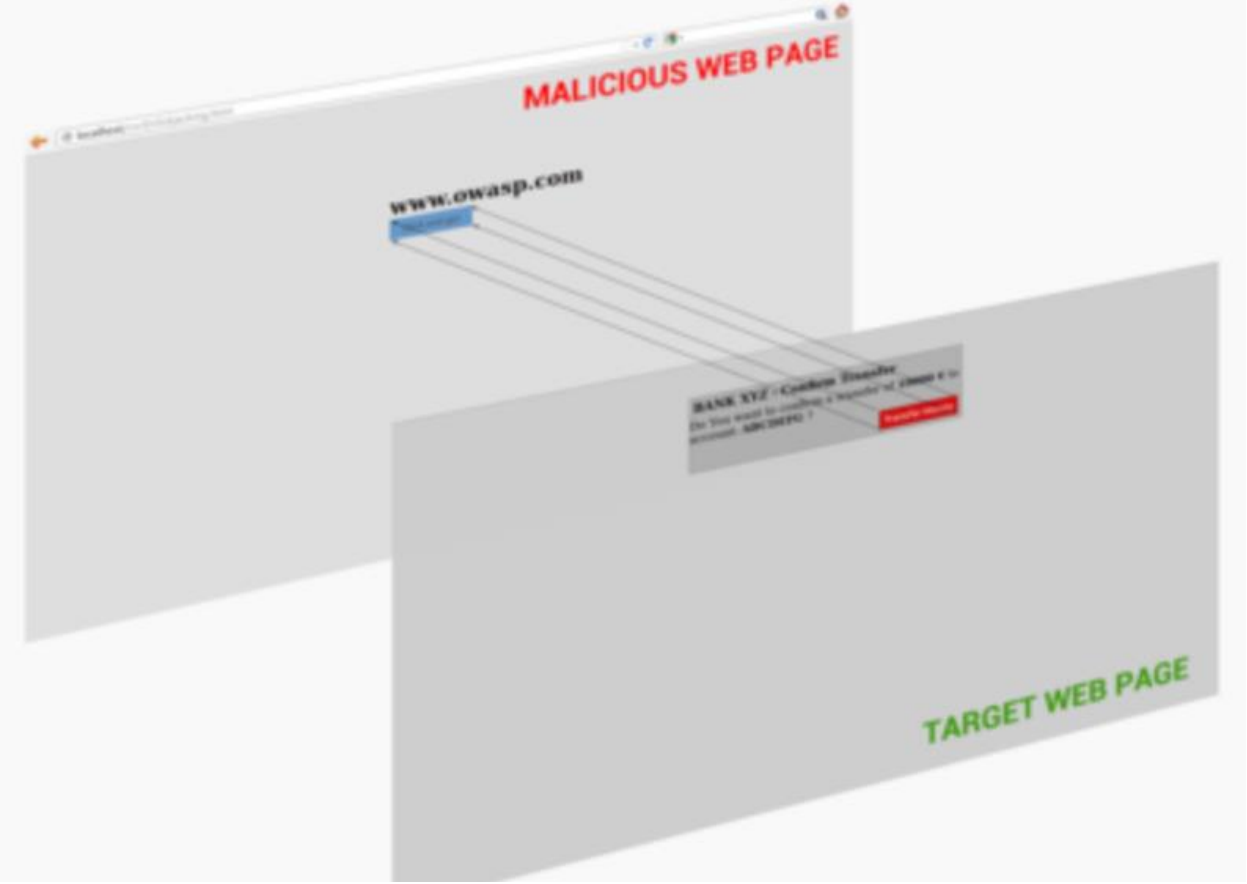


Figure 1.1 Clickjacking web page structure

The exploit process is explained in following diagram.

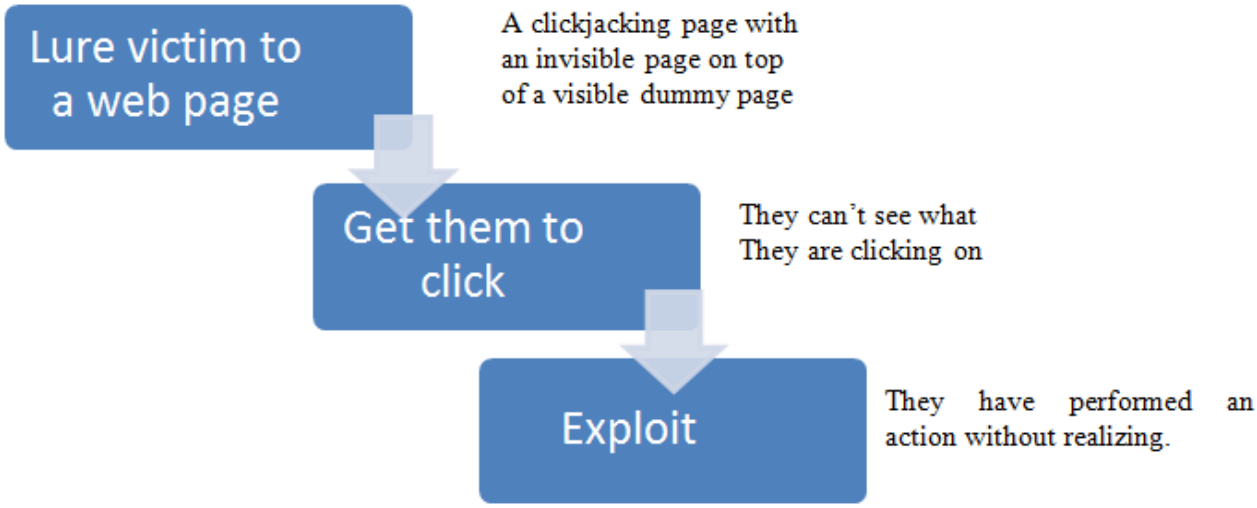


Figure 1.2 Clickjacking exploit process

1.2. Motivation:

The topic was selected due to keen interest in cyber threats. Cyber threats were taken as the base theme. Most of the researchers worldwide take one basic theme and then shrink their research area into a sub-topic. Cyber threats can be broadly classified as follows:

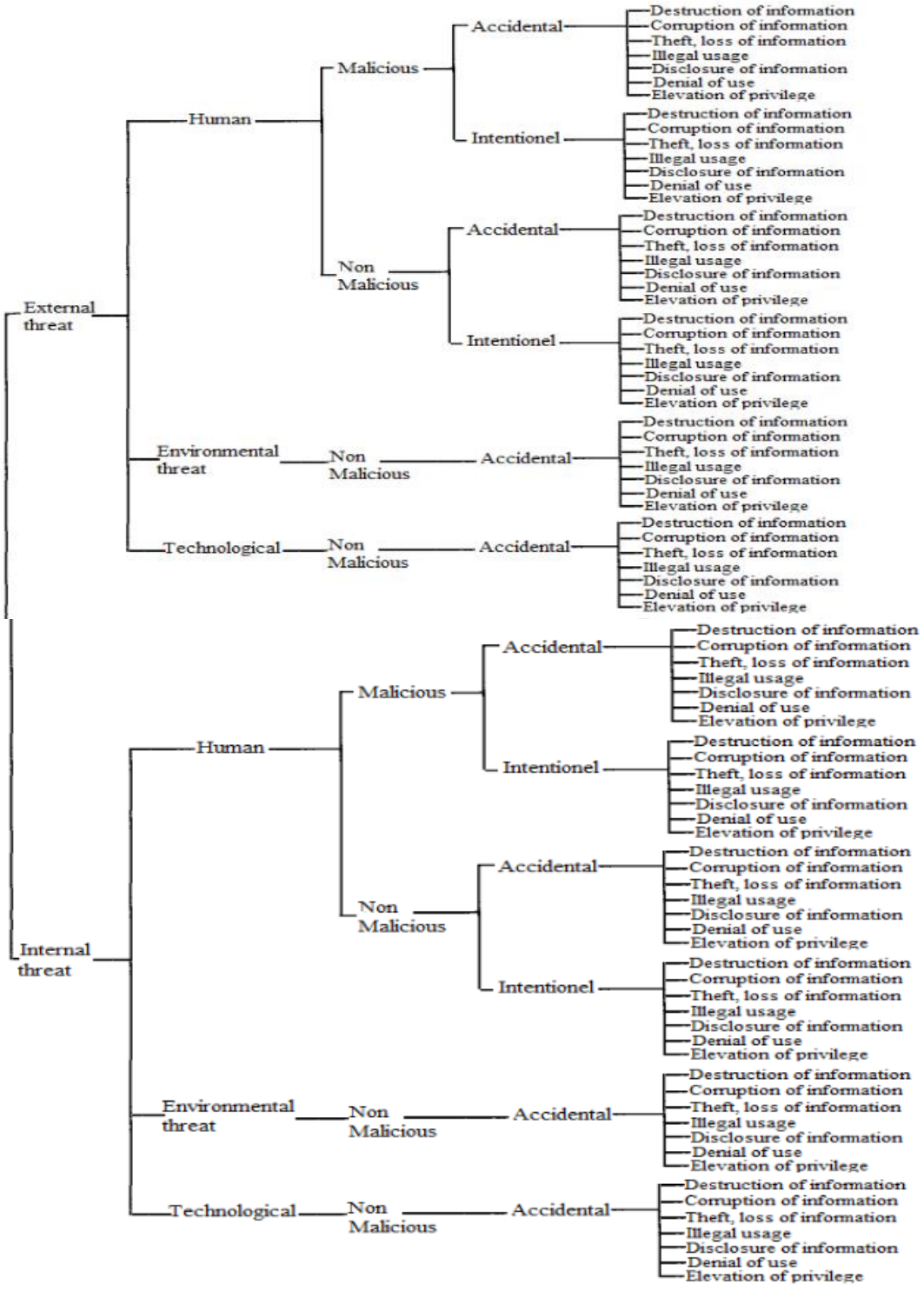


Figure 1.3 web attacks classification

Clickjacking covers a wide range of these types. By hijacking the user clicks the attacker can perform information theft, disclose information, illegally use the information, etc.

1.3. History

Loading a web page inside another web page has always been a security concern. Few years back the web based attacks consisted of nasty websites that loaded legitimate websites into a frame and used JavaScript "key loggers" to steal login credentials. Such attacks are so called Cross Frame Scripting. To thwart this attack login pages of secure sites started implementing "frame busting" [7] also called frame killers using which their pages can be prevented from getting encapsulated in web page of other sites. The following JavaScript code would detect if the page was loaded in another frame, and set itself as the "top" frame.

```
<script type="text/JavaScript">
        if(top != self)
        top.location.replace(location);
</script>
```

With time another risk emerged. A page could be framed, but a malicious webpage can completely be stacked over it. While the malicious parent frame could control the entire display shown to the user, it could also trick users into clicking parts of the hidden child frame. For example a JavaScript game of shooting targets by clicking a mouse, however the user would actually be clicking objects on the web application encapsulated in the frame. This is known as Clickjacking, or a "UI Redress Attack". The term clickjacking was introduced first by Robert Hansen and Jeremiah Grossman at OWASP AppSec 2008.Clickjacking hijacks user clicks to carry out actions that the user did not intend to do.

1.4. Composition

In programming click is an event that leads to some action i.e. when the user makes a click on an object an event listener listens to this event and then performs some action. The action is performed on the execution of the JavaScript code for that event. The events can be as follows;

1. Pressing a button
2. Moving your mouse over a link
3. Submitting a form etc.

Figure 1.4 Event handling on click.

There are a number of possible actions that can be performed on a click.

These actions are listed as below.

1. Submission of a form
2. Loading some other web page
3. Downloading of files
4. Uploading of a file
5. Permitting to access your system resources etc.

1.5. Vulnerabilities

Various programming languages of the web such as html, JavaScript, PHP etc. contain various programming flexibilities. It was never thought that such pliability would be exploited by the hackers to trap the victims and get their tasks done.

Some of them are mentioned below.

1. **IFRAME**: A webpage can contain some other webpage. Example is Google map.

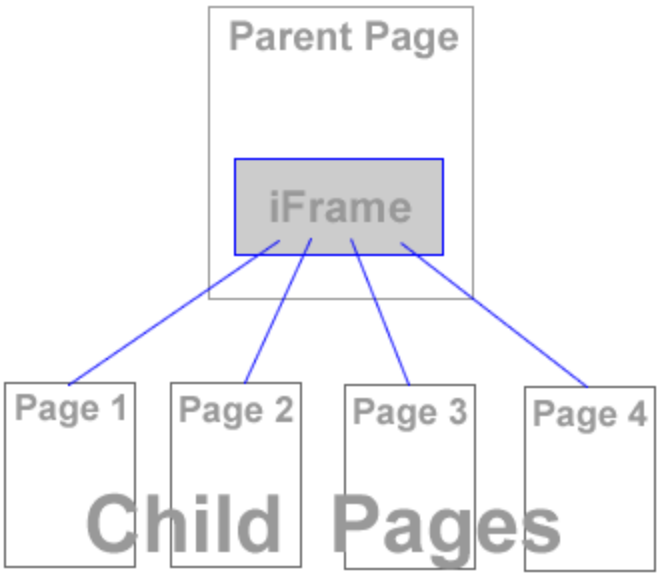


Figure 1.5 iframes in a webpage.

Sample code

```
<iframe width="600" height="450" frameborder="0" style="border: 0"
  src="https://www.google.com/maps/embed/v1/place?key=API_KEY
&q=Space+Needle, Seattle+WA">
</iframe>
```

2. **OPACITY:** HTML elements can be opaque, partially opaque or even invisible.

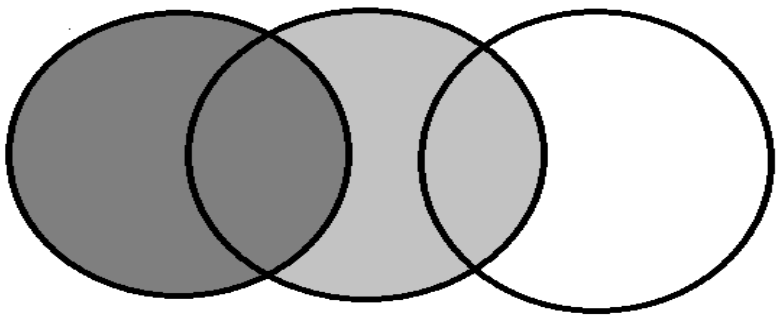


Figure 1.6 opacity

3. **STACKING ORDER:** elements can be stacked on top of one another.

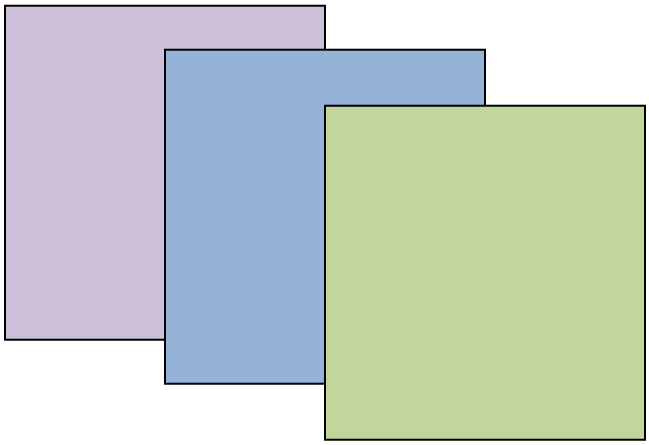


Figure 1.7 Stacking of objects in web page

4. **STACKING +OPACITY:** an element can be on top and invisible.

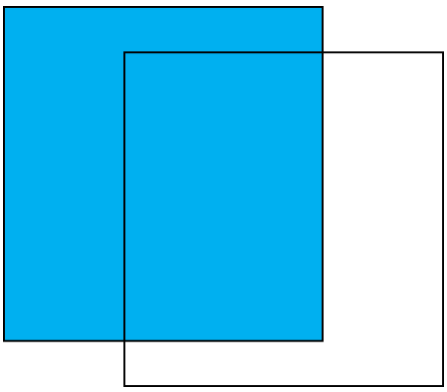


Figure 1.8 stacking +opacity

Consider the following code

```
<Html>
        <h1 style="text-align: center">claim your free nokia lumia</h1>
                <p style="font-size: 40px ;"> you are very lucky!! </p>
                <div style="z-index: 10; opacity: 0; position: absolute; top: 0px; ">
                <iframe scrolling="no"  style="width: 800px; height: 500px;"
                        src="http://www.xyz.com"> </iframe>
                </div>
```

```
<div style="position: absolute; top: 200px; left: 210px ;">
<a href="#">Claim  your prize</a>
</div>
</html>
```

Z-index puts the iframe on top.

Opacity makes the iframe invisible.

Position: absolute lines up the iframe with the dummy page.

Apart from all these vulnerabilities the biggest vulnerability is a naïve user who has very little knowledge of security on web.

5. Human perception

Clickjacking attacks human perception system [28]. The user sees an object on the web page perceives it and then takes some action. This takes place through the sensor motor system.

CHAPTER 2: LITERATURE RIVIEW

2.1. Introduction

A clickjacking attacker is well equipped with all the tools for a web attack [4].

1. possess a domain name and control content of their web servers,
2. Can make a victim visit their site, thereby rendering attacker's content in the victim's browser, when the target victim visits the attacker's page which has hidden sensitive UI element visually or temporally, he is lured to perform unintended action.

2.2. Classification of existing attacks

With time a number of attack variants have emerged. Clickjacking is broadly classified into following;

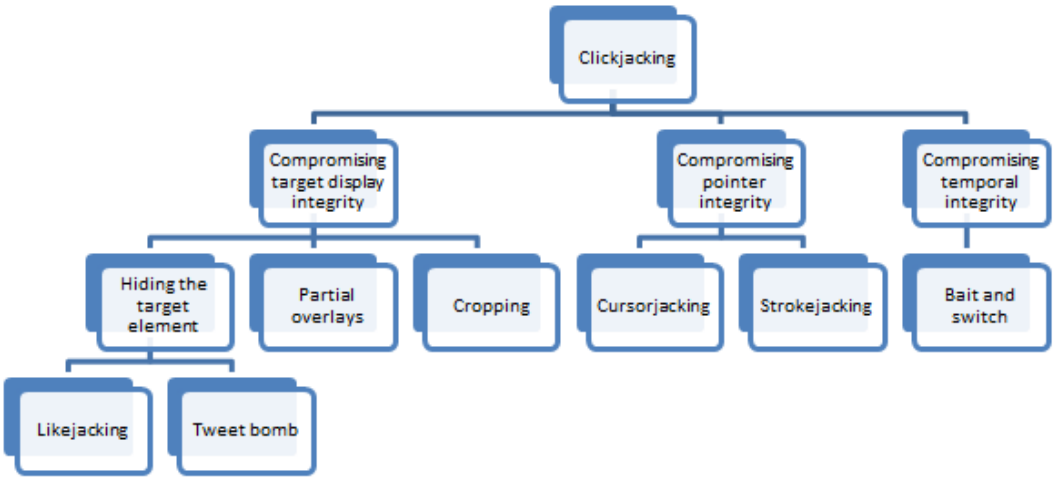


Figure 2.1 classification of clickjacking attacks

2.2.1. Compromising target display integrity

Attacker creates an illusion for the victim by displaying a legitimate object over a target object .The victim thinks he is clicking on the object he sees but actual click lands on the target. This can be performed in either of the following ways.

2.2.1.1. Hiding the target element

The attacker hides the target element by making it transparent and routs the mouse click onto this element .The attacker achieves this by wrapping it in div container with a CSS opacity value set to zero. To tempt the victim to click on it the attacker draws a decoy under the target element by using a lower CSS z-index [4].

Another approach is that the attacker places the decoy directly above the target element and make the decoy unclick able by setting the CSS property pointer-events: none [4].Thus the victim's click will then land on the target.

2.2.1.1.1. Likejacking

Likejacking [3] is a clickjacking predominant on Facebook. Facebook attackers present a web page that contains two iframe stacked over one another. The lower frame is designed with a Facebook "Like" button configured to follow victim's mouse cursor. The upper frame shows some attractive content to lure the victim to launch a click. No matter where the victim clicks on the webpage, the click is landed on the Facebook Like button and further spreading the spam.

Attackers use likejacking as a money earning scheme through affiliate marketing. Affiliate marketing pays the affiliate for every person who views an ad, signs up for a service or registers on a given site. One reason this attack works is that Facebook does not require any confirmation when you click the Like button. Though confirmation would not entirely prevent the attack, it would complicate the attack and potentially discourage its active exploitation.

2.2.1.1.1. Tweet bomb

Twitter bombing [5] is the use of trending topics or popular Twitter hash tags to direct people to unrelated websites or products. Many political groups have taken advantage of this tactic to trash opposing candidates. Companies use popular trending topics from around the world to attach links to their tweets and drive more people to their websites.

The key in Twitter bombing is that multiple accounts (sometimes thousands) are created to overtake the normal trending results. The not-so-accepted way of Twitter bombing involves creating multiple dummy accounts and sending a large number of tweets in a short period of time. This method is usually pretty irritating for Twitter users who are actually trying to use the social media site to gain specific information.

2.2.1.2.Partial overlays:

The victim is confused by concealing a part of the target element [8, 9]. For example attacker can overlay his own information on top of a PayPal checkout iframe to cover the credentials of the victim but keep the pay button intact. This can be achieved using CSS z-index or using attacker's pop up window [6].

2.2.2. Compromising pointer integrity

The attacker exploits the pointer integrity by displaying a fake cursor icon. The victim is confused and he then misinterprets the cursor .Another prominent method is to display a blinking cursor in a text field through programming. The victim clicks in the text field multiple times and his click is hijacked by the attacker.
There are basically two types of attack variants in this.

2.2.2.1. Cursorjacking

Cursorjacking [22, 18] is an attack in which attacker displays a false cursor away from the actual mouse pointer. The victim will have a wrong perception of the current location of the cursor. The attacker achieves this by exploiting the CSS cursor property to hide default cursor and programmatically draws a fake cursor at some other location [10]. An alternative method is to set a custom mouse cursor icon to a misleading image that has a cursor icon shifted a few pixels away from the actual spot [11, 18].

2.2.2.2. Stroke jacking

Stroke jacking [12] attack variant involves a blinking cursor which asks for a keyboard input. The attacker can utilize this technique by embedding the target element in a hidden frame. When the victim is typing the attacker momentarily switches keyboard focus to the while asking the victim to type a text into a misleading input field which is controlled by target element, thus the victim thinks he is typing in attacker's field but actually his keystrokes are landed onto the target field.

2.2.3.Compromising temporal integrity

In this attack variant the UI is manipulated when the user decides to click but before actual click has been launched, the attacker captures the mouse hovering event and when the click is just about to launch he quickly swaps the positions of the target element and the decoy element. To increase the probability of success of the attack the attacker may ask the victim to click multiple times or double times. For example such attacks are mostly utilized in online games that involve multiple clicks.

2.3. Mechanism

2.3.1. Compromising target display integrity

Exploit process for facebook

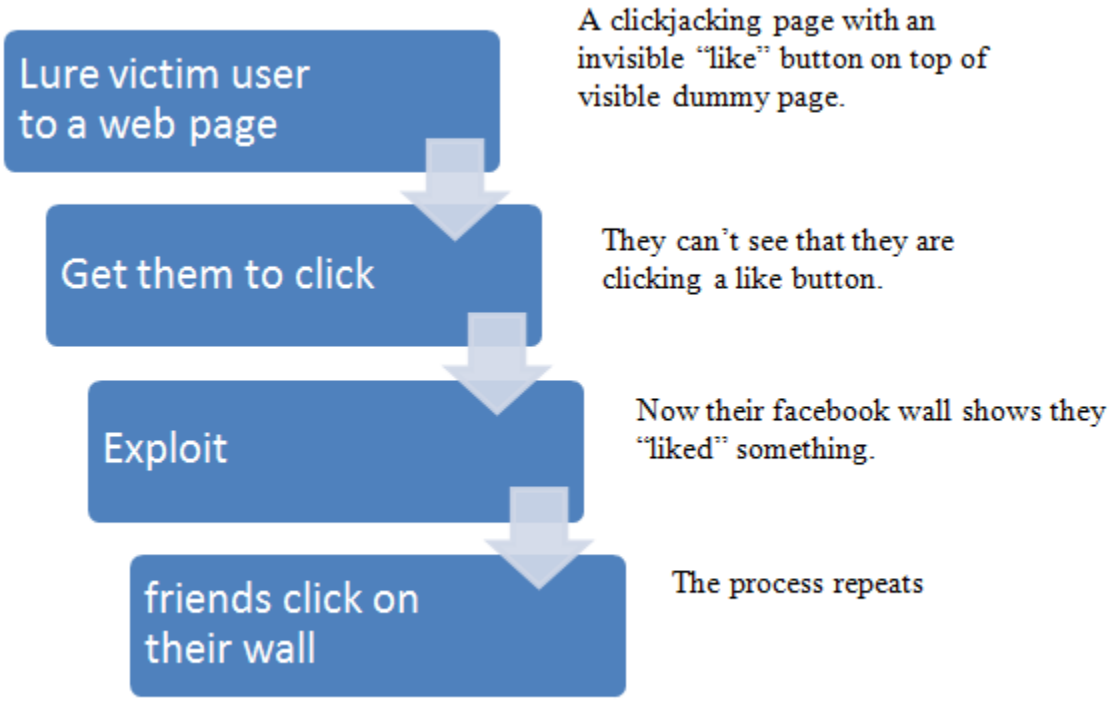


Figure 2.2 facebook exploit process

Sample code:

```
HTML web page with clickjacking (file: trustedPage.html)
<! DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<Html>
<Head>
<Title>Trusted web page</title>
</head>
<Body>
<h1>www.nds.rub.de</h1>
<form action="http://www.nds.rub.de">
```

```
<input type="submit" value="Go">
</form>
<! -- Iframe Code -->
<iframe id="clickjacking" src="clickjacking.html" width="50"
height="300"
Scrolling="no" frame border="none">
</iframe>
<style type="text/css"><!--
#clickjacking {position: absolute; left: 7px; top: 81px;
opacity: 0.0}
//--></style>
     </body>
</html>
```

2.3.1.1. Likejacking

The victim visits the attacker website. The attacker has completely covered the like button with the claim your free iPad button and set it unclick able through programming. The victim clicks the like button and his click lands on the like button.

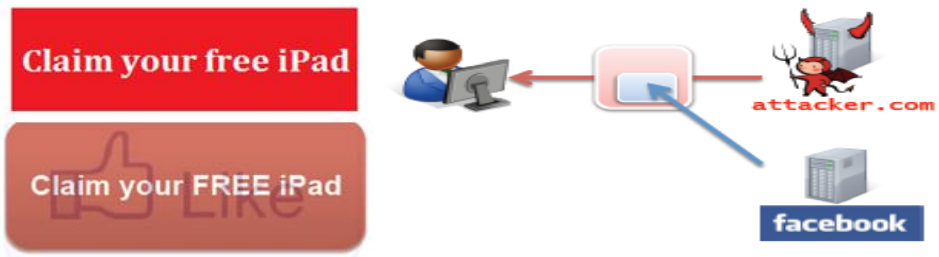


Figure 2.3 likejacking

The overall idea is simple.

1. A visitor is lured to evil page. No matter how. “Click to become a millionaire” or whatever.

2. The evil page puts a “get rich now” link with z-index=-1.
3. The evil page includes a transparent iframe from the victim domain, says `facebook.com` and positions it so that “I like it” button is right over the link.

Here’s how it looks (half-transparent iframe for demo purposes):

```
<Style>
Iframe {/* iframe from facebook.com */
  Width: 300px;
  Height: 100px;
  Position: absolute;
  Top: 0; left: 0;
  Filter: alpha (opacity=40); /* in real life opacity=0 */
  Opacity: 0.5;
}
</style>

<Div>Click on the link to get rich now :< /div>

<Iframe src="/files/tutorial/window/clicktarget.html"></iframe>

<a href="http://www.google.com" target="_blank" style="position:
relative; left: 20px; z-index:-1">CLICK ME! </a>

<Div>you’ll be rich for the whole life! </div>
```

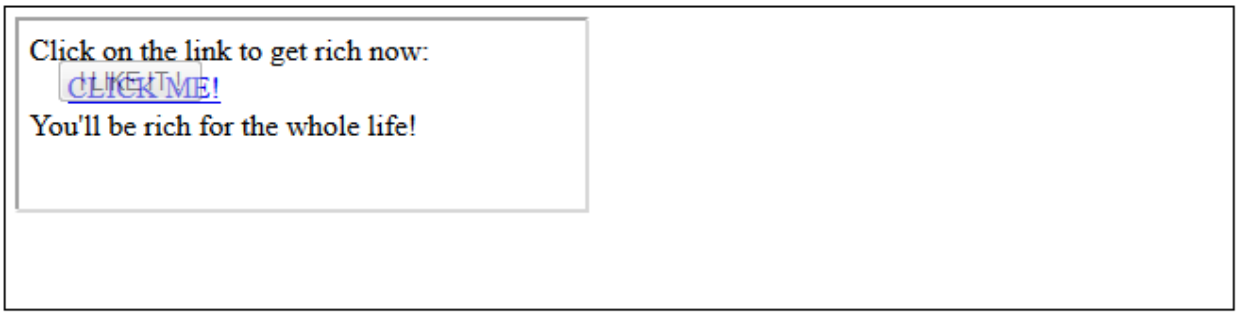


Figure 2.4 likejacking web pages

2.3.2. Compromising pointer integrity

2.3.2.1. Cursorjacking

The attacker creates a fake cursor pointing the decoy button (claim your free iPhone). The actual cursor is on the like button. The victim gets confused and when he launches a click, the click lands on like button.

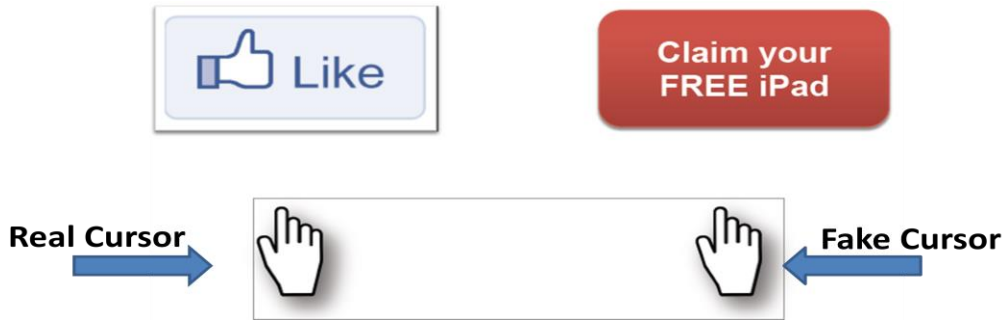


Figure 2.5 Cursorjacking

2.3.3. Compromising temporal integrity

Bait and switch: a mouse comes near "claim your free ipad" button, like moves to its location before the user realizes it.

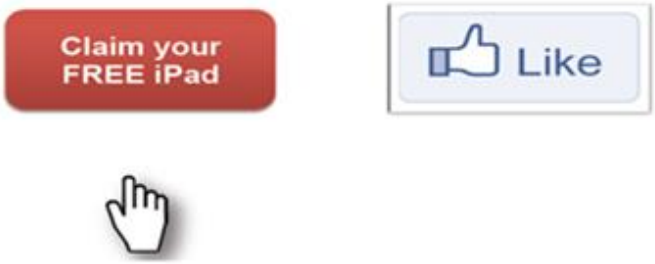


Figure 2.6 Bait and Switch

2.3.4. Clickjacking through online gaming

In this attack variant the attacker creates a dummy web page which loads an online game. Above this dummy page the attacker places a transparent facebook web page .The adjustment of play button is such that it lies just below the like button. Once a naïve user clicks on the play button, the click lands on to the like button and it appears in the newsfeed of the friends of the victim that he has liked a web page. Now, the friends will also do the same mistake and fall in trap.

Figure 2.7 clickjacking through online gaming

2.4. New attack variants

2.4.1. Cursor spoofing attack to steal webcam access

Cursor spoofing [4] involves a fake cursor is programmatically displayed to provide false feedback of the pointer location to user in which the fake cursor gradually shifts away from the hidden real cursor while the pointer keeps shifting. A loud video or audio plays automatically, thus forcing the user to click on skip this add button. The real click is made to land on the target element that allows webcam access to the attacker. This is achieved by utilizing CSS cursor property by setting it to none.

The following image illustrates this.

Figurer 2.8 Cursor spoofing attack

2.4.2. Double-click attack to steal user private data

This is achieved through bait and switch double click attack. The attacker lures the victim to perform a double click on a decoy element. After the first click, the attacker switches in the Google OAuth pop-up window [4] under the cursor right before the second click. This attack can steal a user's email and other private data from the user's Google account.

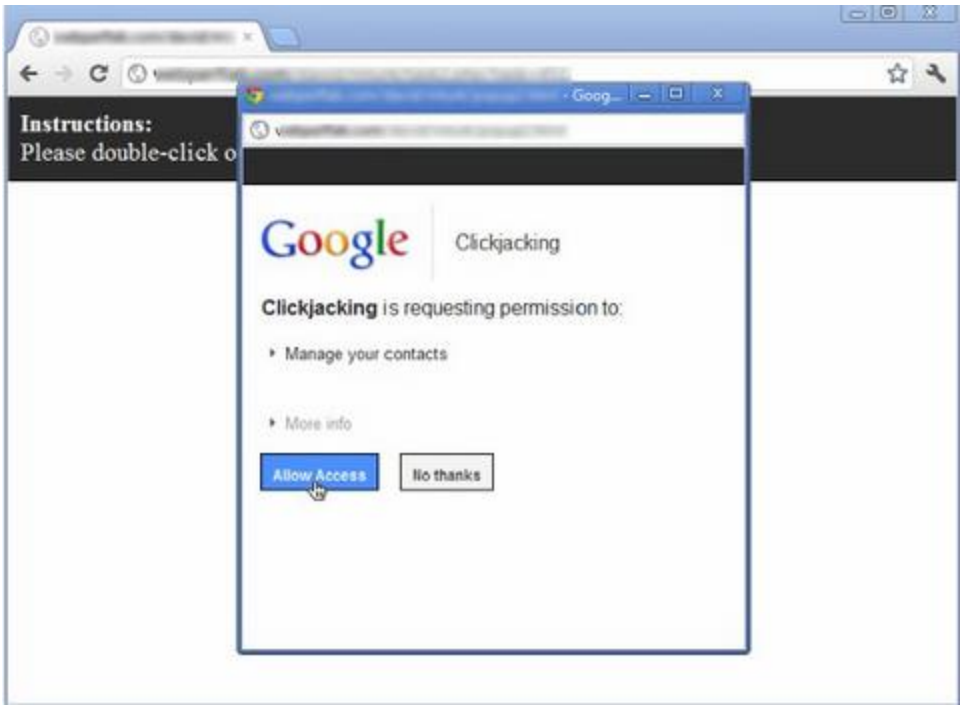


Figure 2.9 Double click attacks

2.4.3. Whack a mole attack to compromise web surfing anonymity

In this attack the victim is asked to play a whack a mole game [4] and to lure them it is displayed that they will earn a reward by clicking on the showed objects. Throughout the attack the attacker displays a fake cursor to the place where the user's attention should be. At a later time, suddenly the clickable object is replaced by a facebook like button and the victim unintentionally hits the like.

The following image successfully makes it clear.

Figure 2.10 whack a mole attack

2.5. Advanced attacks:

2.5.1. Bypassing CSRF using clickjacking and HTTP parameter pollution[2]

The twitter clickjacking is a famous example for this where submission form is involved using 'GET' request. Another method can be submitting POST method -based forms with attacker controlled data using Clickjacking. This works on JSP applications and partially on ASP.NET applications.

Let us take the case of a simple Email ID update form. They are simple but extremely important, if an attacker somehow manages to force a victim to update his primary Email ID with that of the attacker's Email ID then the attacker can perform a password reset and compromise the victim's account. A sample Email ID updates form containing a 'csrf-token' parameter for CSRF protection:

```
<form method="POST">
<input type="text" name="email" value=""></input>
<input type="hidden" name="CSRF-token" value="a1a1a1a1a1a"/>
</form>
```

Let's say this form is available at 'www.testing.com/updateEmail.jsp'.Since this form does not contain an 'action' attribute, on submission the form will be submitted to the current URL in the address bar, which will be 'www.testing.com/updateEmail.jsp'.The source code of 'updateEmail.jsp' would typically look like this:

```
If (request. Parameter ("email").is Set () &&
request.parameter("csrf-token").isValid() )
{
//process the form and update the email ID
}
else
{
```

```
//display an empty form to the user (CSRF token included)
}
```

The application checks if the request contains a valid CSRF token, if not it displays the form to the user. Now to submit our sample form using Clickjacking the attacker can include an iframe like this:

```
'<iframe
src="http://www.testing.com/updateEmail.jsp?email=devil@attacker
semail.com">'
```

When this request goes to the server the application would display the update form. When this form is submitted by the victim using Clickjacking the request that is sent to the server is like this:

```
POST /updateEmail.jsp?email=devil@attackersemail.com HTTP/1.1
Host: www.example.com

email=&CSRF-token=a1a1a1a1a1
```

Since the form was not filled by the victim, the email parameter in the POST body is blank. However since the action attribute of the form was empty the form is submitted to www.testing.com/updateEmail.jsp?email=devil@attackersemail.com. Now the Query String contains the attacker entered value for the 'email' parameter.

This request contains two values for the 'email' parameter, one in POST body and one in Query String. Enter HTTP Parameter Pollution, when the server side JSP code calls request. Parameter ("email"), the value that is returned is the one in the Query String and not the POST body. Since this value can be controlled by the attacker he can trick the victim in to updating his account with the attacker's mail ID. This attack can also work in cases when the form is submitted with JavaScript like this:

```
<form onSubmit=process ()>
<input type="text" name="email" value=""></input>
<input type="hidden" name="csrf-token" value="a1a1a1a1a1a">
</form>

<script>
function process ()
{
//check if email is set
form.action = document.location; //document.location will give
out the entire URL with parameters
form.method = "post";
form.submit();
}
</script>
```

Similar attack is also possible on ASP applications where the form element is of the form described earlier and if it is submitted over 'GET'.

2.5.1. Event jacking: In event-jacking [21] attack variant a web page contains clickable images. The attacker utilizes the “onlick” event handler and quotation marks .Using an invisible iframe the attacker can force the victim to perform the click. Thus the attacker can force the victim to launch an event of his interest without the knowledge of the victim.

2.5.2. Text injection by drag and drop: In this type of attack text can be dragged from one domain to the other domain. The attacker uses this loophole to fool the victim to drag something.

Sample code:

```
<div draggable=”true” on drag start=”some event”>
     <h1>Drop me</h1>
</div>
<iframe src=”http://www.google.com”></iframe>
```

2.5.3. Content Extraction: It is possible for an attacker to retrieve data of one web page and send it to some other web page text area.

2.5.4. Pop-up blocker bypass: Browsers these days have inbuilt pop-up blocker plugins. On blocking these advertisement pop ups the browser displays an alert. The attacker can force the victim to open multiple pop-ups through his hijacked clicks.

See the following code example:

```
<Script>
Function mk_popups ()
{
For (k=1; k<10; k++)
      {
      Window.Open('pop.html','spam_no'+I,'width=40,
      height=50');
                  }
        }
</script>
<Body>
<a href="#" onclick="mk_popups ()">spams</a>
```

2.6. Existing defenses

The most common methods for preventing clickjacking are:

1. Client side protection: frame busting
2. Server side protection: X-Frame options

2.6.1 Client side protection: Frame Busting

It is implemented by using a script in each web page that is not to be framed in iframe. This prevents a site from functioning when loaded inside an iframe.
The script is composed of a "conditional statement" and a "counter-action". For this type of protection, there are some workarounds that fall under the name of "Bust frame busting" which will be described in the later sections.

Sample code:

```
<Style> html {display: none ;} </style>
<Script>
   If (self == top) {
       document.documentElement.style.display = 'block';
   } else {
       top.location = self.location;
   }
</script>
```

2.6.2 Server side protection: X-Frame options

Implemented by Microsoft, X-Frame options consist of a header based defense. The header is sent from the server as a part of HTTP responses and is used to mark web pages that shouldn't be framed. This header can contain either of the values DENY, SAMEORIGIN, ALLOW-FROM origin, or non-standard ALLOWALL. Recommended value is DENY.

The "X-FRAME-OPTIONS" is a very effective solution, and was adopted by major browser, but there are some limitations too which will be described later.

```
<system.webServer>
  ...

  <HttpProtocol>
    <CustomHeaders>
      <add name="X-Frame-Options" value="SAMEORIGIN" />
    </customHeaders>
  </httpProtocol>

  ...
</system.webServer>
```

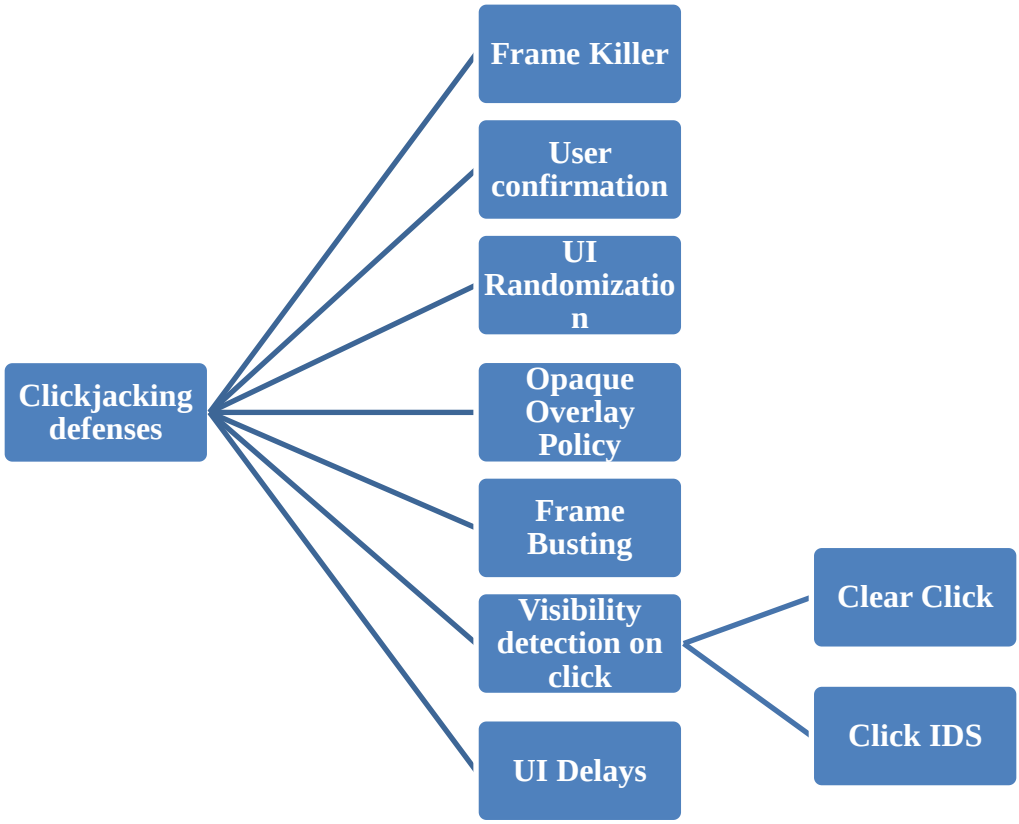


Figure2.11. clickjacking defenses

2.6.3 Protecting visual context

2.6.3.1 User confirmation

When the victim clicks, he is prompted for a confirmation. Facebook currently uses this mechanism for the like button, asking for user confirmation whenever request comes from blacklisted domain [13]. Unfortunately this technique degrades user experience and it is also vulnerable to double click attacks.

2.6.3.2 User interface randomization

The element's UI layout is randomized. For example instead of having facebook like button at a fixed location it should have a random location. This technique has its own shortcoming as the attacker may ask the victim to click multiple times on different locations to guess the actual position of the like button.

2.6.3.3 Frame busting

Frame busting is the technique that does not allow the target element from rendering in an iframe. This can be achieved by using JavaScript code for XFRAME options [14].but this technique faces compatibility issues.

2.6.3.4 Opaque overlay policy

This technique is used in Gazelle web browser [15].all cross origin frames are forced to be rendered opaquely. Again, this technique is also not very robust.

2.6.3.5 Visibility detection on click

In this technique the transparent frames are allowed to be rendered, but mouse click are blocked if the browser detects that the clicked cross origin frame in partially visible or invisible. For example adobe uses such protection to flash player webcam access dialog in response to webcam clickjacking attacks.

2.6.4 Protecting temporal context

One technique is to add a delay after displaying a dialog, so that user can comprehend any UI changes. The user can not click until the delay expires. But this technique annoys a user.

2.7 List of present solutions to clickjacking

- **BetterAuth:** It is a protocol for authentication of the web clients [22]. It is implemented using JavaScript. It works in two steps that are implemented as two successive protocols. First is the mutual authentication protocol. The browser and the server after having knowledge of the password generate a secret for each session. Next step is the authentication tracking scheme based on the signing of the request. Every new request that the browser generates to the server is signed using some secret. Only such requests are considered as authenticated.
- **BeEF:** BeEF [23] is a tool that is used by the penetration testers to demonstrate clickjacking. It concentrates on the web browser.
- **CLEARCLICK:** CLEARCLICK [24] is a module of no script and it prevents web users from clicking on an invisible element of a web page.
- **ViceROI:** ViceROI catches click-spams in ad networks.
- **PROCLICK:** ProClick [20] is a proxy based solution that sanitizes web traffic from some clickjacking signatures.
- **CLICKSAFE:** Clicksafe [19] is a browser-based tool to provide increased security and reliability against click jacking attacks. Click safe is based on three major components. The detection unit detects malicious components in a web page that redirect users to external links. The mitigation unit provides interception of user clicks and gives educated warnings to users who can then choose to continue or not. Click safe also incorporate a feedback unit which records the user's actions, converts them into ratings and allows future interactions to be more informed. Click safe is predominant from other similar tools as the detection and mitigation is based on a comprehensive framework which utilizes detection of malicious web components and incorporating user feedback. We explain the mechanism of click safe, describes its performance, and highlights its potential in providing safety against click jacking to a large number of users
- **CLICKIDS:** Click IDS is a browser plugin. It detects click events and finds any abnormal click behavior. On successful detection it reports any possible clickjacking attacks.

- **Click juggler:** Click Juggler [25] is tool that is used at the developer end for performing testing of the web applications against any possibility of clickjacking. This is an automated tool and it performs various known clickjacking attacks on the web applications. The developer can then edit the application to fix the security loopholes.
- **Regex Based Code Crawler:** Regex Based Code Crawler[26]

CHAPTER 3: PROBLEM STATEMENT

Clickjacking is a web based attack. It takes advantage of the way web browsers work. Although various solutions are being proposed by the researchers and technical experts but they are not fully implemented. Our research problem is to design an algorithm which will act as a genuine solution to clickjacking and implement it in a network. Such an algorithm will detect any abnormal traffic that matches clickjacking signatures and will prevent clicks from getting hijacked.

3.1. Proof of concept:

If a web page can be framed inside an iframe then it is vulnerable to clickjacking. This clearly means that there is no preventive measure implemented from the developer side. As a proof of concept we tried to frame different website using the following simple html code. If a website's webpage can be loaded into an iframe then that website is vulnerable to clickjacking.

```
<Html>
   <Head>
     <Title>Clickjacking testing page</title>
   </head>
   <Body>
     <p>The Website is click jacked!</p>
     <iframe src="http://www.target.site" width="500"
height="500"></iframe>
   </body>
</html>
```

Using the above code I loaded various websites in an iframe and found that many of them were vulnerable to clickjacking.

Figure 3.1 clickjacking proof of concept

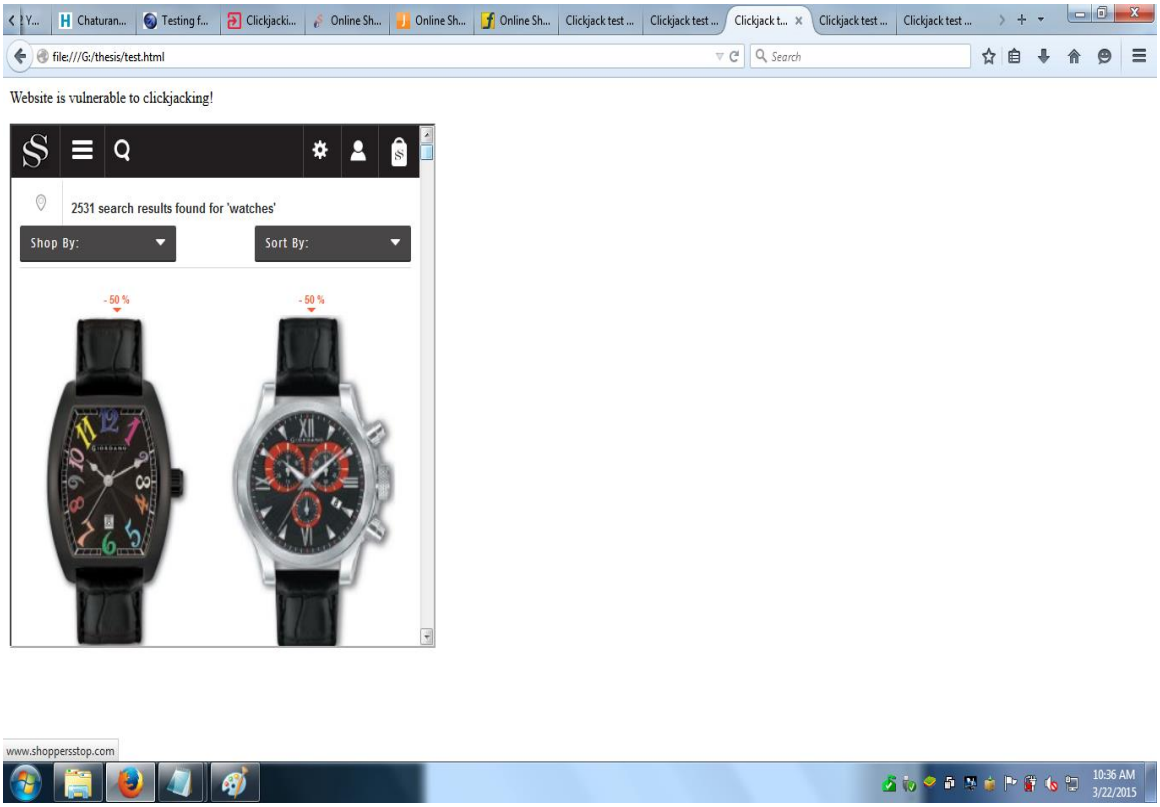


Figure 3.2 clickjacking proof of concept

Figure 3.3 clickjacking proof of concept

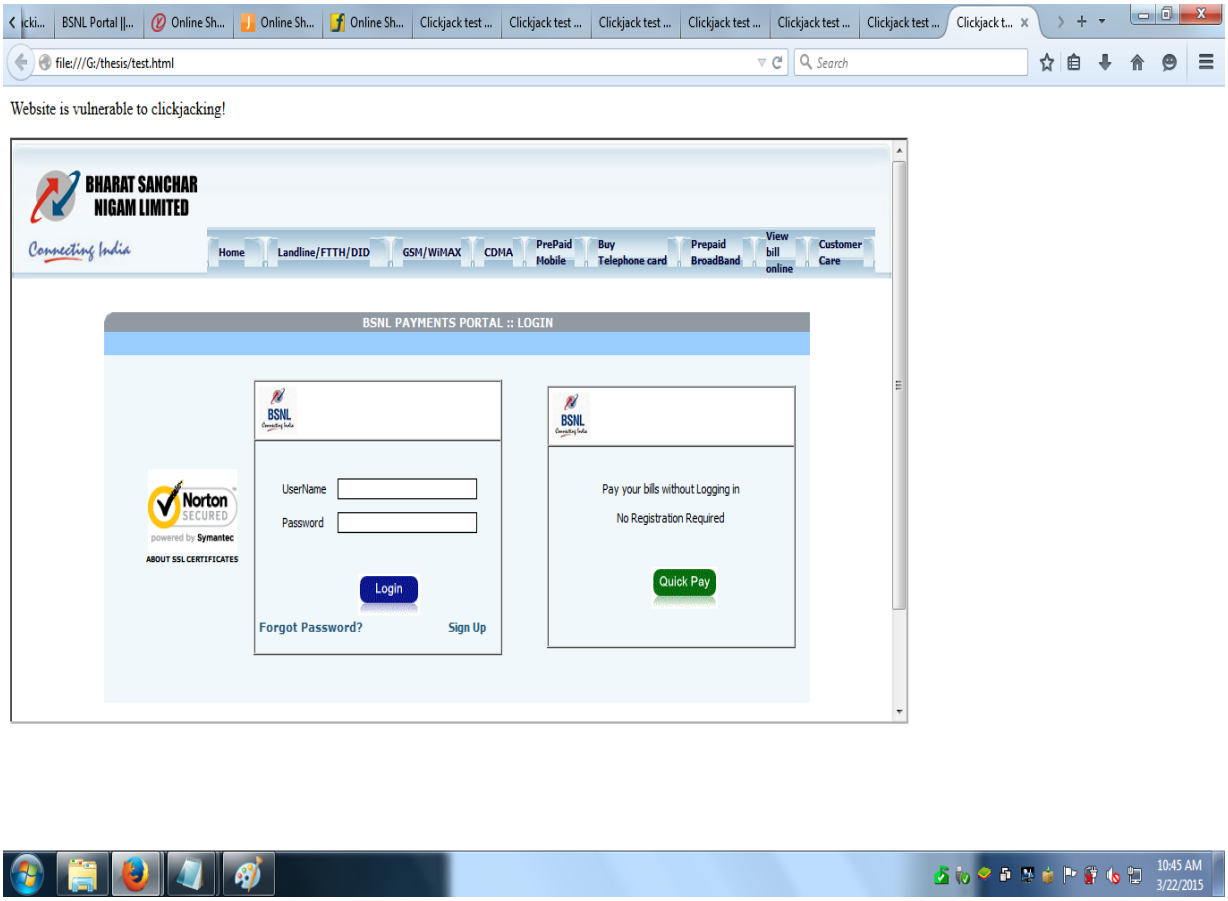


Figure 3.4 clickjacking proof of concept

3.2. Sub-tasks and objectives:

- Study the current research issues in clickjacking.
- Comprehensive study of clickjacking attack vectors and attack variants.
- Choose the research issue among all the solutions proposed so far.
- Propose a genuine solution to thwart clickjacking.
- Simulate the algorithm and compare the results with the existing algorithm and find the improvement percentage.

CHAPTER 4: IMPLIMENTATION

4.1. CAAPS (CLICKJACKING ALERT AND PREVENTION SYSTEM)

4.1.1 Introduction

The software caters to the needs of LAN of an organization.

The proposed system is capable of the following tasks which in comparison to current situation are advantageous to both faculty and students.

This application CLICKJACKING ALERT AND PREVENTION SYSTEM will be installed on the server/Gateway of the LAN. The proposed system will act as a middleman between the client computers and the server machine on the outside network and will shuffle http packets back and forth between the parties.

4.1.2 Access control:

- Provide authentication to users through user name and passwords.
- Provide privileges to the users based on whether it is a student or Faculty.
- Uniformly distribute the bandwidth among various users.
- URL based content blocking [2].
- Block file download for certain extensions.
- Text based content analysis and blocking [2].
- Will give network access only to specific IP Addresses and not to all the computers of the LAN.
- Will allow the users to send and receive both http and https packets.
- It will hide the local user ip addresses from the users outside the LAN.
- We can filter the objectionable content.
- The Administrator can keep a check on the user activities.
- It will act as a kind of audit.

4.1.3 Clickjacking alert and safety:

- It will protect inside client machines from clickjacking [1] and will block websites with possible clickjacking attacks using attack pattern recognition.
- It will keep a record of all the websites with possible clickjacking attacks.
- Keep a time stamp and time to live for user clicks to prevent clickjacking.
- Adds clickjacking prevention parameters to the web content before delivering it to the client machines.
- Enables the network administrator to perform penetration testing of web content based on log.

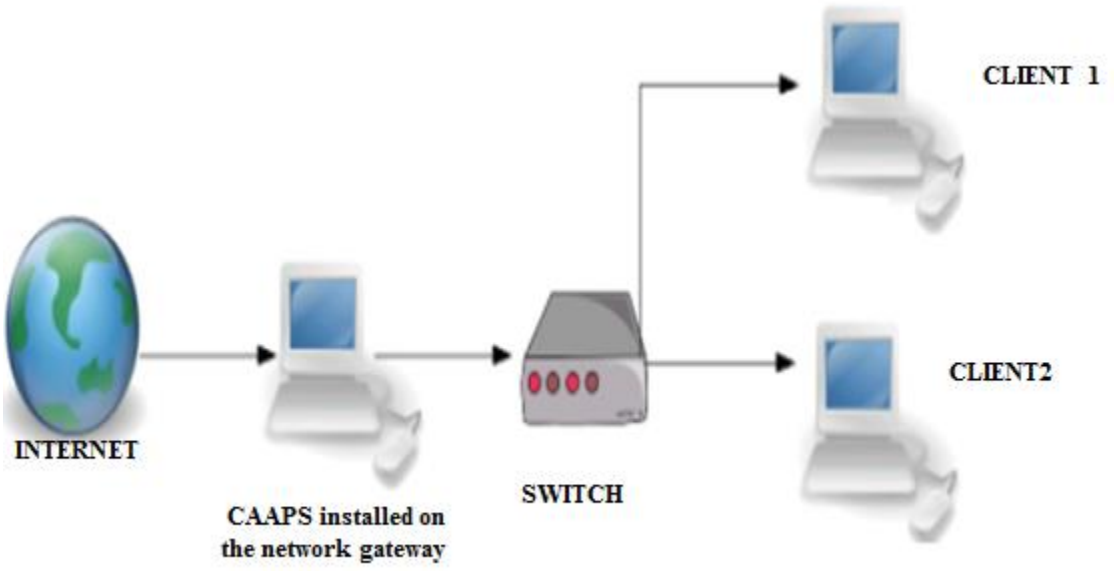


Figure 4.1 Layout for the deployment of the Clickjacking alert and prevention system.

4.1.4 Books used for Programming

- Complete reference to JAVA 1.6. by Herbert Schildt
- O'Reily HTTP definitive guide.
- SAM'S HTTP in 24 hours

4.1.5 Inputs on client side:

URL:

The main input from the client side will be the URL that will be entered by the client in the browser.

The client on **LAN** will fill the URL in the browser window and this is the http request. The browser will break the URL which will be composed of any of the following methods:

1) **GET** Method: The Get method is used for retrieving the data from a web server. It appends the parameters passed as query string to a URL, in the form of key- value pairs

2) **POST** Method: The post method is used for sending data to the server. In post method the query string is appended along the request object, they do not get appended in the URL, so parameters transfer in hidden form.

3) **CONNECT** Method: Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy. Then the proxy server will break the http request to get various http headers.

Example:

HTTP client and an HTTP server running on www.abc.com, port 80.

Client request: http://www.abc.com/index.html

This will be broken in parts like

GET /index.html HTTP/1.1

Host: www.abc.com

Resource: index.html

A client request (consisting in this case of the request line and only one header) is followed by a blank line, so that the request ends with a double newline, each in the form of a carriage return followed by a line feed. The "Host" header distinguishes between various DNS names sharing a single IP address, allowing name-based virtual hosting. While optional in HTTP/1.0, it is mandatory in HTTP/1.1.

Server response:

HTTP/1.1 200 OK

Date: Mon, 13 2005 22:38:34 GMT

Server: Apache/1.3.3.7 (UNIX) (Red-Hat/Linux)

Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT

Etag: "3f80f-1b6-3e1cb03b"

Accept-Ranges: bytes

Content-Length: 438

Connection: close

Content-Type: text/html; char set=UTF-8

A server response is followed by a blank line and text of the requested page.

The client will only give URL as input to the browser on client side that is connected to a gateway computer or a proxy server and then proxy server will forward the request on the internet and give the response back to the client.

4.1.6 Constraints/Conditions

The conditions that are required to make the system work are as follows:

- **2 LAN Cards:**

One LAN card is used to connect the gateway or the proxy server to the internet and the second is required to connect the clients that are on LAN to the proxy server. Without 2 LAN cards system will not be able to work.

- **Working Internet connection:**

There should be a working internet connection needed, to which the proxy server can connect. Only then the proxy server will be able to fulfill the client's request.

- **Proxy settings:**

The client's IP address should be in accordance with the proxy server so that the client system will be able to connect to the proxy server.

The IP address and the port number will be set on the client side and these settings can be applied using tools options in the internet explorer. And the proxy server will have different settings to connect with the internet.

Microsoft Internet Explorer can be configured to use a proxy server to connect to the Internet. Internet Explorer 6.0

- On the Tools menu in Internet Explorer, click Internet Options, click the Connections tab, and then click LAN Settings.
- Under Proxy server, click to select the Use a proxy server for your LAN check box.
- In the Address box, type the IP address of the proxy server.
- In the Port box, type the port number that is used by the proxy server for client connections (by default, 8080).
- You can click to select the Bypass proxy server for local addresses check box if you do not want the proxy server computer to be used when you connect to a computer on the local network (this may speed up performance).
- Click OK to close the LAN Settings dialog box.
- Click OK again to close the Internet Options dialog box.

4.1.7 OUTPUTS

The outputs of the system depend on the input.

- The file is downloaded at client's side if the client is requested to download a file.
- The website will be opened at client's end if the client is requested to open a website.
- Output can also be an error message if the client will try to request for that sort of site or file, that are banned by the system or for which the user's don't have permissions to access or download particular type of file.

4.1.8 DATABASE DESIGN

The database consists of following tables:-

Table1: contains the usernames, passwords and usage of each client:

Username	password	Type_of_user	Allocated Bandwidth	Used Bandwidth
User182	4004	faculty/student	1GB	25MB

Table2: consists of one column and contains the blacklisted websites:

Blacklisted websites
www.songs.com
www.youtube.com
www.facebook.com

Table3: consists of one column that includes the extensions that are not allowed to be downloaded:

Blocked extensions
.exe
.mp3

Table4: consists of one column that includes the ip-addresses that are allowed to access the web:

White listed-ip
168.12.1.1
168.123.1.1

Table5: consists of column that includes blocked keywords.

Blocked Keywords
Under 18
abc

Table 6: click jacked websites

Click jacked websites
www.songs.pk.com
www.abc.com

Table 7: click time to live

IP	REQUESTED CLICK	TIME TO LIVE FOR CLICK	TIME OF CLICK	STATUS
172.31.1.89	WWW.ABC.COM	*****	****	ALIVE/DEAD

Figure 5.1: System Model

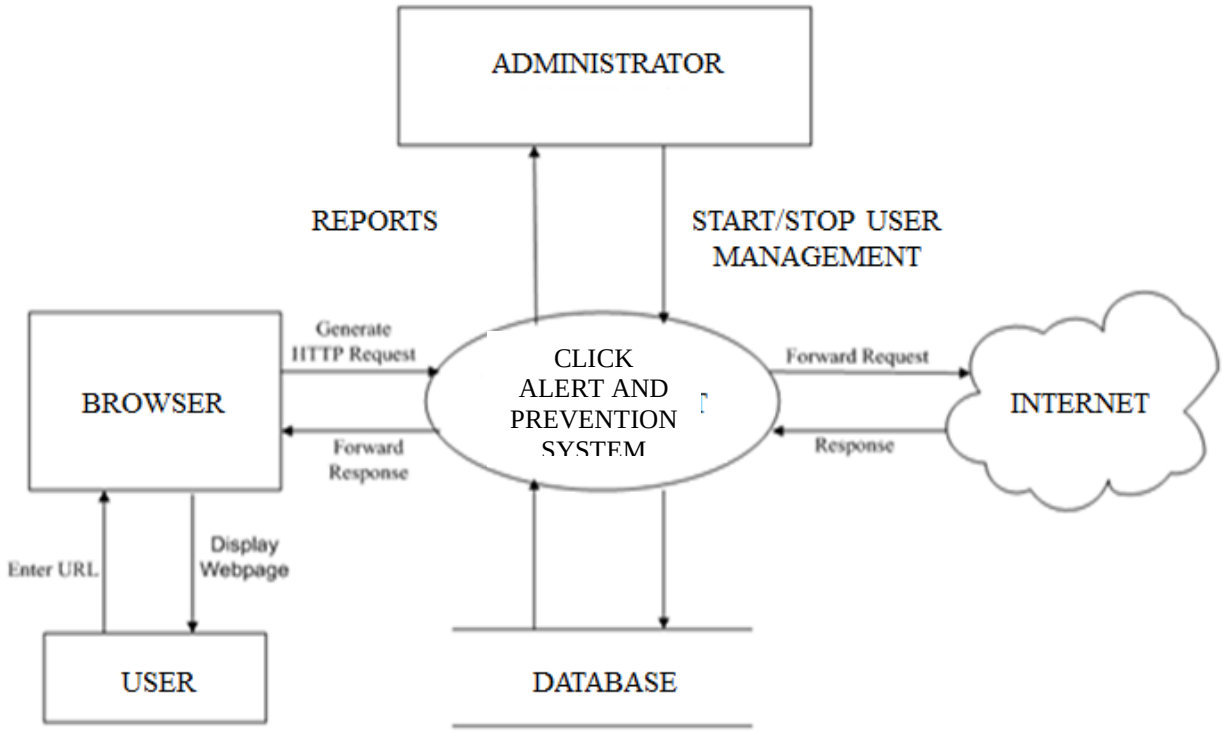


Figure 5.2: Extension Check and Website Blocking

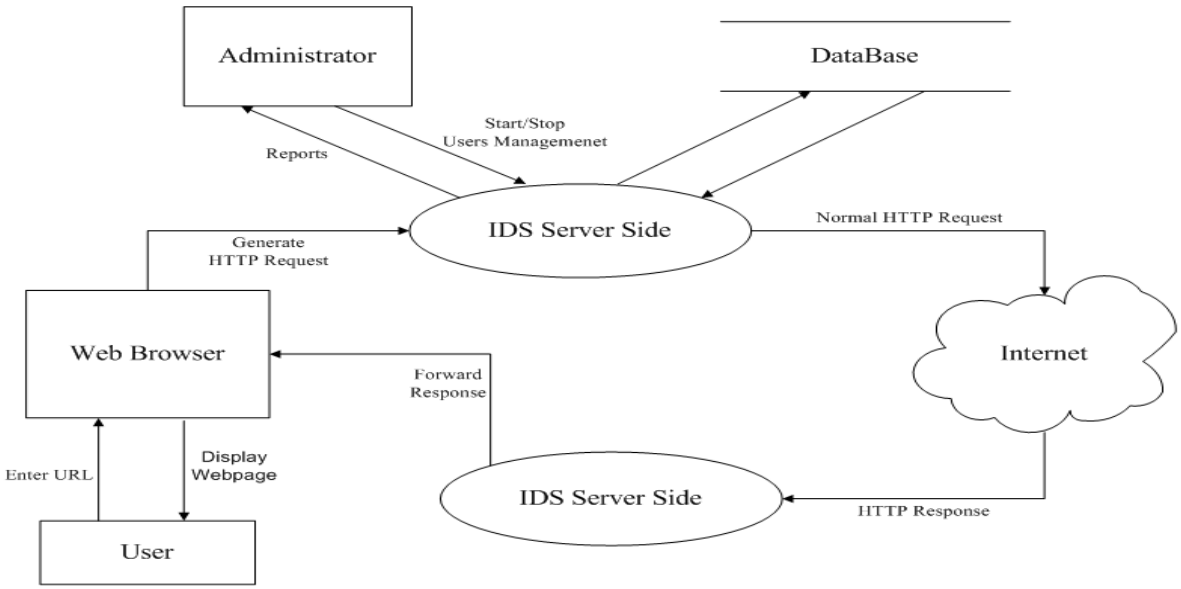


Figure 5.3: Allowing access to White Listed User

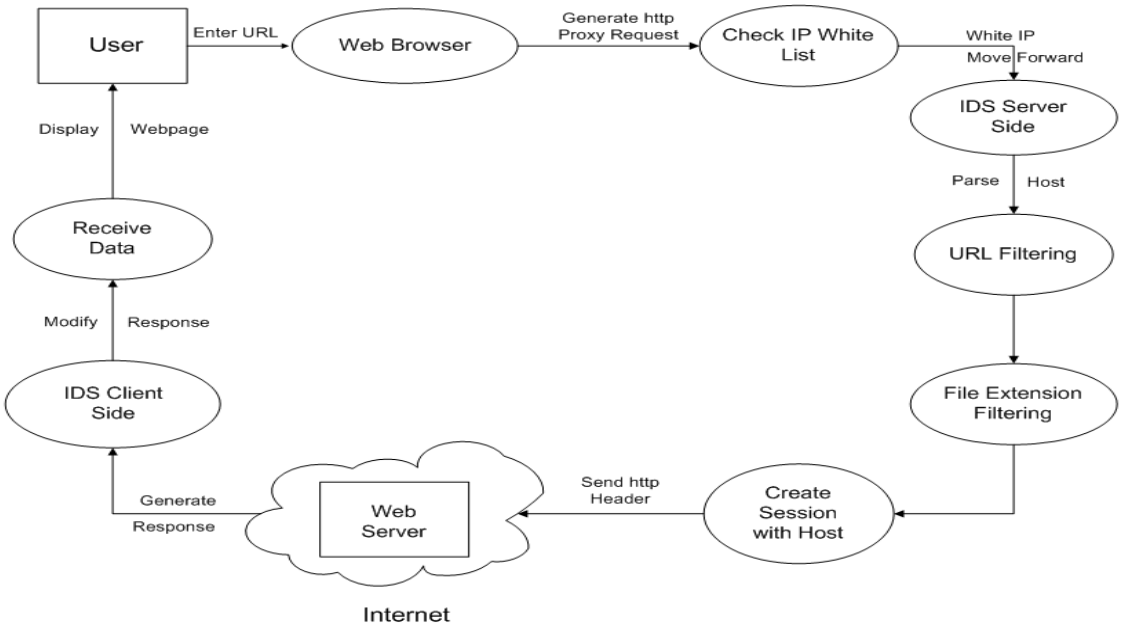
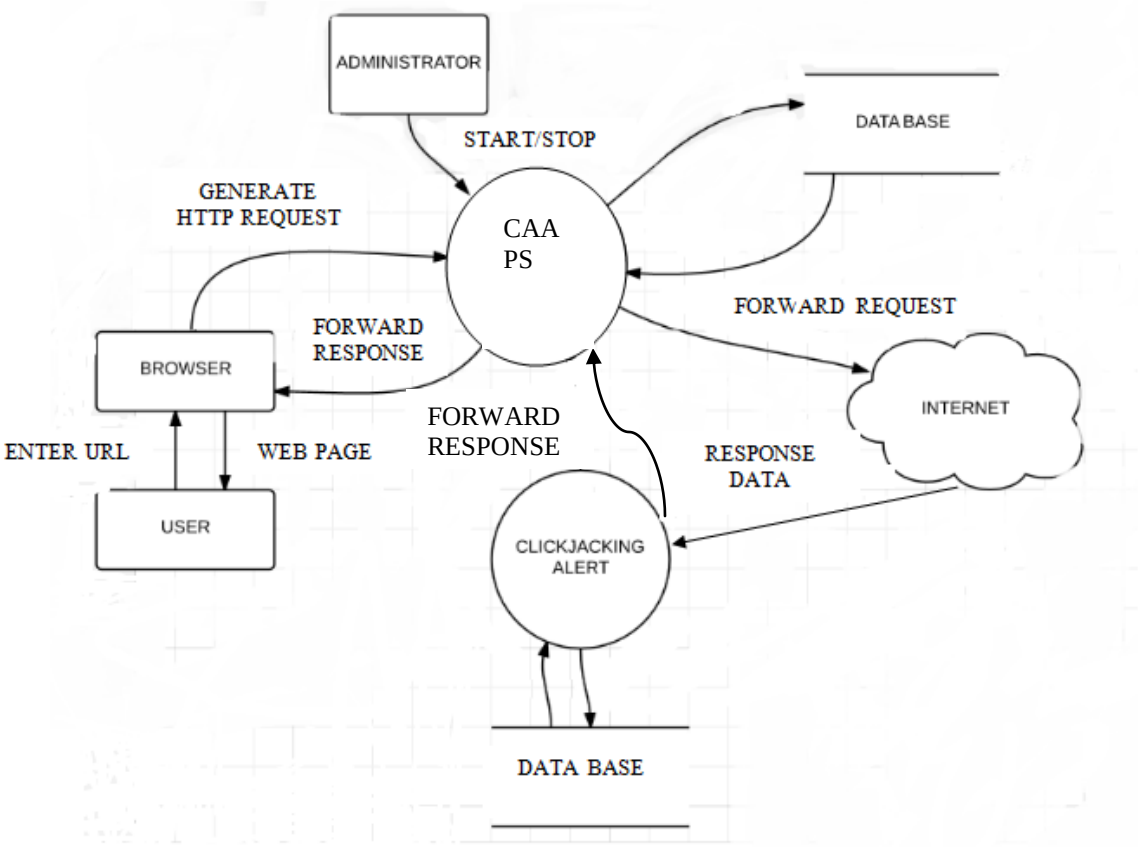


Figure 5.4 clickjacking alert and prevention system



5.2 FLOWCHARTS

Figure5.5. Flow Chart Client Side (Web Browser)

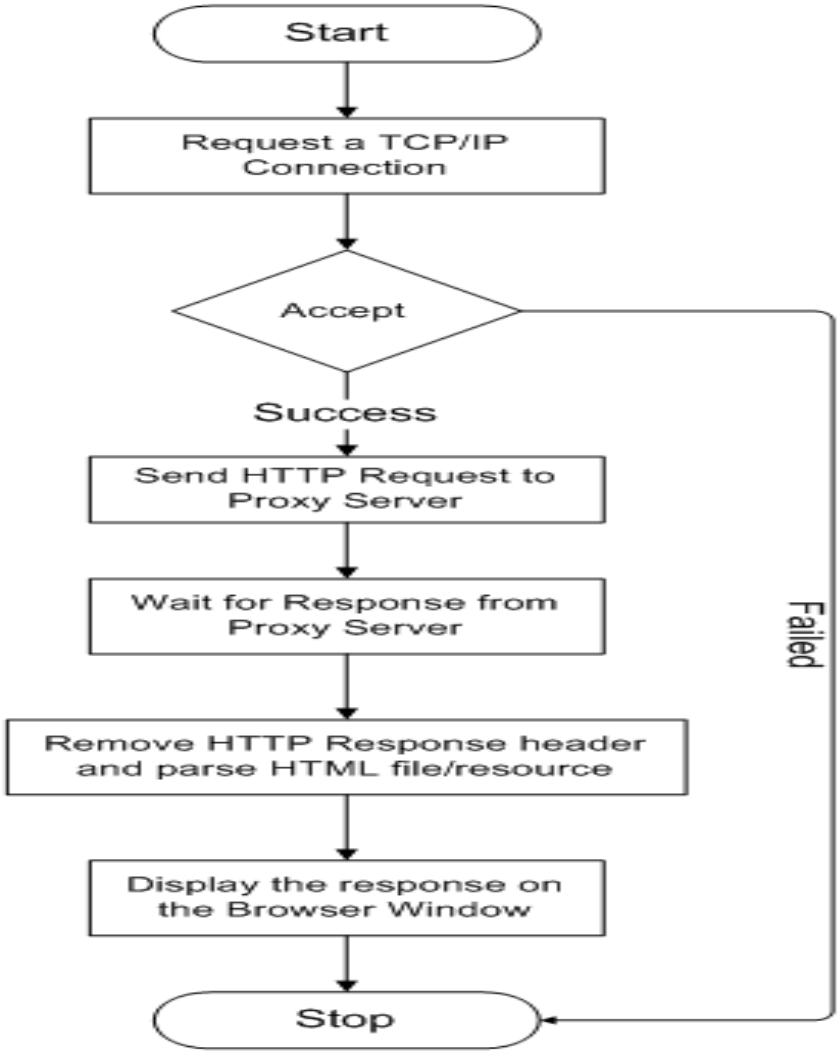


Figure 5.6. Flow Chart Server Side (clickjacking alert and prevention system):

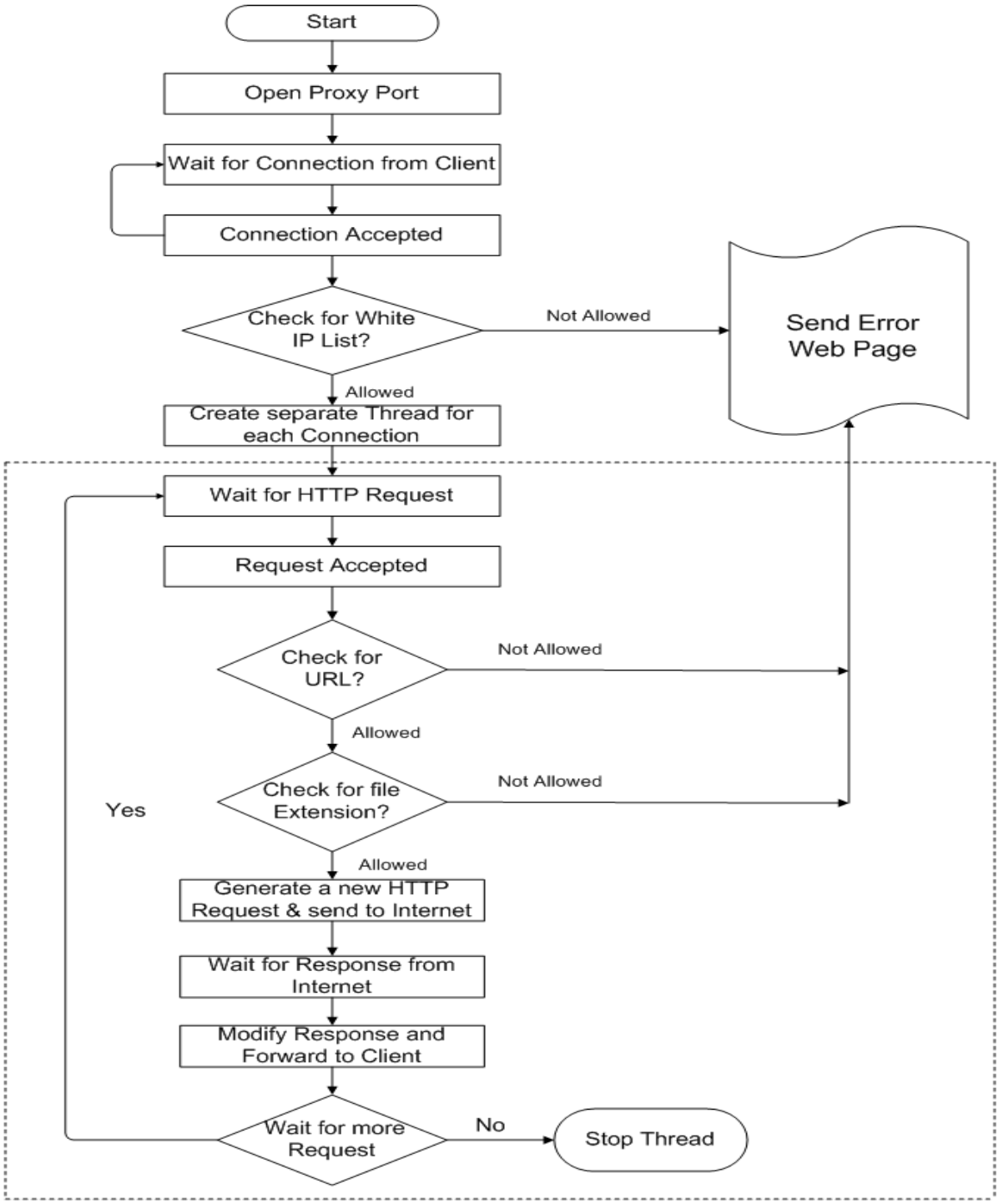


Figure 5.7 Flow Chart for URL Filtering

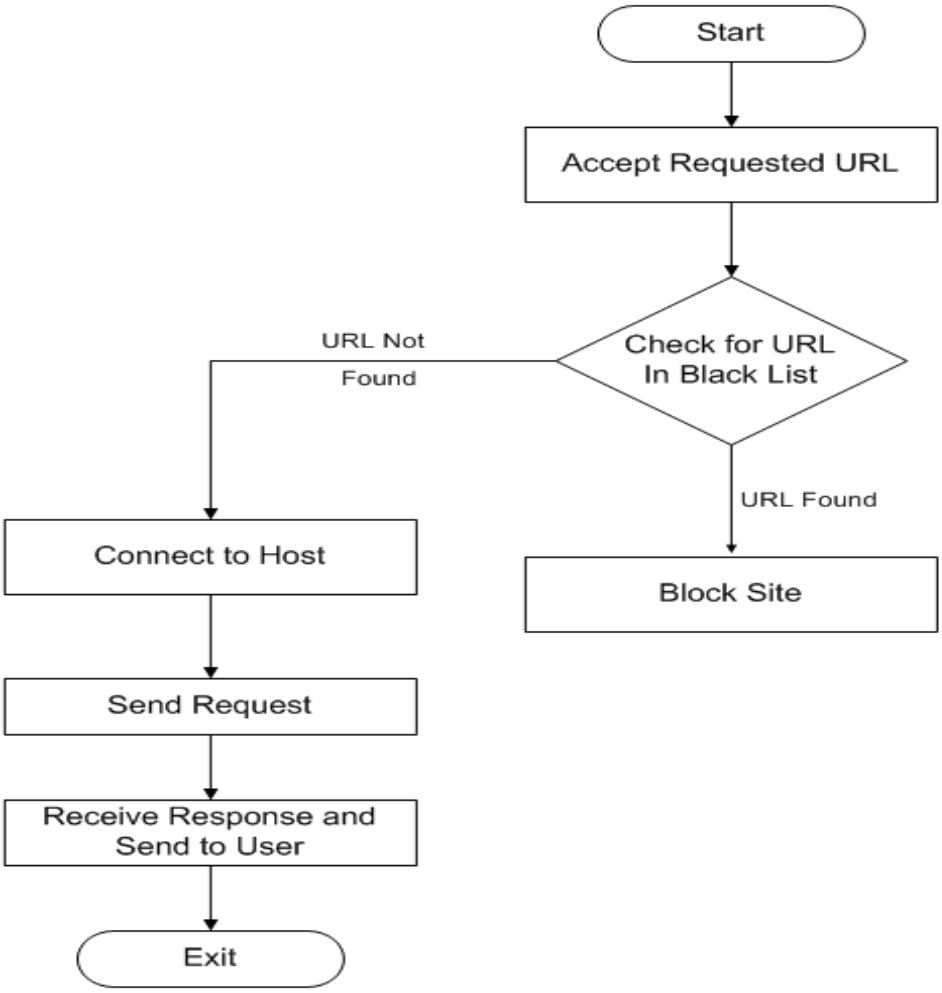


Figure5.8 Flow Chart for IP Checking

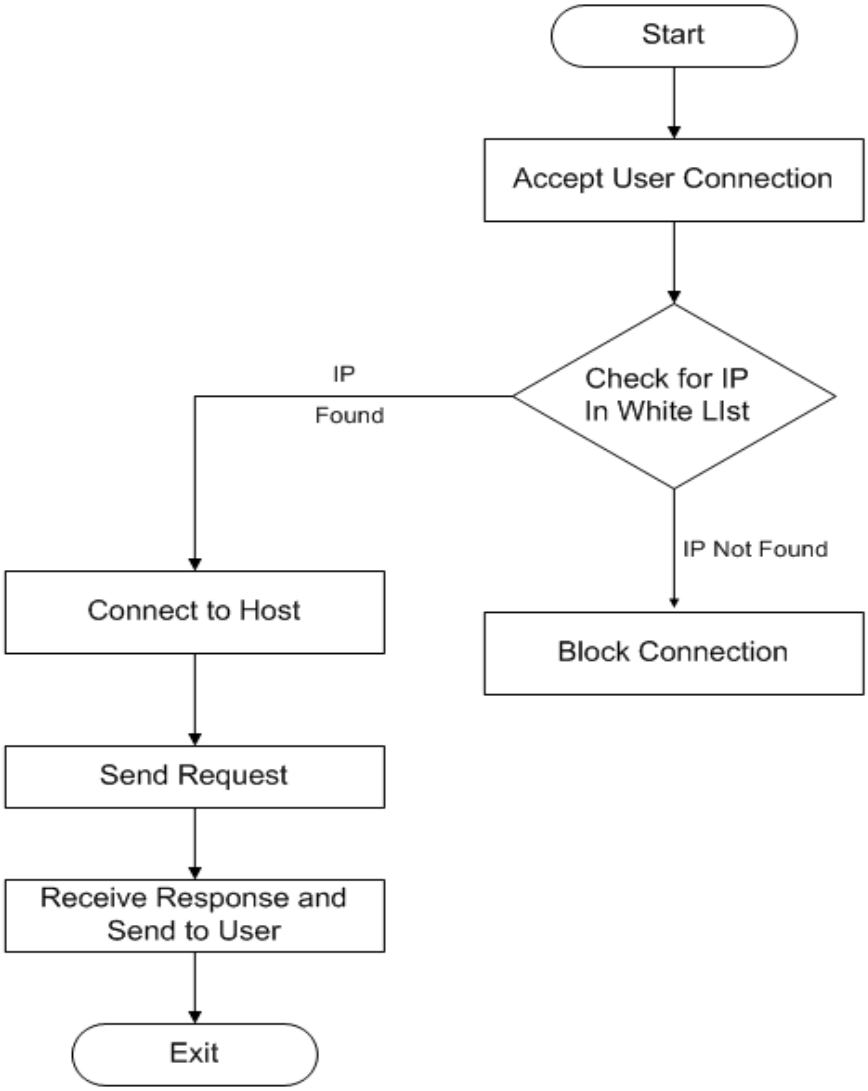


Figure5.9 Flow Chart Checking for File Extension

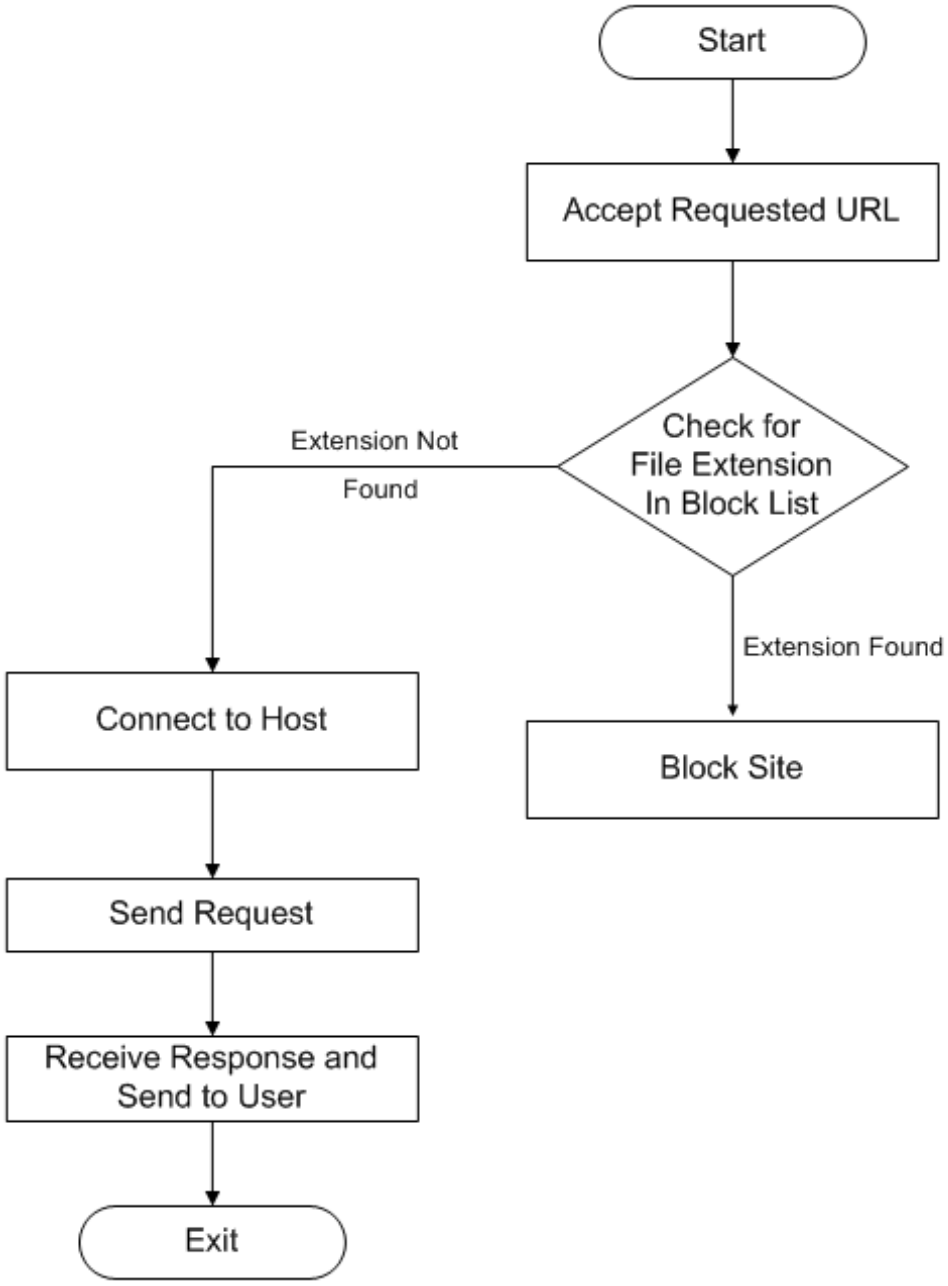


Figure5.10 Flow chart for keyword check

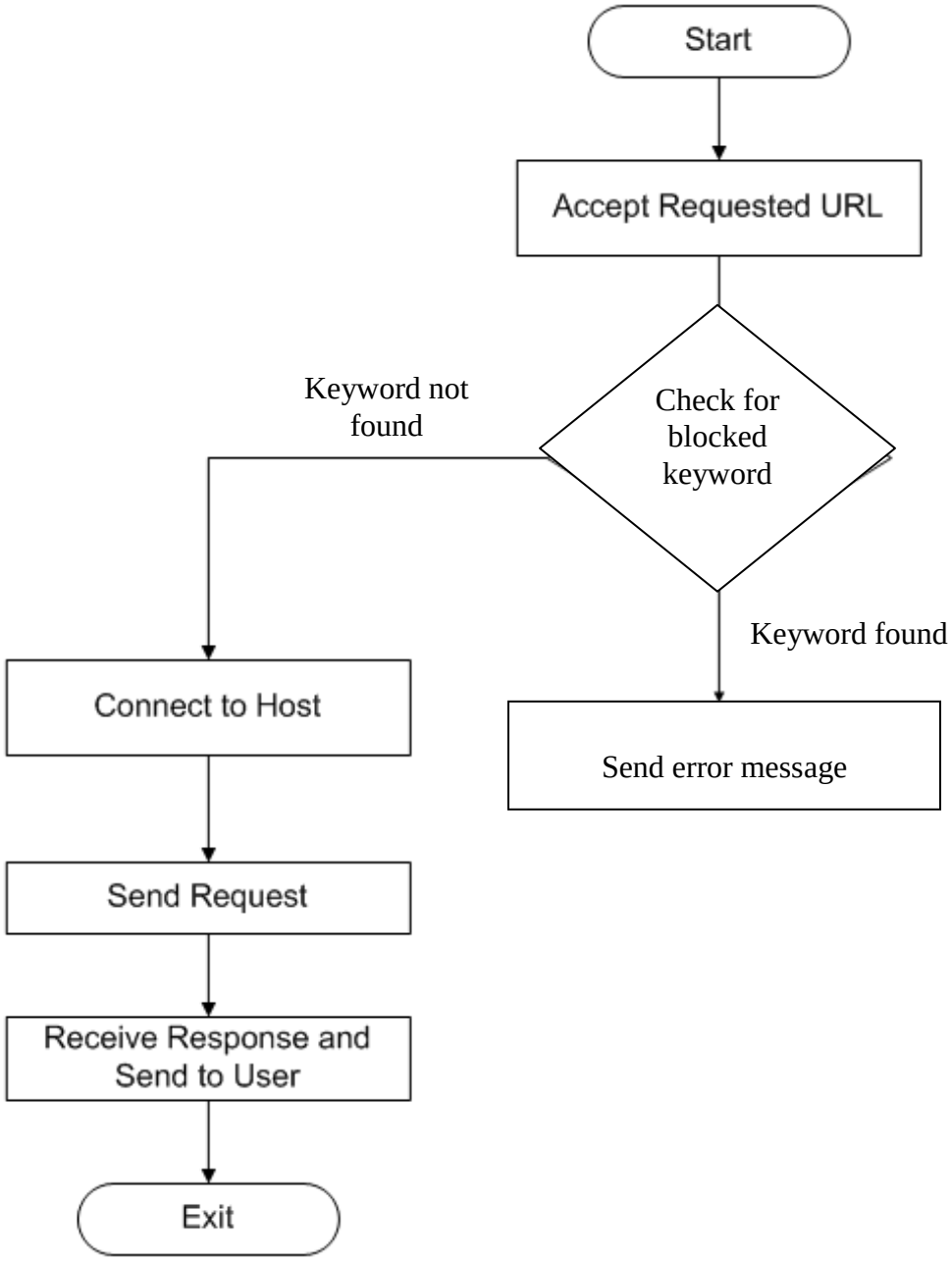


Figure 5.11 Server side keyword blocking

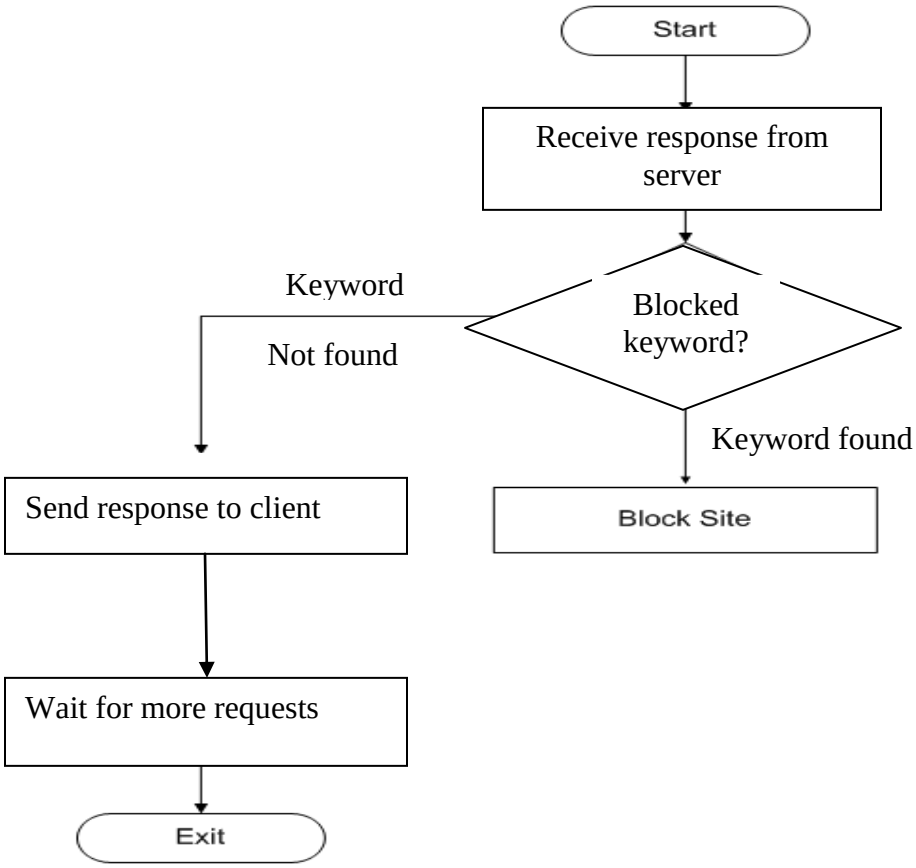
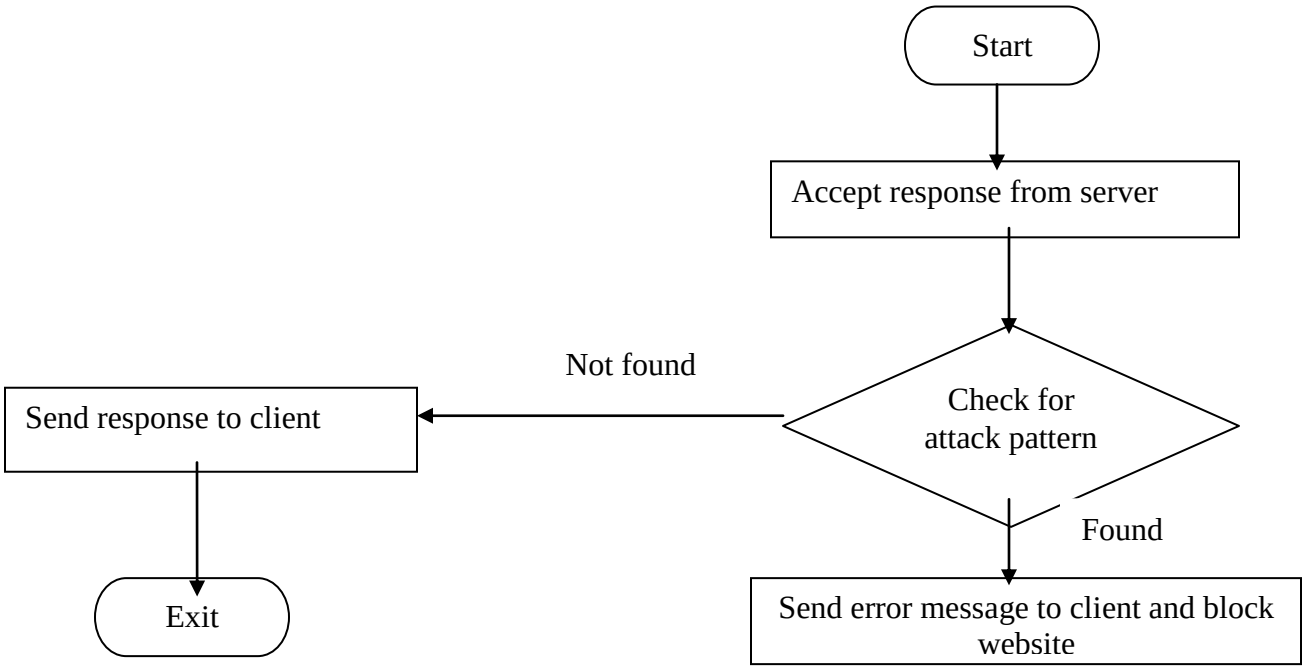


Figure 5.12 Clickjacking alert systems



CHAPTER 6: SCREENSHOTS

Figure 6.1 execute the jar file from command prompt

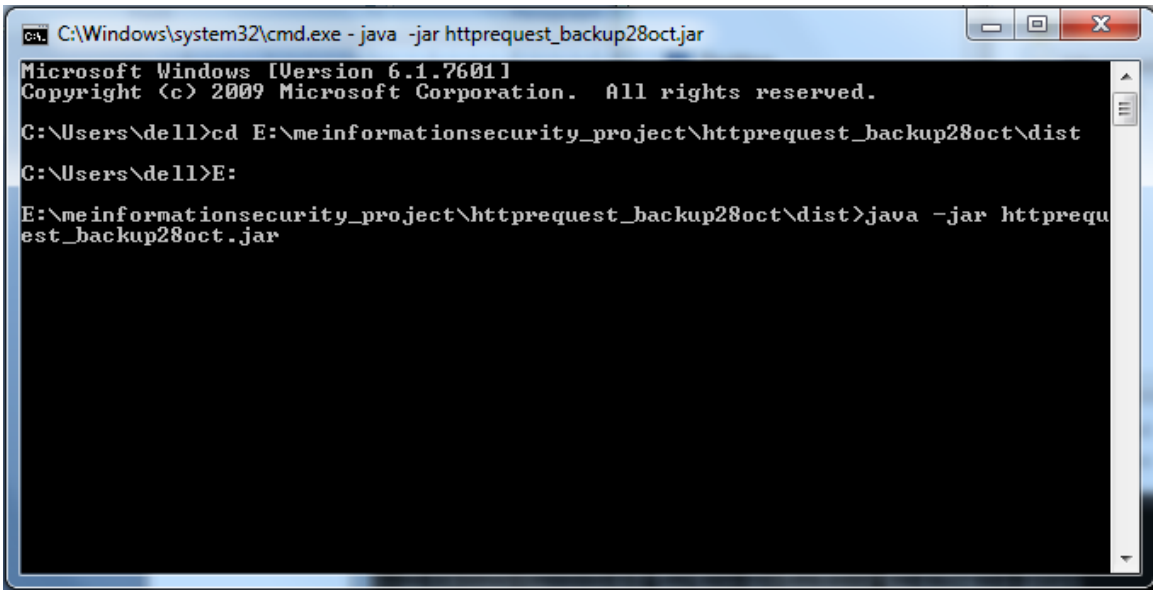


Figure 6.2 Project started

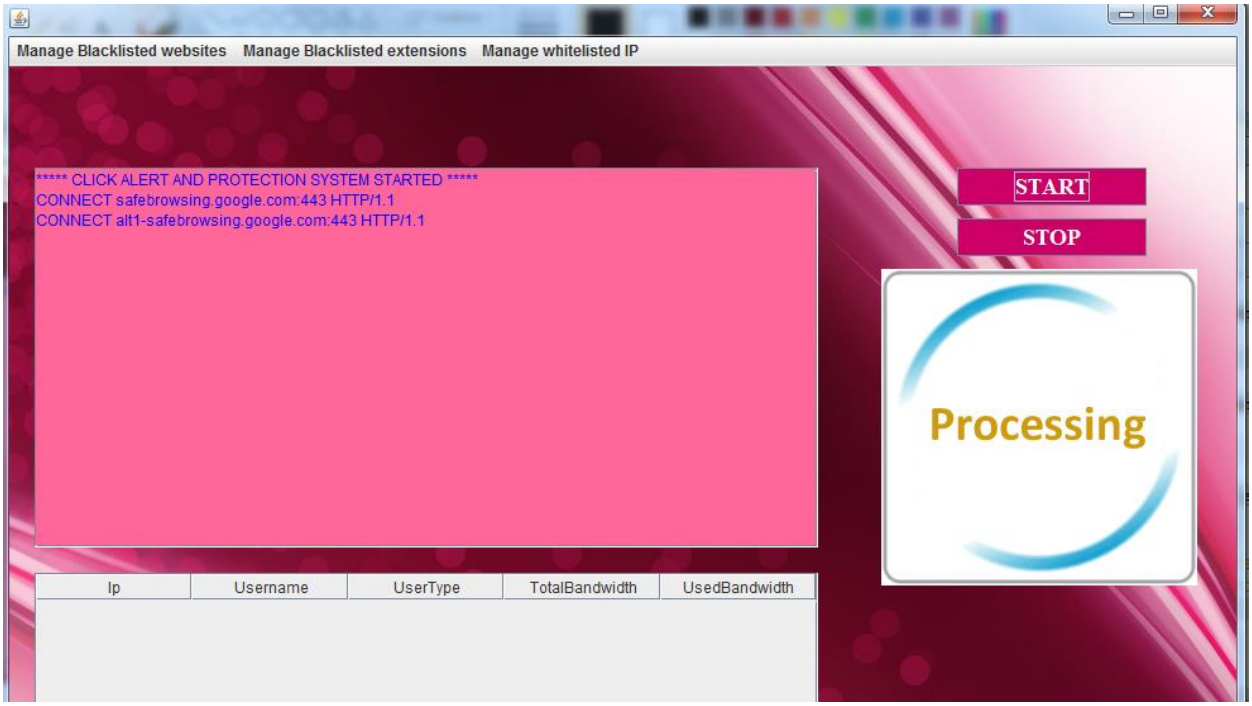


Figure 6.3 Manage user

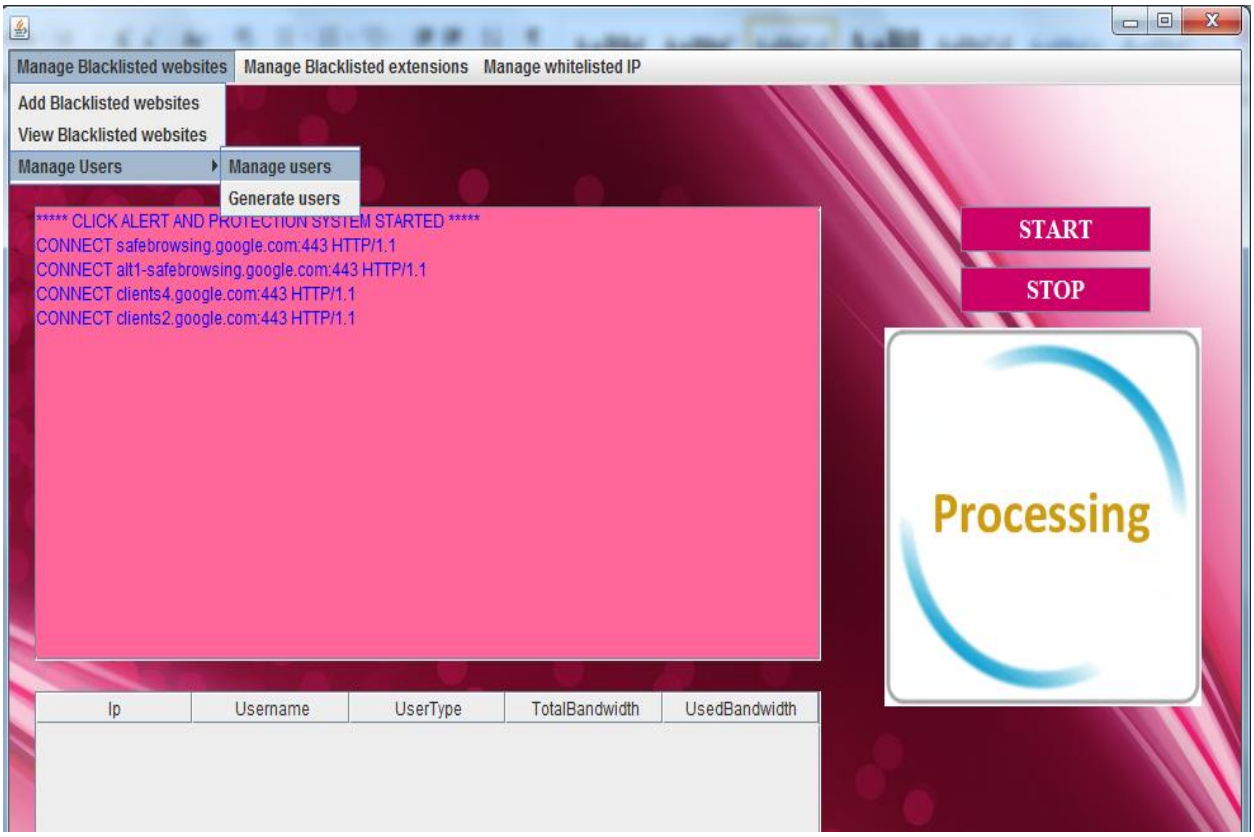


Figure 6.4 Manage login

Figure 6.5 Generate Bulk Users

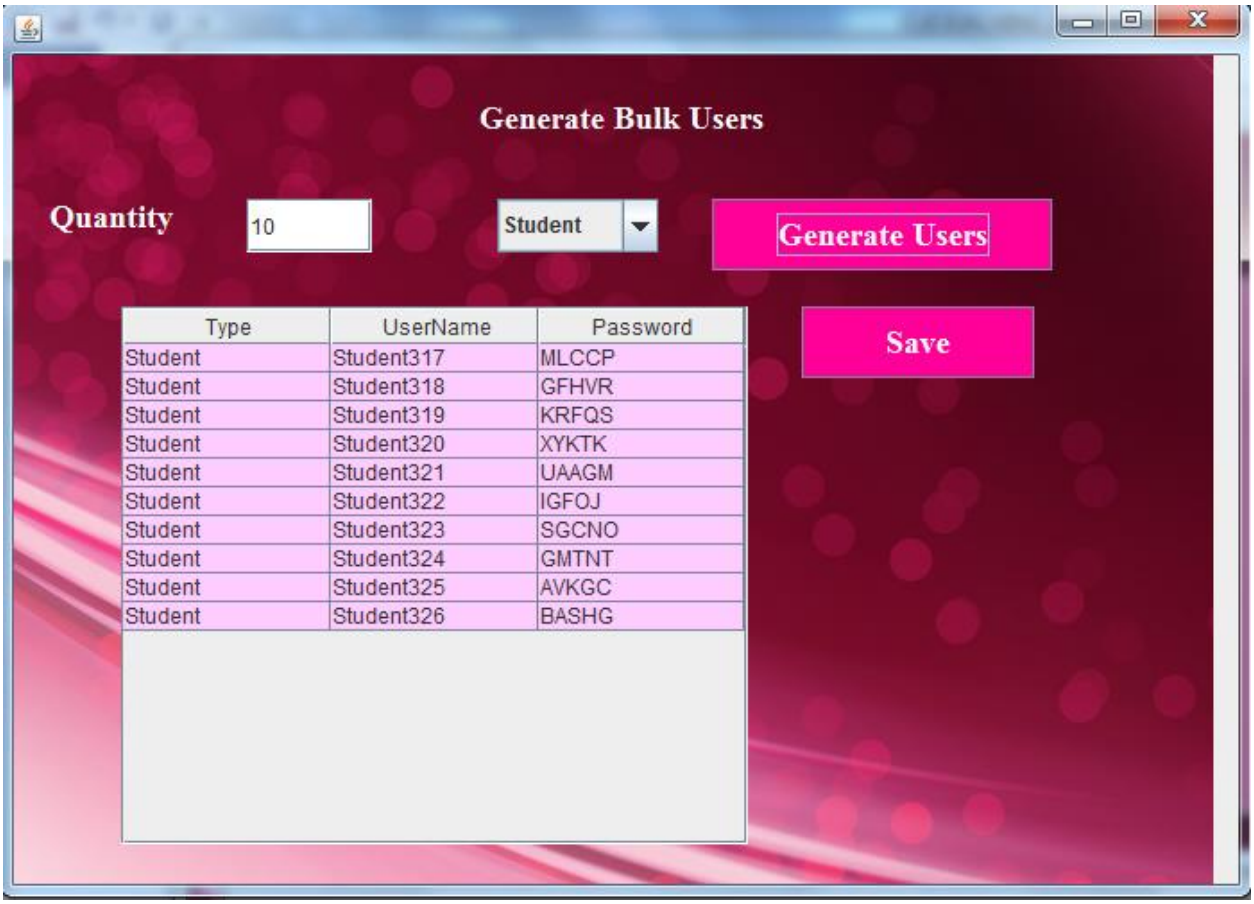


Figure 6.6 Manage Blacklisted Extensions

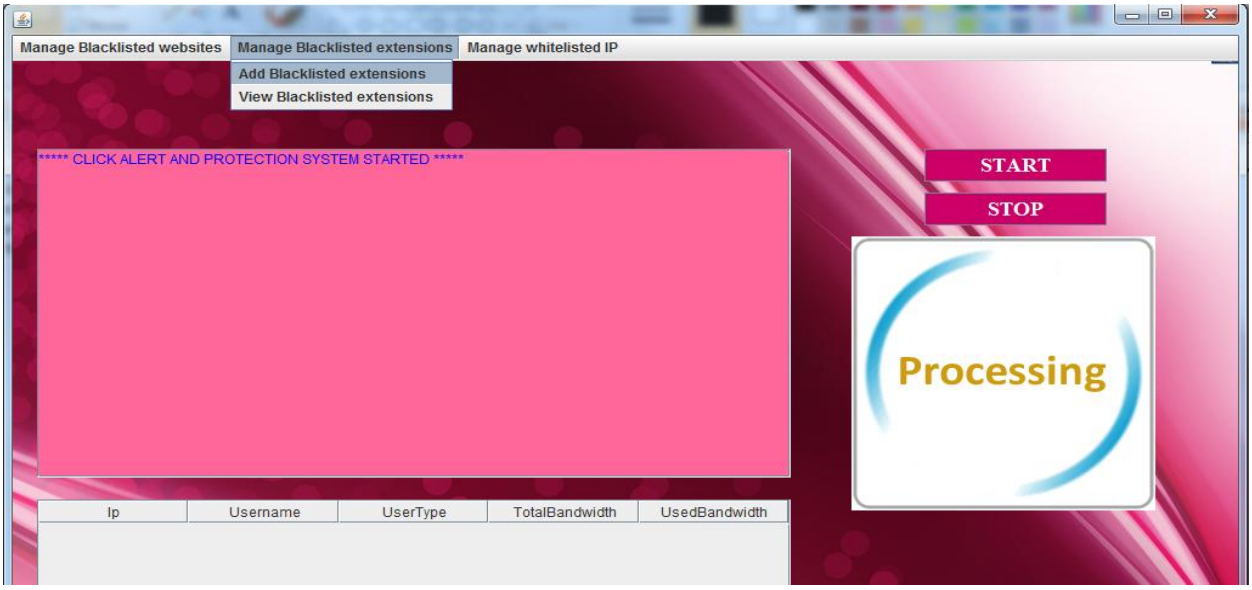


Figure 6.7 Manage white listed IP

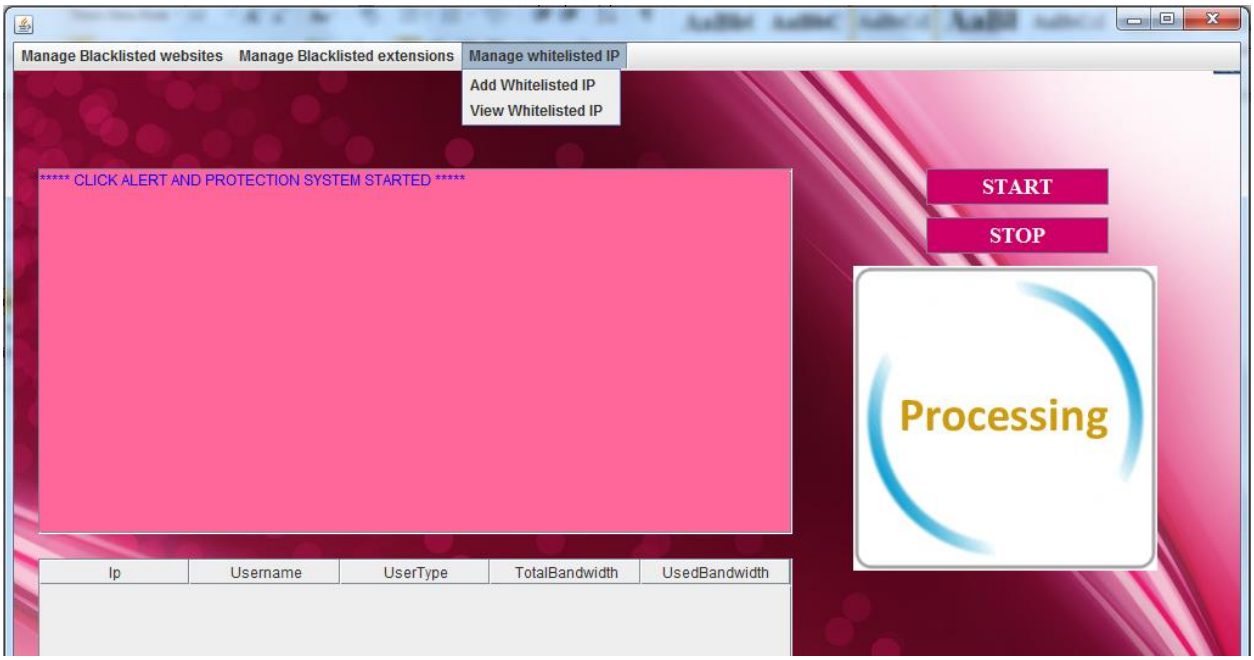


Figure 6.8 Add Blacklisted extensions

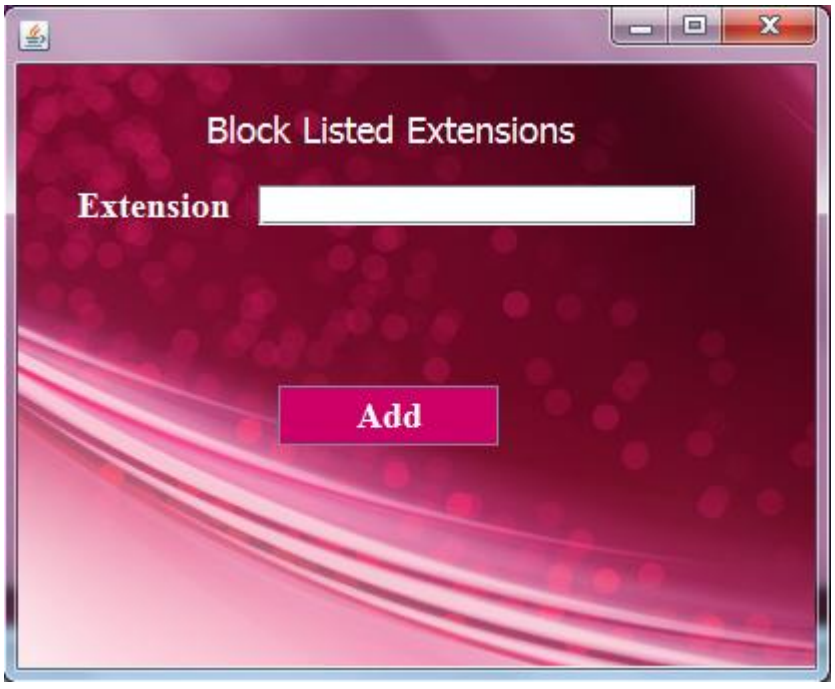


Figure 6.9 View Blacklisted Extensions

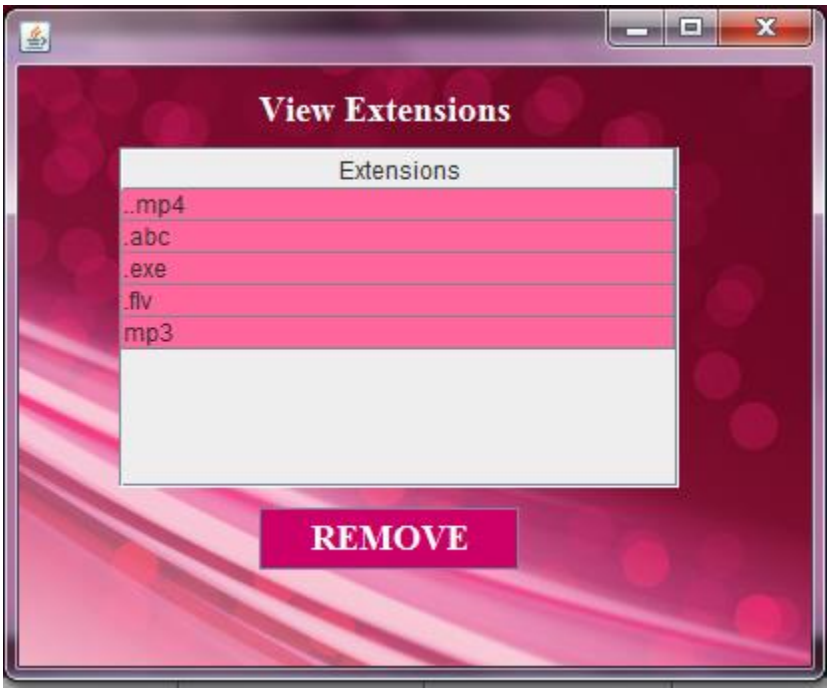


Figure 6.10 White listed IP

Figure 6.11 View White listed IP

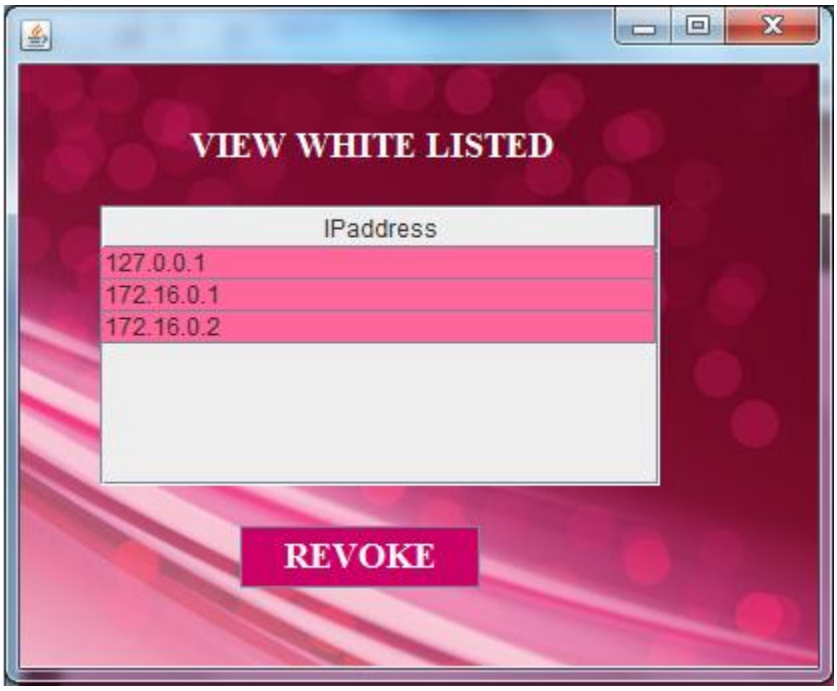


Figure 6.12 CLICKTTL

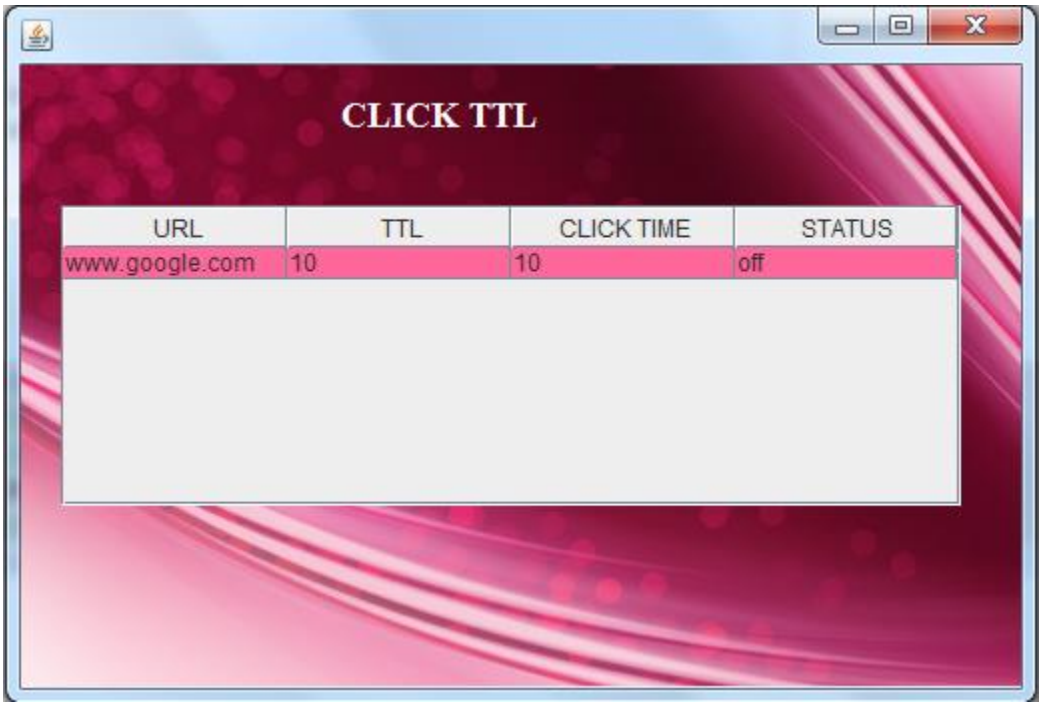


Figure 6.13 407 Authentication required

Figure 6.14 clickjacking alert and prevention system window

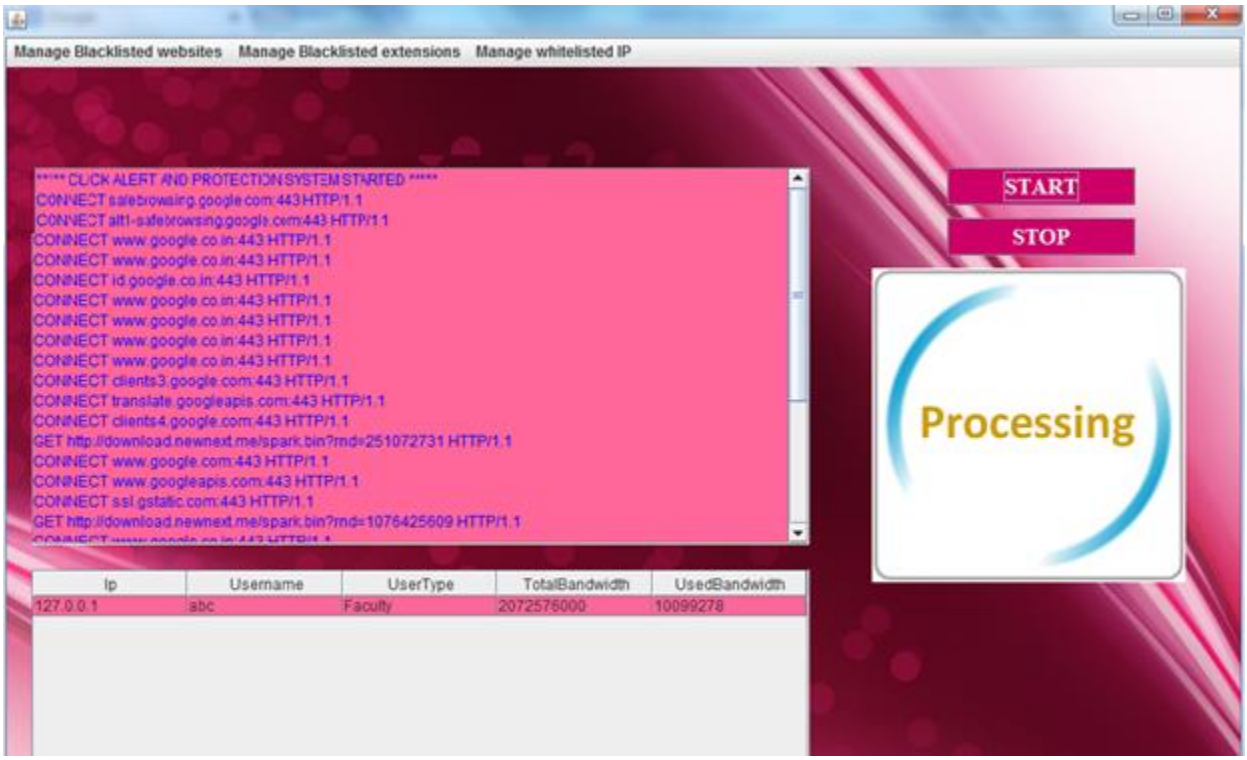


Figure 6.15 Login successful

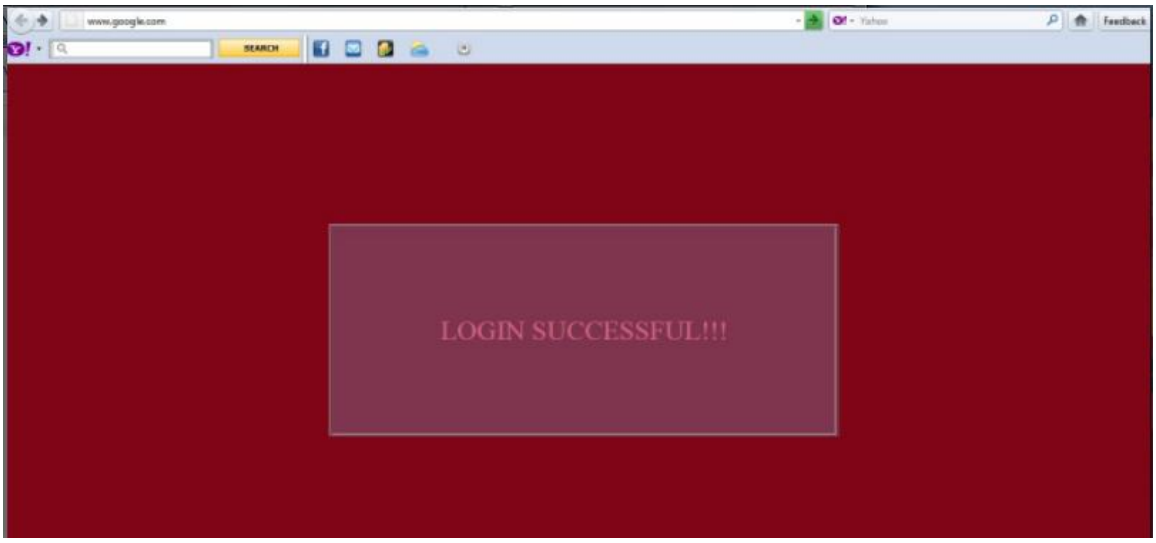


Figure 6.16 Authentication required in case of unsuccessful login

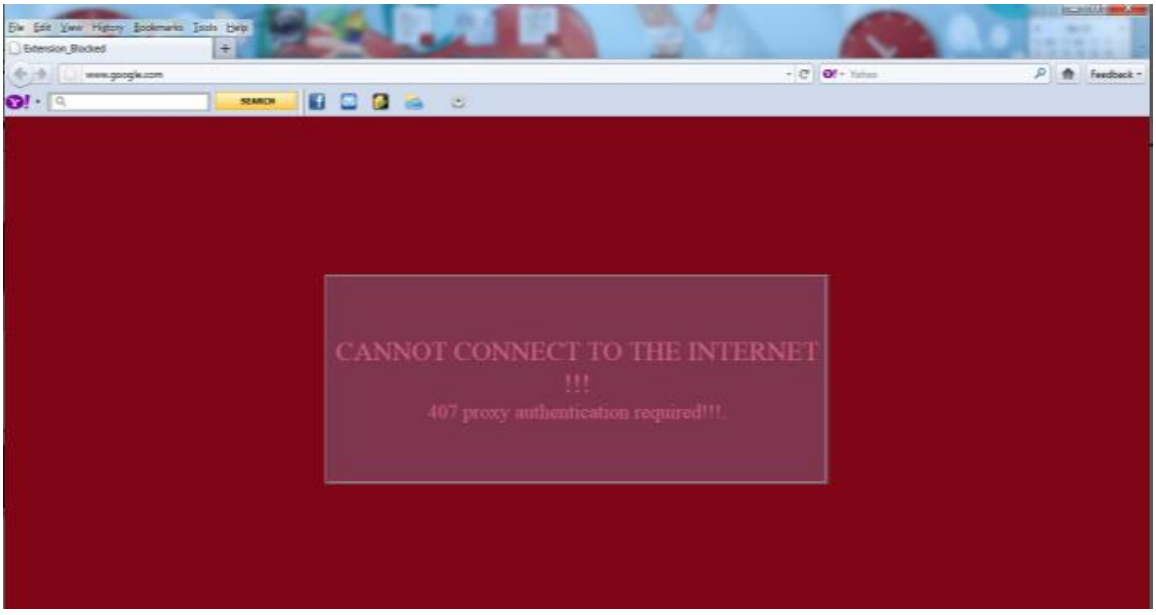


Figure 6.17 Google web page

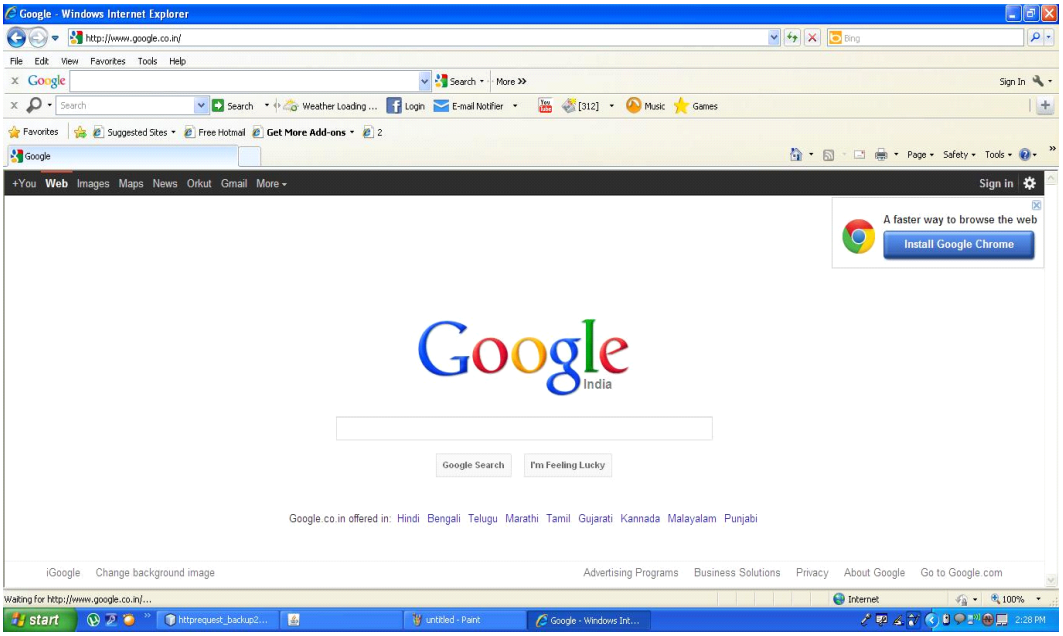


Figure 6.18 Adding Blacklisted websites

Figure 6.19 Viewing blacklisted websites

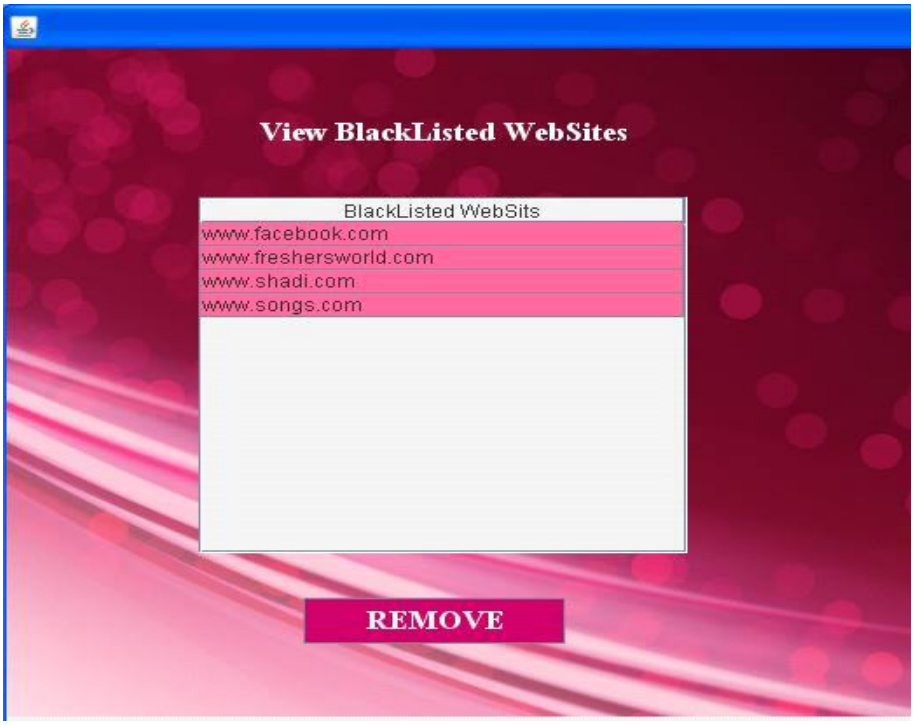


Figure 6.20 Allocating bandwidth

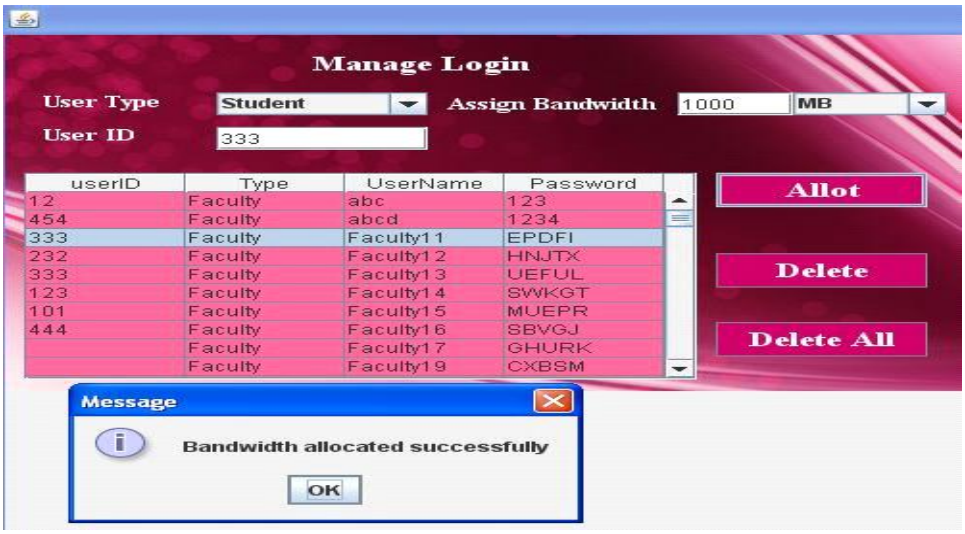


Figure 6.21 viewing bandwidth used by users

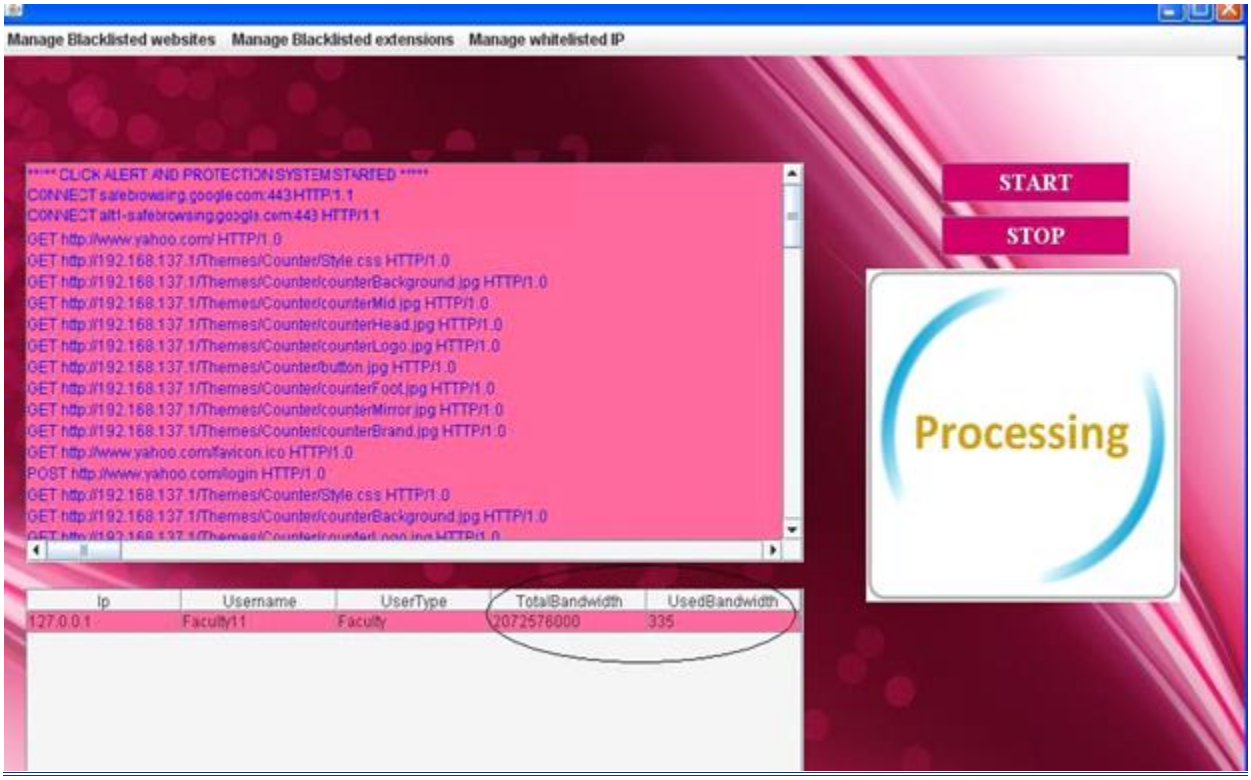


Figure 6.22 Adding blocked extensions

Figure 6.23 Viewing blocked extensions

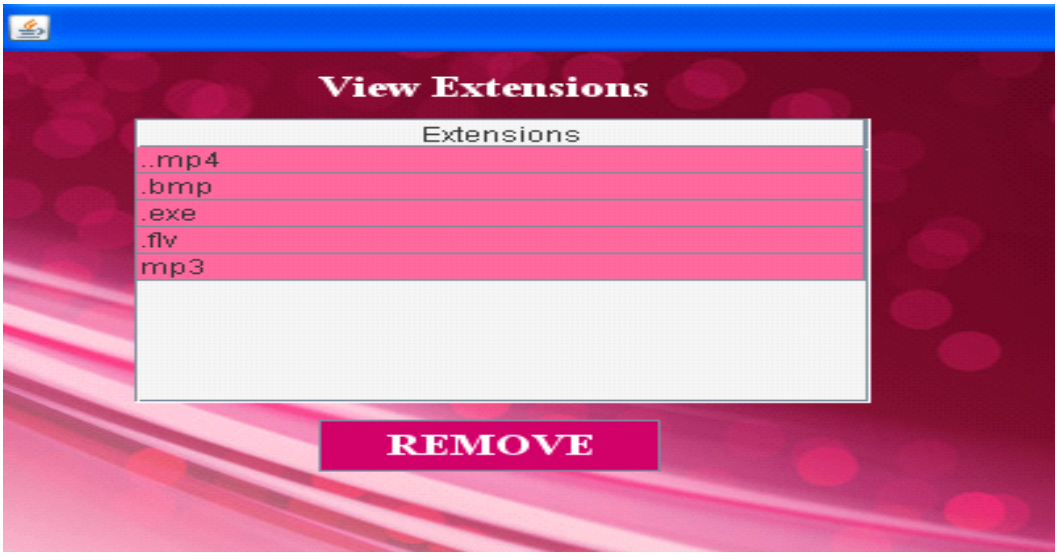


Figure 6.24 HTTP runs successfully

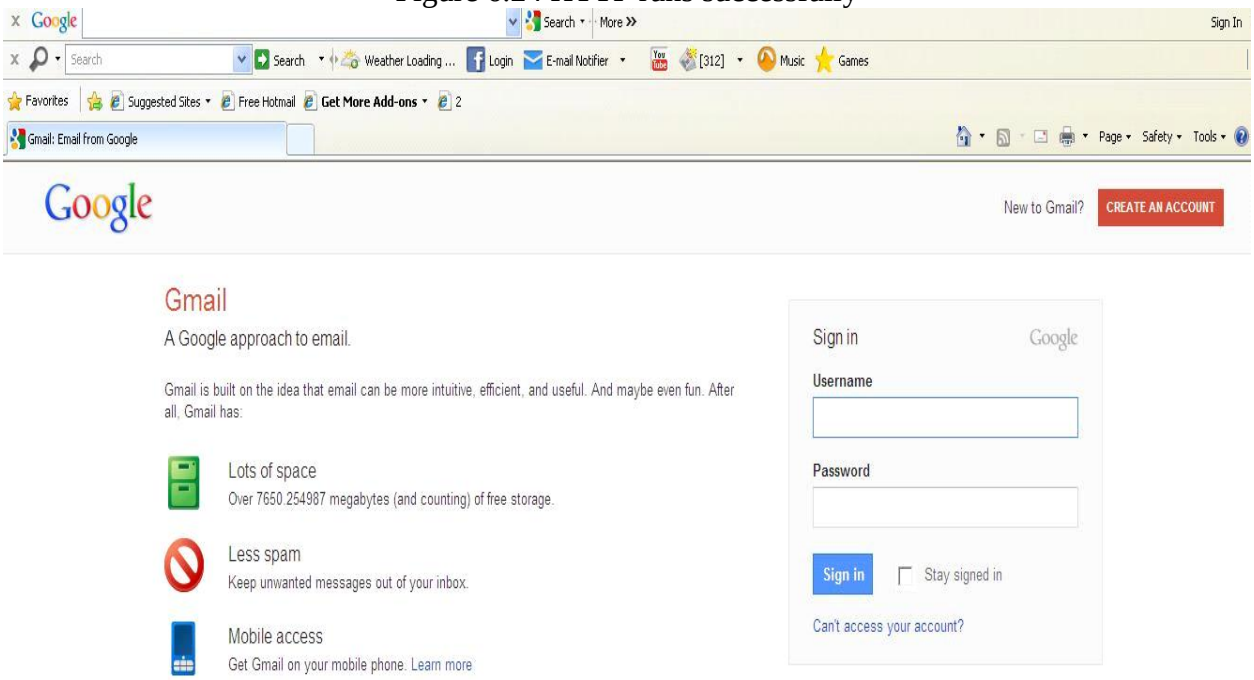


Figure 6.25 web page for blocked websites, blocked keywords and click jacked websites

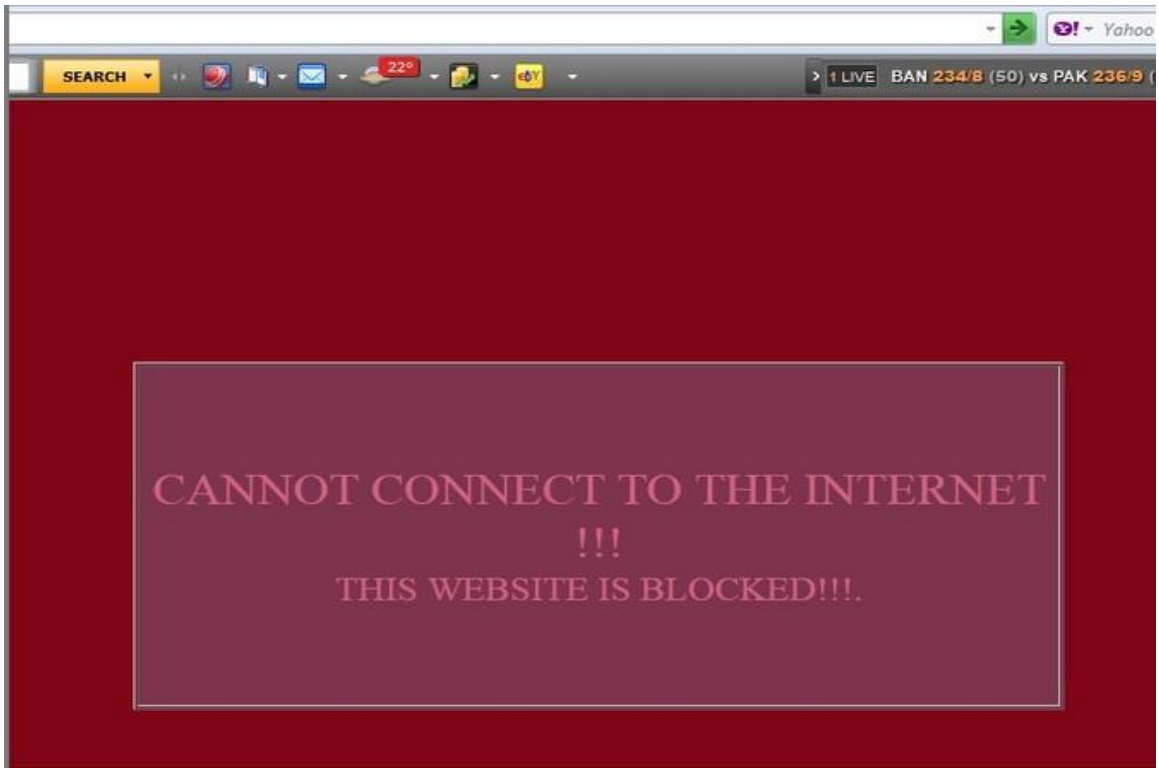


Figure 6.26 GET request working properly

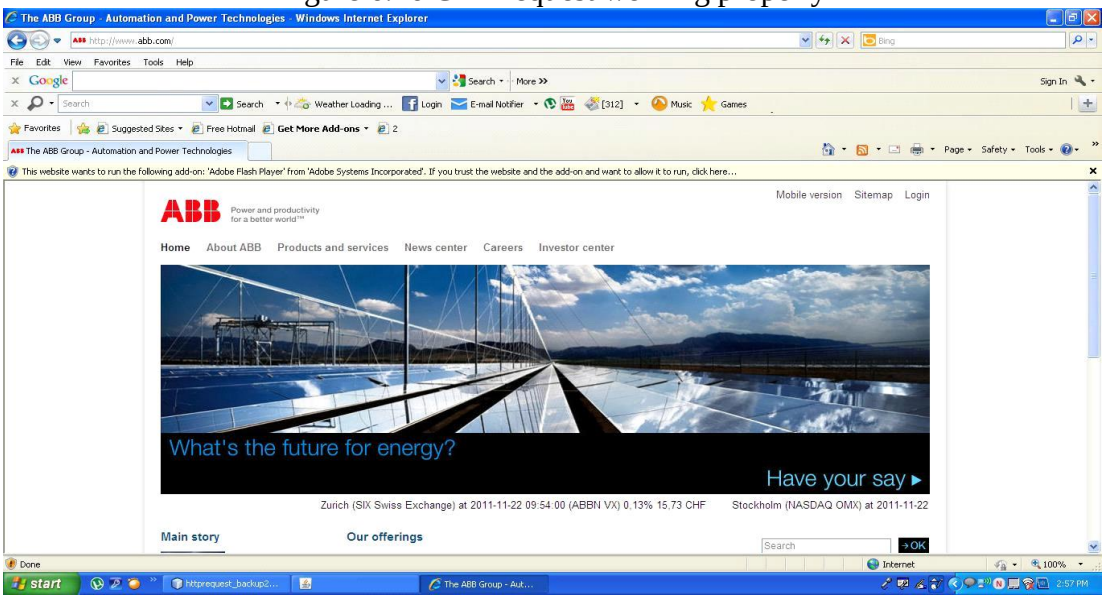


Figure 6.27 POST request working properly

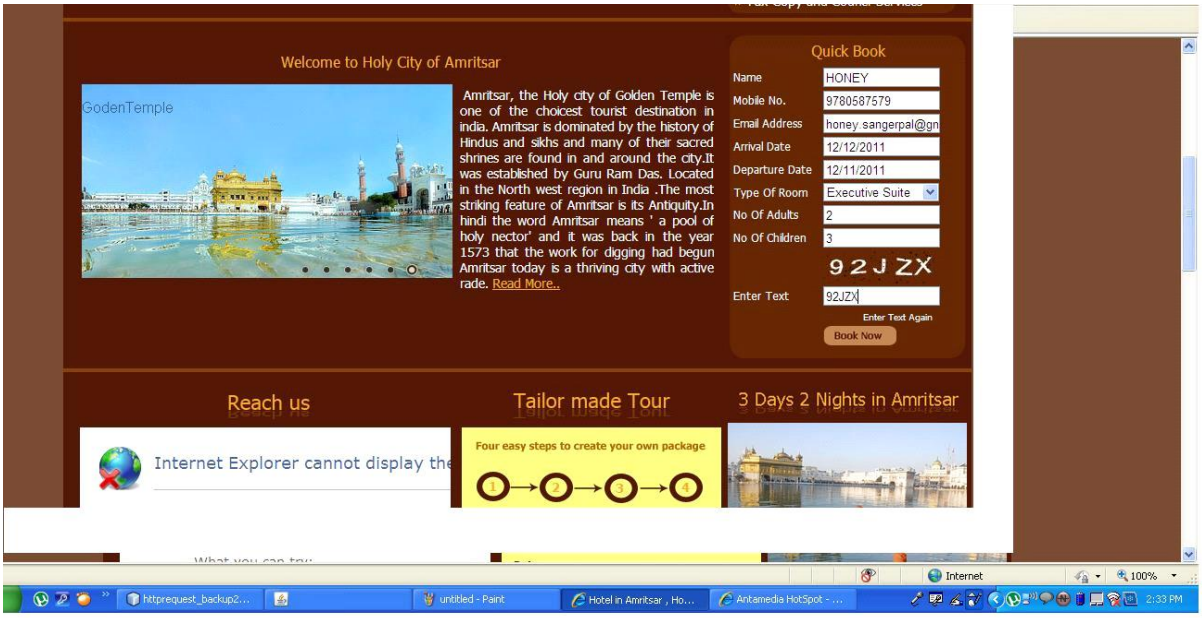


Figure 3.28 CONNECT request working properly

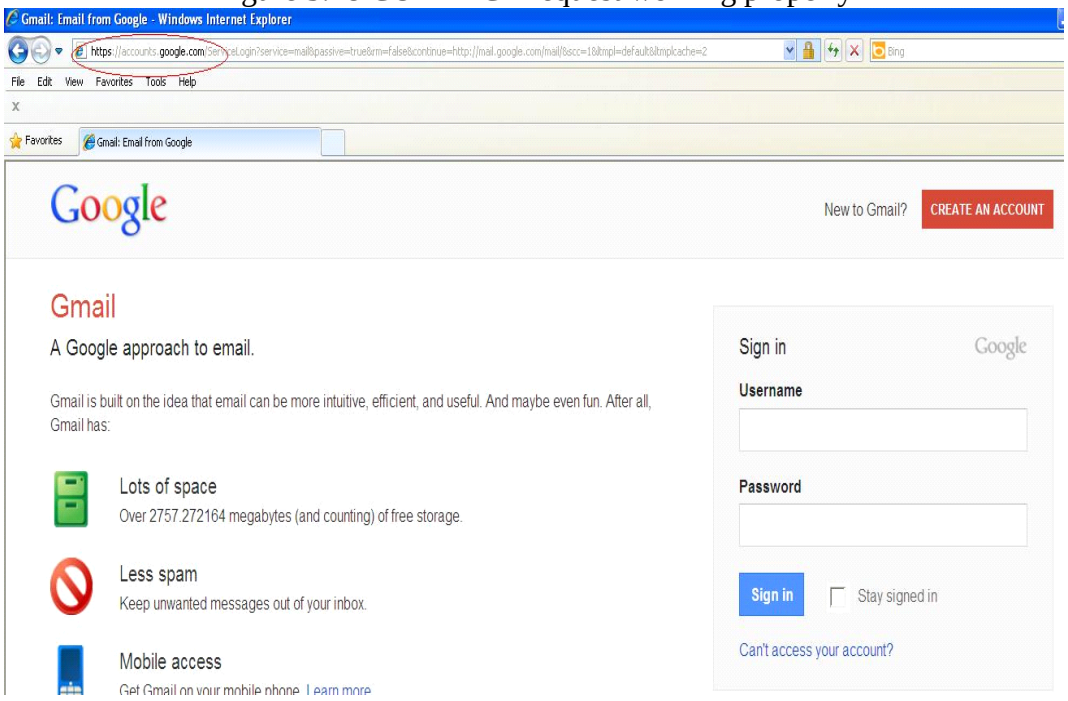
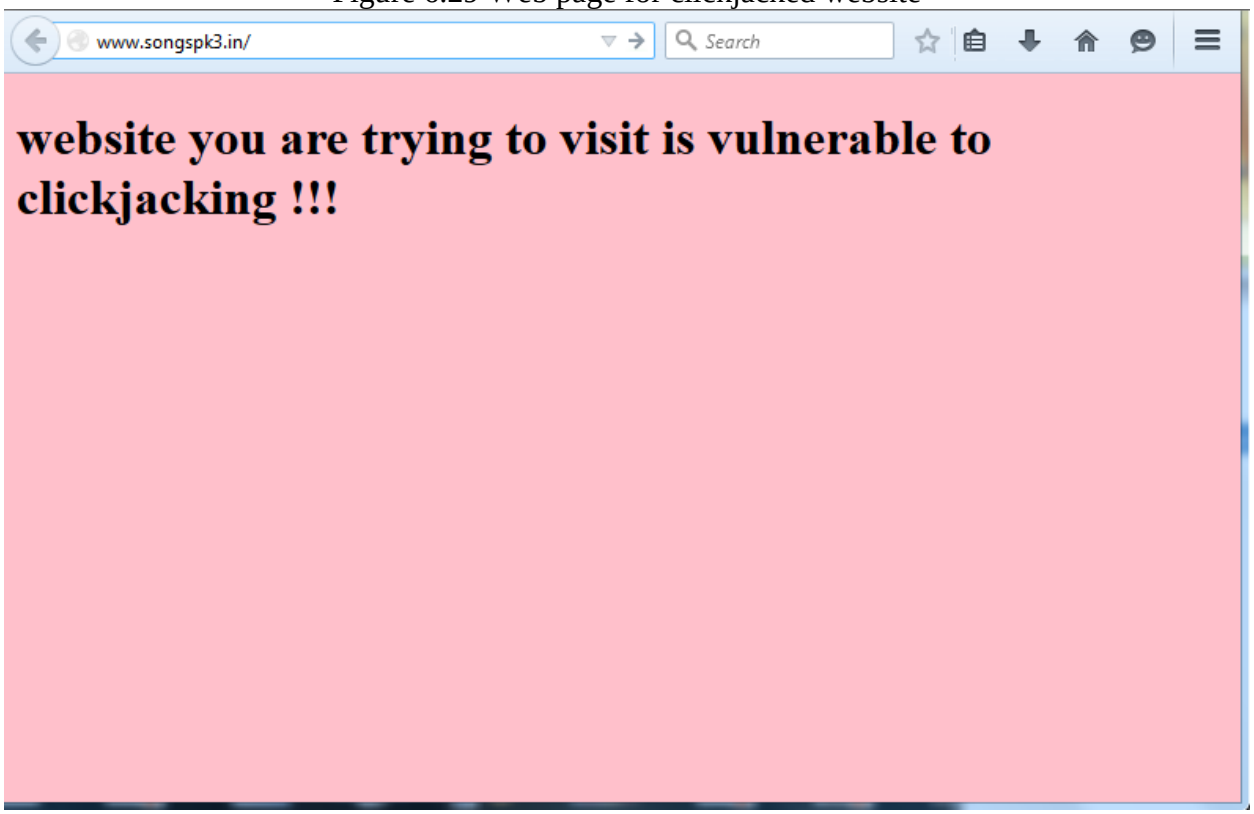


Figure 6.29 Web page for clickjacked website



CHAPTER 7: TESTING

This section describes the validity and evaluation of the implementation. The client browser surfs various websites and the CAAPS system tries to find various clickjacking vulnerabilities in the web content and fixes it in real time.

7.1 clickjacking web page

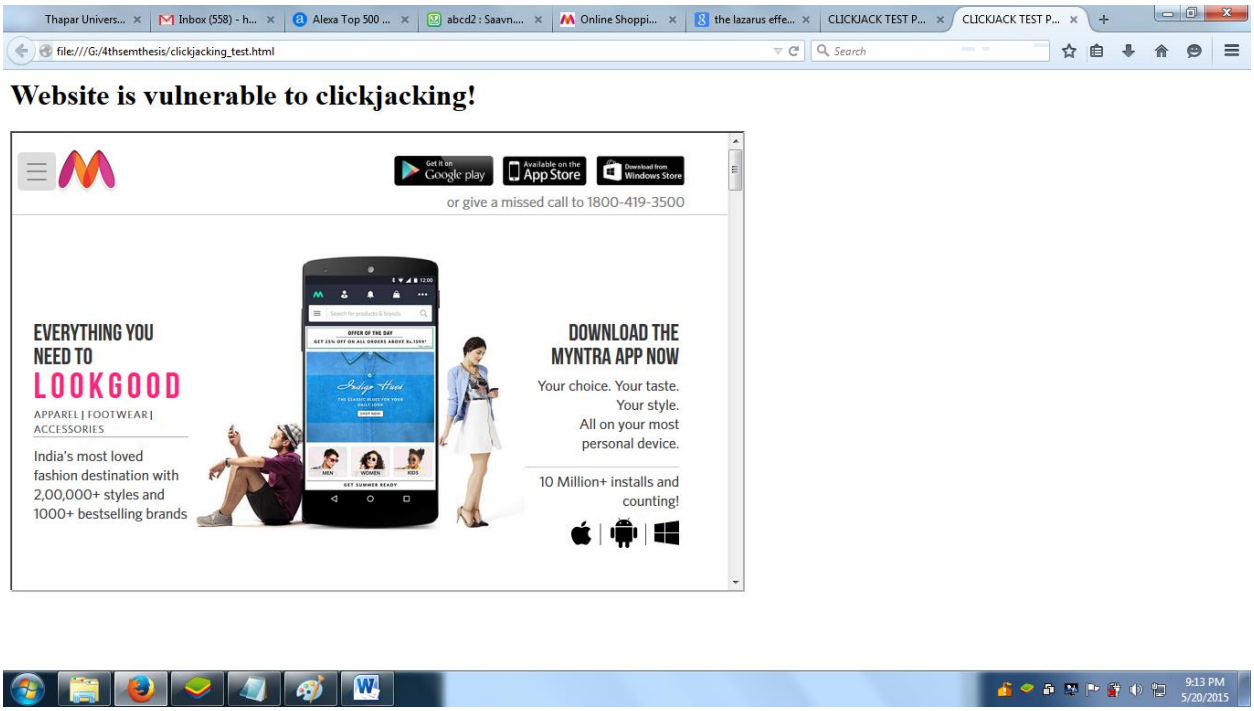
Using the following code we created a web page to implement clickjacking

Step 1: we tried to frame a webpage of some known website in an iframe.

Source code:

```
<Html>
   <Head>
     <Title>CLICKJACK TEST PAGE</title>
   </head>
   <body STYLE="background-color: transparent">
     <H1>Website is vulnerable to clickjacking'</H1>
     <iframe    src="http://www.targetwebsite.c m/"    width="800"
height="500" ALLOWTRANSPARENCY="true"></iframe>
   </body>
</html>
```

Figure 7.1 Clickjacking framing webpage in iframe



We found that the target website was successfully loaded in the iframe.

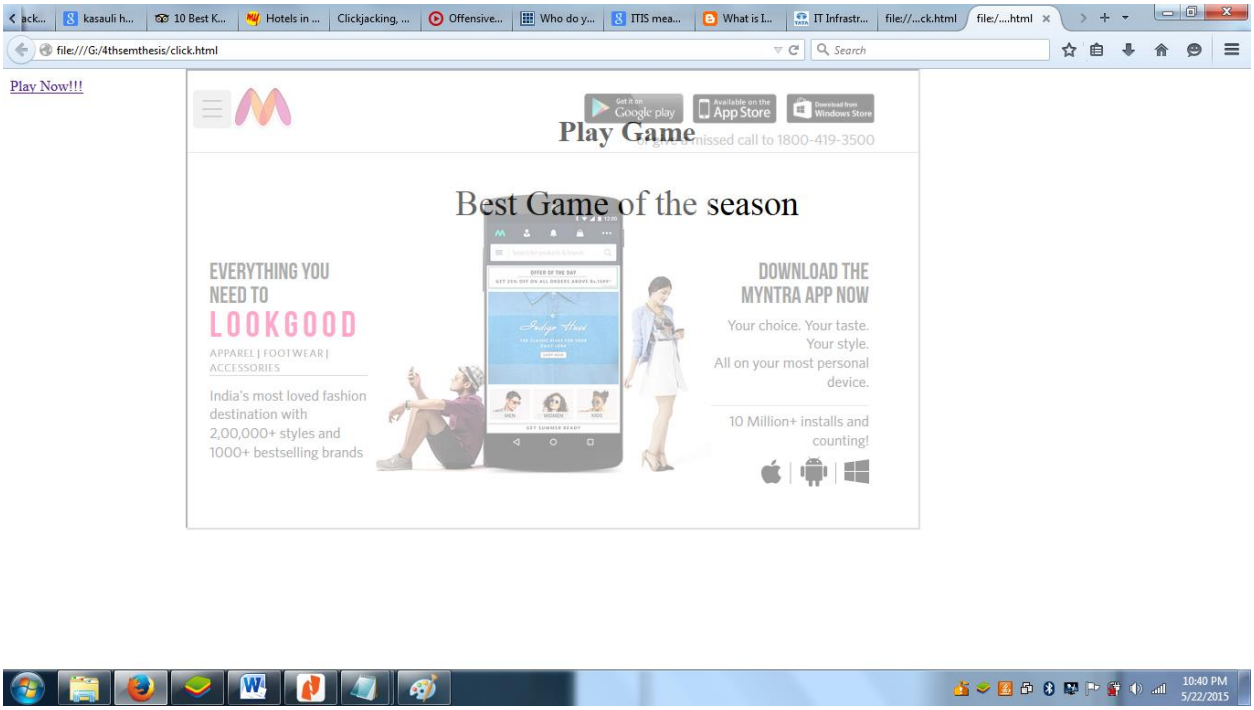
Step 2: Now with a slight modification we created a partially opaque iframe that loaded the target website inside it.

Source code:

```
<Html>
<a href="#">Play Now!!!</a>
<h1 style="text-align: center">Play Game</h1>
<Script>
   window.onbeforeunload = function ()
   {
      Return "Do you want to leave click.site?";
   }
</script>
<Body>
<p   style="font-size:   38px;"align=center>Best   Game   of   the
season</p>
```

```
 <div   style="z-index:   10;   opacity:0.4;   position:   absolute
top:0px; left:200px; ">
  <iframe  scrolling="no"  style="  width:  800px;  height:  500px;'
src="http://www.myntra.com/"> </iframe>
 </div>
<div style="top: 0px; left: 200px ;">
</div>
</body>
</html>
```

Figure 7.2 web page loaded in partially transparent iframe



Step2:

This web page was hosted on the virtual machine that acted as the attacker machine.

Step 3: Now our target machine is the system on which CAAPS is installed.

Then we tried to determine whether or not the system is able to find critical tokens in the web content that are related to clickjacking.

CAAPS captures the network traffic that is going from server to the clients. It basically tries to recognize following type of tokens.

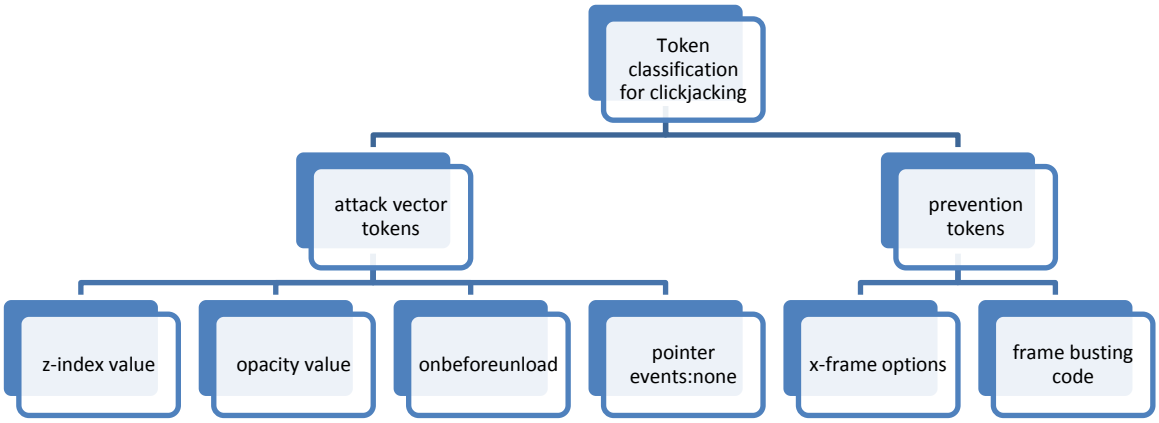


Figure 7.3 Classification of tokens in CAAPS

Step 4: On accessing this web page we found that our CAAPS system is able to detect the clickjacking tokens and is able to alert the user and at the same time block this link.

CHAPTER 8: RESULTS

We evaluated CAAPS by visiting ALEXA top websites and found following websites lacking basic clickjacking prevention mechanisms. These are the websites that are visited day to day by many people.

Website	IP on which website is hosted
www.accuweather.com	63.146.70.50 63.146.70.35
www.alibaba.com	205.204.96.36 198.11.132.23
www.apple.com/	17.172.224.47 17.178.96.59 17.142.160.59
www.ask.com	66.235.120.127
www.cnn.com	157.166.226.25 157.166.226.26
filehippo.com	69.172.201.208
www.irctc.com	202.93.154.101
www.airtel.com	59.145.174.210 125.21.240.17
www.tatadocomo.com	209.54.49.112
www.India.acm.org	64.238.147.38
www.Ieee.org	140.98.193.141
www.springer.com/in/	62.50.45.35
www.dailymail.co.in	209.99.40.220
www.windows.microsoft.com	134.170.119.140
www.saavn.com	46.137.244.142

www.jabong.com	23.62.114.87
www.myntra.com	180.179.147.11

CHAPTER 9: CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

Clickjacking is a new form of attack that exploits the security loopholes in the web applications and widgets. This thesis has presented CAAPS(clickjacking alert and prevention system) as a prototype system that is capable of detecting the attack signatures and adding prevention code on the fly .With some more effort and research one can enhance and expand its functionality .This prototype system can also be used for penetration testing of the web content against possible clickjacking attack. Every system in this universe is never said to be complete hence CAAPS is not a complete system and thus with time it may need further improvement. This is because the clickjacking is such an advanced attack that attackers come up with new attack vectors and hence new attack signatures are to be added to the system.

9.2. Future scope of the project:

The designed system is a prototype system. It works perfectly for a small network. With further testing and enhancement it can be extended to support big network.

The click alert system is based on certain known signatures. These signatures can be updated from time to time.

CHAPTER 10: REFERENCES

[1] R. Hansen, J. Grossman. "Clickjacking," *SecTheory Retrieved, vol.* 5, no. 3, 2008.

[2] M. Niemietz. "UI redressing: Attacks and countermeasures revisited," *in Confidence,* 2011.

[3] R. Lundeen, J. Ou and T. Rhodes, "New ways I'm Going to Hack Your Web App," *Black hat AD*, 2011.

[4]L. Huang, A. Moshchuk, H. Wang, S. Schechter, C. Jackson. "Clickjacking: Attacks and Defenses," *in proc. 21st USENIX conference on Security symposium*, pp. 22-22.

[5]Don't Click Clickjacking Tweet bomb, http://softwareas.com/explainingthe-dont-click-clickjacking-tweetbomb *accessed on 24/04/2014*.

[6]Minor browser UI nitpicking, http://seclists.org/fulldisclosure/2010/Dec/328 *accessed on 26/04/2014.*

[7] P. P. Koch, "Frame busting".

[8]Stealing mouse clicks for banner fraud, http://ha.ckers.org/blog/20070116/stealing-mouse-clicks-for-banner-fraud/ accessed on 27/04/2014.

[9] About CSS attacks, http://css-attacks.html *accessed on 27/04/2014.*

[10] Cursorjacking again, http://blog.kotowicz.net/2012/01/cursorjacking-again.html *accessed on 30/04/2014.*

[11] Proof of Concept-Cursorjacking, http://static.vulnerability.fr/noscript-cursorjacking.html, *accessed on 5/05/2014.*

[12] Stroke triggered XSS and Strokejacking,http://blog.andlabs.org/2010/04/stroke-triggered-xss-and-strokejacking_06.htm accessed on 11/5/2014.

[13]Facebook adds speed to slow down likejackers, http://nakedsecurity.sophos.com/2011/03/30/facebook-adds-speed-bump-to-slow-down-likejackers/ accessed on 15/04/2014.

[14] T. Gondrom, D. Ross. "HTTP Header X-Frame-Options," Draft for WEBSEC IETF, 2013.

[15] H. J. Wang, C. Grier, A. Moshchuk, S. T. King, P. Chaudhary, & H. Venter ,"The Multi-Principal OS Construction of the Gazelle Web Browser," In proc. *18th conference on USENIX Security Symposium (Microsoft Research)* ,Vol. 28,2009.

[16] J. Grossman, "Clickjacking-OWASP AppSec Talk," 2008.

[17] G. McGraw, "Silver bullet talks with Jeremiah Grossman," (IEEE) in *Security & Privacy,* vol.7, no. 2, pp. 10-14, 2009.

[18]E. Bordi, "Proof of concept-cursorjacking (no script)," 2010.

[19] J. A. Shamsi, S. Hameed, W. Rahman, F. Zuberi, K. Altaf & A. Amjad, "Clicksafe: Providing Security against Clickjacking Attacks," In *High-Assurance Systems Engineering (HASE), IEEE 15th International Symposium on*, pp. 206-210, 2014.

[20]H. Shahriar, V.K. Devendran, & H. Haddad, "ProClick: a framework for testing clickjacking attacks in web applications," In *proc. of the 6th International Conference on Security of Information and Networks*, pp. 144-151, 2013.

[21]M. Niemietz,"UI redressing: Attacks and countermeasures revisited," In *Confidence. 2011.*

[22]M. Johns, S. Lekies, B. Braun, & B. Flesch. "BetterAuth: web authentication revisited," (ACM) In *Proc. of the 28th Annual Computer Security Applications Conference* (pp. 169-178).2012

[23]B. Lundeen & J. Alves-Foss. "Practical clickjacking with BeEF,"In *Homeland Security (HST), 2012 IEEE Conference on Technologies for,* pp. 614-619, 2012.

[24]G. Maone, "Hello clearclick, goodbye clickjacking!" Blog, October 2008, http://hackademix.net /2008/10/08 hello-clearclick-goodbye-clickjacking/.

[26] B. V. Ramaraju, & s. Suganya. "A Unique Solution for Clickjacking Attack Using Regex Based Code Crawler". *Software Engineering and Technology*, *7*(3), 66-69.2015

[27] D. Kavitha and S. Ravikumar. "Click jacking Vulnerability Analysis and Providing Security against WEB Attacks Using White listing URL analyzer."2015.

[28] I. Gulenko. "Social against social engineering: Concept and development of a Facebook application to raise security and risk awareness". *Information Management & Computer Security*, *21*(2), 91-101.2013

CHAPTER 11: BIBLIOGRAPHY

1. David Gourley and Brain. HTTP: The Definitive Guide. O'Reilly Media, Inc., 27-Sep-2002.

2. Herbert Schildt. Java The Complete Reference, 8th ed., McGraw Hill Professional, 17-Jan-2011.

3. Chris Shiflett. HTTP Developer's Handbook. Sam's Publishing, 2003.

4. Cay S. Horstmann, Gary Cornell. Core Java 2, Volume 1. Sun Microsystems Press, 2005.

CHAPTER12: LIST OF PUBLICATIONS

[1] H. Pal, A.K. Verma, “Clickjacking: A web page can hear and see you”, published in journal Anveshanam: the journal of computer science and application, vol.3 ISSN no. 2279-0101, 2015.

CHAPTER 13: LINK TO THE UPLOADED VIDEO

https://www.youtube.com/watch?v=98I8QOK0vr0&feature=youtu.be

.