RESEARCH INVENTY: International Journal of Engineering and Science
ISSN: 2278-4721, Vol. 1, Issue 6 (October 2012), PP 14-31
www.researchinventy.com

Sigfree-A Signature-Free Buffer Overflow Attack Blocker

Chinta.Srihari, [1] Lalu Banothu, [2]

1. *Asst. Professor. Department of Computer Science Engineering, School of Engineering GuruNanak Institutions Technical Campus, Hyderabad, India.*
2. *Assoc. Professor. Department of Computer Science Engineering, School of Engineering GuruNanak Institutions Technical Campus, Hyderabad, India.*

Abstract: *This project propose SigFree, a real time, signature-free, out-of the-box, application layer blocker for preventing buffer overflow attacks, one of the most serious cyber security threats. Sigfree can filter out code-injection buffer overflow attack messages targeting at various Internet services such as web service. Motivated by the observation that buffer overflow attacks typically contain executables whereas legitimate client requests never contain executables in most Internet services, SigFree blocks attacks by detecting the presence of code. SigFree first blindly dissembles and extracts instruction sequences from a request. It then applies a novel technique called code abstraction, which uses data flow anomaly to prune useless instructions in an instruction sequence. Finally it compares the number of useful instructions to a threshold to determine if this instruction sequence contains code. SigFree is signature free, thus it can block new and unknown buffer overflow attacks; SigFree is also immunized from most attack-side code obfuscation methods. Since SigFree is transparent to the servers being protected, it is good for economical Internet wide deployment with very low deployment and maintenance cost. We implemented and tested SigFree; our experimental study showed that SigFree could block all types of code injection attack packets (above 250) tested in our experiments. Moreover, SigFree causes negligible throughput degradation to normal client requests*.

Keywords *— Intrusion detection, buffer overflow attacks, code-injection attacks.*

I Introduction

Throughout the history of cyber security, buffer overflow is one of the most serious vulnerabilities in computer systems. Buffer overflow vulnerability is a root cause for most of the cyber attacks such as server breaking in, worms, zombies, and botnets. A buffer overflow occurs during program execution when a fixed-size buffer has had too much data copied into it. This causes the data to overwrite into adjacent memory locations, and depending on what is stored there, the behavior of the program itself might be affected [1]. Although taking a broader viewpoint, buffer overflow attacks do not always carry binary code in the attacking requests (or packets),1 code-injection buffer overflow attacks such as stack smashing probably count for most of the buffer overflow attacks that have happened in the real world.

Although tons of research has been done to tackle buffer overflow attacks, existing defenses are still quite limited in meeting four highly desired requirements: (R1) simplicity in maintenance; (R2) transparency to existing (legacy) server OS, application software, and hardware; (R3) resiliency to obfuscation; (R4) economical Internet-wide deployment. As a result, although several very secure solutions have been proposed, they are not pervasively deployed, and a considerable number of buffer overflow attacks continue to succeed on a daily basis. To see how existing defenses are limited in meeting these four requirements, let us break down the existing buffer overflow defenses into six classes, which we will review shortly in Section 2: (1A) Finding bugs in source code.(1B) Compiler extensions. (1C)OS modifications.(1D) Hardware modifications. (1E) Defense-side obfuscation [3], [4]. (1F) Capturing code running symptoms of buffer overflow attacks [5], [6], [7], [8]. (Note that the above list does not include binary-code-analysis-based defenses, which we will address shortly.) We may briefly summarize the limitations of these defenses in terms of the four requirements as follows: 1) Class 1B, 1C, 1D, and 1E defenses may cause substantial changes to existing (legacy) server OSes, application software, and hardware, thus they are not transparent. Moreover, Class 1E defenses generally cause processes to be terminated. As a result, many businesses do not view these changes and the process termination overhead as economical deployment. 2) Class 1F defenses can be very secure, but they either suffer from significant runtime overhead or need special auditing or diagnosis facilities, which are not commonly available in commercial services. As a result, Class 1F defenses have limited transparency and potential for economical deployment. 3) Class 1A defenses need source code, but source code is unavailable to many legacy applications. Besides buffer overflow defenses, worm signatures can be generated and used to block buffer overflow attack packets [9], [10], [11]. Nevertheless, they are also limited in meeting the four requirements, since they either rely on signatures, which introduce maintenance overhead, or are not very resilient to attack-side obfuscation.

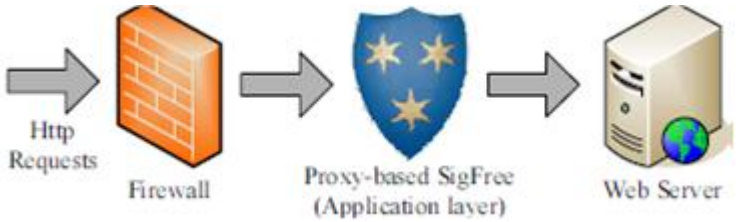


Figure 1 : Architecture of the system

1.2 SCOPE:

To overcome the above limitations, in this paper, we propose SigFree, an online buffer overflow attack blocker, to protect Internet services. The idea of SigFree is motivated by an important observation that "the nature of communication to and from network services is predominantly or exclusively data and not executable code" [12]. In particular, as summarized in [12], 1) on Windows platforms, most web servers (port 80) accept data only; remote access services (ports 111, 137, 138, 139) accept data only; Microsoft SQL Servers (port 1434), which are used to monitor Microsoft SQL Databases, accept data only. 2) On Linux platforms, most Apache web servers (port 80) accept data only; BIND (port 53) accepts data only; SNMP (port 161) accepts data only; most Mail Transport (port 25) accepts data only; Database servers (Oracle, MySQL, PostgreSQL) at ports 1521, 3306, and 5432 accept data only.Since remote exploits are typically binary executable code, this observation indicates that if we can precisely distinguish (service requesting) messages containing binary code from those containing no binary code, we can protect most Internet services (which accept data only) from code injection buffer overflow attacks by blocking the messages that contain binary code.

- SigFree is signature free, thus it can block new and unknown buffer overflow attacks.
- Without relying on string matching, SigFree is immunized from most attack-side obfuscation methods.
- SigFree uses generic code-data separation criteria instead of limited rules. This feature separates SigFree from [12], an independent work that tries to detect code-embedded packets.
- Transparency. SigFree is an out-of-the-box solution that requires no server side changes.
- SigFree is an economical deployment with very low maintenance cost, which can be well justified by the aforementioned features.

The rest of this paper is organized as follows: In Section 2, we summarize the work related to ours. In Section 3, we give an overview of SigFree. In Sections 4 and 5, we introduce the instruction sequence distiller component and the instruction sequence analyzer component of SigFree, respectively. In Section 6, we show our experimental results. Finally, we discuss some remaining research issues in Section 7 and conclude this paper in Section 8.

We proposed SigFree, a realtime, signature free, out of- the-box blocker that can filter code-injection buffer overflow attack messages, one of the most serious cyber security threats, to various Internet services. SigFree does not require any signatures, thus it can block new, unknown attacks.

II Existing System

Detection of Data Flow Anomalies There are static or dynamic methods to detect data flow anomalies in the software reliability and testing field. Static methods are not suitable in our case due to its slow speed; dynamic methods are not suitable either due to the need for real execution of a program with some inputs.

2.2 PROPOSED SYSTEM

Their scheme is rule-based, whereas SigFree is a generic approach which does not require any pre-known patterns. Then, it uses the found patterns and a data flow analysis technique called program slicing to analyze the packet's payload to see if the packet really contains code Four rules (or cases) are discussed in their project: Case 1 not only assumes the occurrence of the call/jmp instructions, but also expects the push instruction appears before the branch; Case 2 relies on the interrupt instruction; Case 3 relies on instruction ret; Case 4 exploits hidden branch instructions. Besides, they used a special rule to detect polymorphic exploit code which contains a loop. Although they mentioned that the above rules are initial sets and may require updating with time, it is always possible for attackers to bypass those pre-known rules. Moreover, more rules mean more overhead and longer latency in filtering packets. In contrast, SigFree exploits a different data flow analysis technique, which is much harder for exploit code to evade.

We proposed SigFree, a realtime, signature free, out of- the-box blocker that can filter code-injection buffer overflow attack messages, one of the most serious cyber security threats, to various Internet services. SigFree does not require any signatures, thus it can block new, unknown attacks.

III Architecture Design:

The phase of the design of computer architecture and software architecture can also be referred to as high-level design. The architectural design defines the relationship among major structural elements of the program.

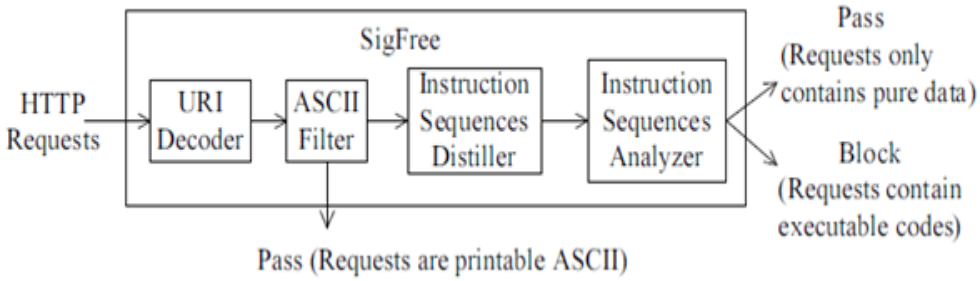


Figure 2: Architecture Design

3.2 Module Design:

The module design phase can also be referred to as low-level design. The designed system is broken up into smaller units or modules and each of them is explained so that the programmer can start coding directly.

3.3 Main Modules:

3.3.1 Prevention/Detection of Buffer Overflows:

Throughout the history of cyber security, buffer overflow is one of the most serious vulnerabilities in computer systems. Buffer overflow vulnerability is a root cause for most of the cyber attacks such as server breaking-in, worms, zombies, and botnets. Buffer overflow attacks are the most popular choice in these attacks, as they provide substantial control over a victim.

Class 1A: Finding bugs in source code:

Buffer overflows are fundamentally due to programming bugs. Accordingly, various bug-finding tools have been developed. The bug-finding techniques used in
these tools, which in general belong to static analysis, include but not limited to model checking and bugs-asdeviant- behavior.

Class 1B Compiler extensions:

"If the source code is available, a developer can add buffer overflow detection automatically to a program by using a modified compiler."

Class 1C: OS modifications:

Modifying some aspects of the operating system may prevent buffer overflows such as Pax , LibSafe and e-NeXsh. Class 1C techniques need to modify the OS. In contrast, SigFree does not need any modification of the OS.

Class 1D: Hardware modifications*:*

A main idea of hardware modification is to store all return addresses on the processor [41]. In this way, no input can change any return address.

Class 1E: Defense-side obfuscation:

Address Space Layout Randomization (ASLR) is a main component of PaX . Bhatkar and Sekar proposed a comprehensive address space randomization scheme. Address space randomization, in its general form, can detect exploitation of all memory errors.

Class 1F:Capturing code running symptoms of buffer overflow attacks:

Fundamentally, buffer overflows area code running symptom. If such unique symptoms can be precisely captured, all buffer overflows can be detected.

3.4 DESIGN OVERVIEW

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

Data flow diagram:

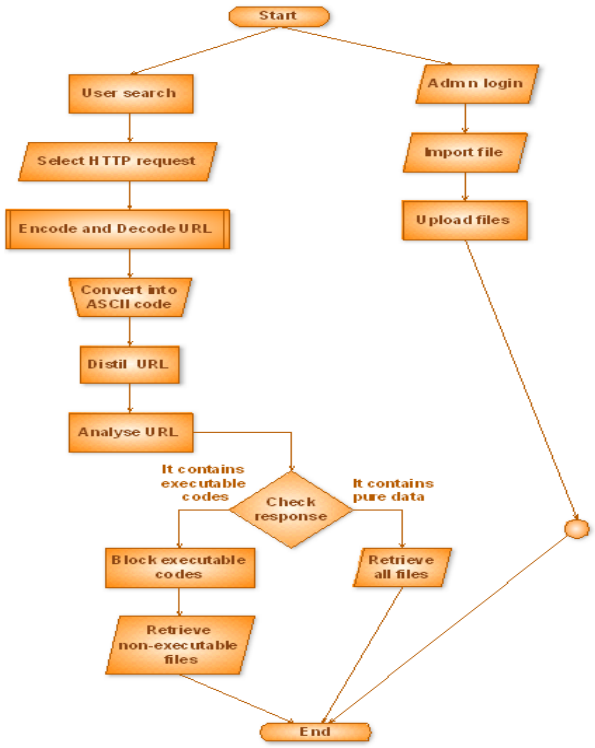


Figure 3: Data Flow Diagram

Goal of UML:

The primary goals in the design of the UML were:

- Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
- Provide extensibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of the OO tools market.
- Support higher-level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

The UML standard defines nine types of diagram:

- **Use Case**: models the functionality of a system in terms of user interaction

3.5 Unified Modeling Language:

The Unified Modeling Language (UML) is a visual modeling language used to specify, visualize, construct and document a software intensive system. The embedded real-time software systems encountered in applications such as telecommunications, school systems, aerospace, and defense typically tends to be large and extremely complex. It is crucial in such systems that the software is designed with a sound architecture. A good architecture not only simplifies construction of the initial system, but also, readily accommodates changes forced by a steady stream of new requirements.

- **Class**: define the structure of a system
- **Activity**: model the flow of a system from action to response
- **Collaboration**: represents interaction between objects as a series of messages
- **Component**: describe the organization of software components
- **Deployment**: depict the physical resources of a system
- **Object**: describe the static structure of a system at a particular time.
- **Sequence**: describes the interaction between classes in terms of message exchange
- **Statechart**: describe the dynamic behavior of a system in response to external stimulus

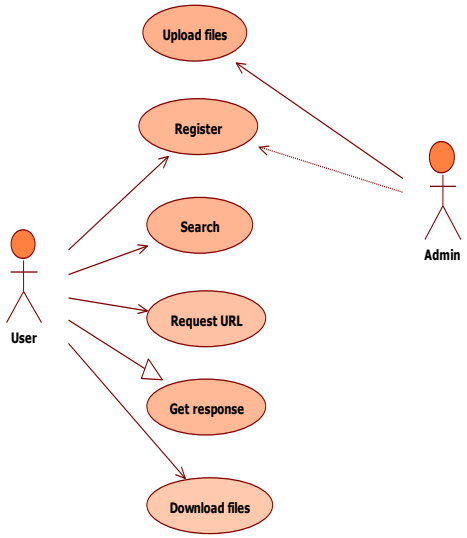


Figure 4: Usecase Diagram

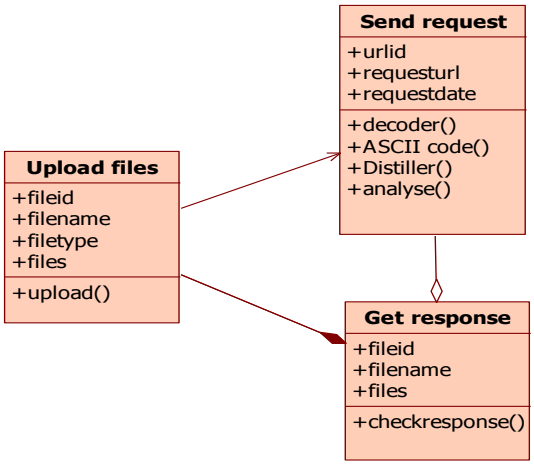


Figure 5: Class Diagram

IV Implementation

Implementation includes all those activities that take place to convert from the old system to the new. The new system may be totally new; replacing an existing manual or automated system, or it may be a major modification to an existing system. Proper implementation is essential to provide reliable system to meet the organizational requirements. Successful implementation may not guarantee improvement in the organizational using the new system, as well as, improper installation will prevent any improvement.

The implementation phase involves the following tasks:

- Careful Planning
- Investigation of system and constraints
- Design of methods to achieve the changeover
- Training of staff in the changeover phase
- Evaluation of changeover

Algorithms used

4.1 Algorithm 1: Distill all instruction sequences from a request

initialize EISG G and instruction array A to empty
for each address i of the request **do**
add instruction node i to G
i ← the start address of the request
while i <= the end address of the request **do**
inst decode an instruction at i
if inst is illegal **then**
A[i]←illegal instruction inst
set type of node i "illegal node" in G
else
A[i]←instruction inst
if inst is a control transfer instruction **then**
for each possible target t of inst **do**
if target t is an external address **then**
add external address node t to G
add edge e(node i, node t) to G
else
add edge e(node i, node i + inst.length) to G
i ← i + 1

4.2 Algorithm 2: To check if the number of useful instructions in an execution path exceeds a threshold

Input: entry instruction of an instruction sequence, EISG G
total 0; useless 0 ; stack empty
initialize the states of all variables to "undefined"
push the entry instruction,states,total and useless to stack
while stack is not empty **do**
pop the top item of stack to i,states,total and useless
if total − useless greater than a threshold **then**
return true
if i is visited **then**
continues
mark i visited
total total + 1
Abstractly execute instruction i (change the states of variables according to instruction i)
if there is a define-define or define-undefine anomaly **then**
useless useless + 1
if there is a undefine-reference anomaly **then**
useless useless + 1
for each instruction j directly following i in the G **do**
push j, states ,total and useless to stack
return false

4.3 PSEDUO CODE

Admin File Upload.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" Code Behind="AdminFileUpload.aspx.cs"
Inherits="Sigfree.AdminFileUpload" MasterPageFile="~/HomeMaster.Master" %>
<asp:Content ID="cnt" ContentPlaceHolderID="mainContent" runat="server">
<asp:Panel ID="fileUpload" runat="server" BorderColor="Blue" BorderStyle="Solid" BorderWidth="2px"
style="padding:10px;">
<div style="float:right;">
<a href="Login.aspx">Logout</a>
</div>
<div>
   <asp:FileUpload ID="FileUpload1" runat="server" />
</div>
<div>
<asp:Button ID="fileUploaded" runat="server" Text="Upload" OnClick="fileUploadedClick" /><br />
<asp:Label ID="lblFileError" runat="server" />
</div>
</asp:Panel>
</asp:Content>
```

ASCII Filter.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="AsciiFilter.aspx.cs"
Inherits="Sigfree.AsciiFilter"
MasterPageFile="~/HomeMaster.Master" %>
<asp:Content ID="content" ContentPlaceHolderID="mainContent" runat="server">
 <div align="center">
 <h3>
 ASCII Filter</h3>
 </div>
  <div align="center">
```

```
 <asp:DataList ID="DataList1" runat="server">
 <HeaderTemplate>
 <table cellpadding="0" cellspacing="0" style="border: 2px solid #E2E2E2;">
 </HeaderTemplate>
 <ItemTemplate>
 <tr>
 <td style="border: 2px solid #E2E2E2;width:130px;">
 <%# Eval("Ascii") %>
</td>
  </tr>
 </ItemTemplate>
 <AlternatingItemTemplate>
 <tr>
  <td style="border: 2px solid #E2E2E2;width:130px; background-color:            #ECF5FA;">
  <%# Eval("Ascii") %>
  </td>
  </tr>
  </AlternatingItemTemplate>
  <FooterTemplate>
  </table>
  </FooterTemplate>
  </asp:DataList>
  <div>
 <asp:Button ID="btnNext" runat="server" OnClick="btnNextClick" Text="Next" />
  </div>
 </div>
</asp:Content>
```

Default.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="Sigfree._Default"  MasterPageFile="~/HomeMaster.Master" %>
<asp:Content ID="content" ContentPlaceHolderID="mainContent" runat="server">
 <asp:DataList ID="DataList1" runat="server">
 <HeaderTemplate>
 <table cellpadding="0" cellspacing="0" >
 </HeaderTemplate>
 <ItemTemplate>
  <tr><td>
<a href="URL.aspx" target="_blank"><asp:Image runat="server" ID="imge" ImageUrl='<%#
Eval("ImagePath").ToString() %>' />
 <br /><asp:Literal ID="User" runat="server" Text='<%# Eval("UserName") %>' />
 </a>
 </td>
 </tr>
</ItemTemplate>
 <FooterTemplate>
 </table>
 </FooterTemplate>
 </asp:DataList>
</asp:Content>
```

Execute.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="execute.aspx.cs"
Inherits="Sigfree.execute" MasterPageFile="~/HomeMaster.Master" %>
<asp:Content ID="content" ContentPlaceHolderID="mainContent" runat="server">
<script type="text/javascript">
function error()

{
```

```
   alert("the executables can be accessed");
}
</script>
<asp:Panel ID="pnl" runat="server" BorderColor="Black" BorderWidth="2px" BorderStyle="Solid"
style="padding:10px;">
<div>
<asp:DataList ID="dlURL" runat="server" >
<HeaderTemplate>
<table cellpadding="2" cellspacing="2" style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;">
<tr>
<th style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;">ExecFileID</th>
<th style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;">ExecFilePath</th>
<th style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;">Created Date</th>
</tr>
</HeaderTemplate>
<ItemTemplate>
<tr>
<td style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;"><%# Eval("ExecFileID")%> </td>
<td style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;"><a onclick="error();"><%#
Eval("ExecFilePath")%></a></td>
<td style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;"><%# Eval("CreatedDate") %></td>
</tr>
</ItemTemplate>
<AlternatingItemTemplate>
<tr>
<td style="border: 1px solid #E2E2E2;background-color: #ECF5FA; height: 40px; padding: 5px;"><%#
Eval("ExecFileID")%> </td>
<td style="border: 1px solid #E2E2E2;background-color: #ECF5FA; height: 40px; padding: 5px;"><a
onclick="error();"><%# Eval("ExecFilePath")%></a></td>
<td style="border: 1px solid #E2E2E2;background-color: #ECF5FA; height: 40px; padding: 5px;"><%#
Eval("CreatedDate") %></td>
</tr></AlternatingItemTemplate>
<FooterTemplate></table></FooterTemplate>
</asp:DataList></div>
</asp:Panel></asp:Content>
```

File.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="File.aspx.cs" Inherits="Sigfree.File"
MasterPageFile="~/HomeMaster.Master" %>
<asp:Content ID="content" ContentPlaceHolderID="mainContent" runat="server">
<asp:Panel ID="pnl" runat="server" BorderColor="Black" BorderWidth="2px" BorderStyle="Solid"
style="padding:10px;">
<div>
<asp:DataList ID="dlURL" runat="server" >
<HeaderTemplate>
<table cellpadding="2" cellspacing="2" style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;">
<tr>
<th style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;">FileID</th>
<th style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;">FilePath</th>
<th style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;">Created Date</th>
</tr>
</HeaderTemplate>
<ItemTemplate>
<tr>
<td style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;"><%# Eval("FileID")%> </td>

<td style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;"><a href='<%# Eval("FilePath")%>'><%#
Eval("FilePath")%></a></td>
```

```
<td style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;"><%# Eval("CreatedDate") %></td>
</tr>
</ItemTemplate>
<AlternatingItemTemplate>
<tr>
<td style="border: 1px solid #E2E2E2;background-color: #ECF5FA; height: 40px; padding: 5px;"><%#
Eval("FileID")%> </td>
<td style="border: 1px solid #E2E2E2;background-color: #ECF5FA; height: 40px; padding: 5px;"><a
href='<%# Eval("FilePath")%>'><%# Eval("FilePath")%></a></td>
<td style="border: 1px solid #E2E2E2;background-color: #ECF5FA; height: 40px; padding: 5px;"><%#
Eval("CreatedDate") %></td>
</tr>
</AlternatingItemTemplate>
<FooterTemplate>
</table>
</FooterTemplate>
</asp:DataList>
</div>
</asp:Panel>
</asp:Content>
```

Images.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Images.aspx.cs"
Inherits="Sigfree.Images" MasterPageFile="~/HomeMaster.Master" %>
<asp:Content ID="content" ContentPlaceHolderID="mainContent" runat="server">
<asp:Panel ID="pnl" runat="server" BorderColor="Black" BorderWidth="2px" BorderStyle="Solid"
style="padding:10px;">
<div>
<asp:DataList ID="dlURL" runat="server" >
<HeaderTemplate>
<table cellpadding="2" cellspacing="2" style="border: 1px solid #E2E2E2; height: 40px; padding: 5px ;">
<tr>
<th style="border: 1px solid #E2E2E2; height: 40px; padding: 5px ;">ImageID</th>
<th style="border: 1px solid #E2E2E2; height: 40px; padding: 5px ;">Image Path</th>
<th style="border: 1px solid #E2E2E2; height: 40px; padding: 5px ;"> Created Date</th>
</tr>
</HeaderTemplate>
<ItemTemplate>
<tr>
<td style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;"><%# Eval("ImageID")%> </td>
<td style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;"><a href='<%#
Eval("ImagePath")%>'><%# Eval("ImagePath")%></a></td>
<td style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;"><%# Eval("CreatedDate") %></td>
</tr>
</ItemTemplate>
<AlternatingItemTemplate>
<tr>
<td style="border: 1px solid #E2E2E2;background-color: #ECF5FA; height: 40px; padding: 5px;"><%#
Eval("ImageID")%> </td>
<td style="border: 1px solid #E2E2E2;background-color: #ECF5FA; height: 40px; padding: 5px;"><a
href='<%# Eval("ImagePath")%>'><%# Eval("ImagePath")%></a></td>
<td style="border: 1px solid #E2E2E2;background-color: #ECF5FA; height: 40px; padding: 5px;"><%#
Eval("CreatedDate") %></td>
</tr>

</AlternatingItemTemplate>
<Footer Template>
</table>
```

```
</Footer Template>
</asp:DataList>
</div>
</asp:Panel>
</asp:Content>
```

Instruction Distiller.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="instructiondistiller.aspx.cs"
Inherits="Sigfree.instructiondistiller" MasterPageFile="~/HomeMaster.Master" %>
<asp:Content ID="content" ContentPlaceHolderID="mainContent" runat="server">
<div align="center">
<div>
<h3>Instruction Distiller</h3>
</div>
<asp:DataList ID="DataList1" runat="server">
 <HeaderTemplate>
<table cell padding="0" cellspacing="0" style="border: 2px solid #E2E2E2 ;">
</Header Template>
 <Item Template>
  <tr>
 <td style="border: 2px solid #E2E2E2; width: 130px ;">
  <%# Eval("Instruction") %>
 </td>
 </tr>
 </ItemTemplate>
<AlternatingItemTemplate>
<tr>
<td style="border: 2px solid #E2E2E2; background-color:#ECF5FA;width:200px;">
 <%# Eval("Instruction")%>
 </td>
 </tr>
 </AlternatingItemTemplate>
<FooterTemplate>
 </table>
 </FooterTemplate>
</asp:DataList>
<div>
<asp:Button ID="btnNext" runat="server" OnClick="btnNextClick" Text="Next" />
</div>
</div>
</asp:Content>
```

Login.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Login.aspx.cs"
Inherits="AIOpnionMining.Login"    MasterPageFile="~/HomeMaster.Master" %>
<asp:Content ID="loginContent" ContentPlaceHolderID="mainContent" runat="server">
<script language="javascript" type="text/javascript">
function validate_form()
  {
 if(document.getElementById('<%= txt_username.ClientID %>').value == "")
  {
    document.getElementById('<%= lbl_errmsg.ClientID %>').innerHTML="Username cannot be blank";
        document.getElementById('<%= lbl_errmsg.ClientID %>').style.display="inline";

            return false;
  }
         else if(document.getElementById("<%= txt_password.ClientID %>").value == "")
    {
```

```
            document.getElementById('<%= lbl_errmsg.ClientID %>').innerHTML="Password                cannot
be blank";
  document.getElementById('<%= lbl_errmsg.ClientID %>').style.display="inline";
            return false;
          }
       }
   </script>
   <table width="100%" border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td align="left" valign="top">
       <table width="100%" border="0" cellspacing="0" cellpadding="0">
     <tr>
     <td height="30" align="left" valign="middle">
     <%--<div id="buttons">
                                  <ul>
                                    <li><a href="#"><strong>Home</strong></a></li>
                                    <li><strong>></strong></li>
                                    <li><span><strong>Login </strong></span></li>
                                   </ul>
                                  </div>--%>
        </td>
        </tr>
        <tr>
              <td align="center" valign="top">
              </td>
               </tr>
               <tr>
                 <td align="center" valign="top">
               <table width="63%" border="0" cellspacing="0" cellpadding="0">
                <tr>
                 <td align="right" valign="top">
                  <a href="Default.aspx">Search</a>
                  </td>
                 </tr>
                  <tr>
                  <td align="left" valign="top">
                 <span class="login_user"><strong>
                 <asp:Literal runat="server" Text='Member Login Section' /></strong></span>
                            </td>
                            </tr>
                             <tr>
                            <td align="left" valign="top" class="body_middle_border">
      <table width="100%" border="0" cellspacing="5" cellpadding="0" class="login_border">
                                  <tr>
                                  <td align="right" valign="middle">
                                   
                                   </td>
                                   <td>
                                     
                                    </td>
                                    <td align="left" valign="middle">
                                       

                                       </td>
                                  </tr>
                                  <tr>
                                    <td width="30%" align="right" valign="middle">
                                     <span class="login_user"><strong>
```

```
                                    <asp:Literal runat="server" Text='UserName' /></strong></span>
                                     </td>
                                     <td width="10">
                                       
                                     </td>
                                     <td width="70%" align="left" valign="middle">
  <asp:TextBox ID="txt_username" CssClass="set_inputbox" runat="server" Width="170px" />
 <asp:RequiredFieldValidator ControlToValidate="txt_username"
ErrorMessage='PLEASEENTERUSERNAME'
                                    runat="server" />
                                    </td>
  <%--<input name="userid" type="text" class="set_inputbox" id="userid"/>--%>
                                    </tr>
                                    <tr>
                                    <td width="30%" align="right" valign="middle">
                                     <span class="login_user"><strong>
 <asp:Literal ID="Literal1" runat="server" Text='Password' /></strong></span>
                                      </td>
                                      <td width="10">
                                       
                                      </td>
                                      <td width="70%" align="left" valign="middle">
 <asp:TextBox ID="txt_password" CssClass="set_inputbox" runat="server" TextMode="Password"
Width="170px" />
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" ControlToValidate="txt_password"
  ErrorMessage='Enter Password' runat="server" />
                                      </td>
  <%--<input name="password" type="password" class="set_inputbox" id="password"/>--%>
                                    </tr>
                                    <tr>
                                    <td width="30%" align="right" valign="middle">
                                       
                                      </td>
                                       <td width="10">
                                        
                                        </td>
                                        <td width="70%" align="left" valign="middle">
                                         <div style="float: left; width: 20%;">
<asp:ImageButton ID="imgbtn_login" AlternateText="login" ImageUrl="images/login.png"
                                            runat="server" OnClick="imgbtn_login_Click" />
                                           </div>
                                           <div style="float: left; width: 80%;">
<asp:Label ID="lbl_errmsg" runat="server" Style="color: Red;" Visible="false" />
                                    </div>
                                     </td>
<%--<input name="login" type="image" id="login" src="images/login.png" alt="login" onclick="
validate_form();" />--%>
                                    </tr>
                                    <tr>
                                    <td width="30%" align="right" valign="middle">
                                      

                                      </td>
                                      <td width="10">
                                        
                                       </td>
                                       <td width="70%" align="left" valign="middle">
                                         
```

```
                                    </td>
                                   </tr>
                                 </table>
                                  </td>
                    </tr>
                   </table>
                    </td>
                   </tr>
                   <tr>
                   <td>
                    
    </td>
     </tr>
     </table>
      </td>
      <! -- <td width="20" align="left" valign="top"> </td> -->
      </tr>    </table>
   </asp:Content>
```

URL.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="URL.aspx.cs" Inherits="Sigfree.URL"
MasterPageFile="~/HomeMaster.Master" %>
<asp:Content ID="content" ContentPlaceHolderID="mainContent" runat="server">
<asp:Panel ID="pnl" runat="server" BorderColor="Black" BorderWidth="2px" BorderStyle="Solid"
style="padding:10px;">
<div>
<asp:DropDownList ID="ddlURLList" runat="server" >
</asp:DropDownList>
<asp:Button ID="btnSend" runat="server" Text="Send" OnClick="btnSendClick" />
</div>
<div>
<asp:DataList ID="dlURL" runat="server" >
<HeaderTemplate>
<table cellpadding="2" cellspacing="2" style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;">
<tr>
<th style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;">BufferID</th>
<th style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;">URL</th>
<th style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;">Created Date</th>
</tr>
</HeaderTemplate>
<ItemTemplate>
<tr>
<td style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;"><%# Eval("BufferID") %> </td>
<td style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;"><%# Eval("URL") %></td>
<td style="border: 1px solid #E2E2E2; height: 40px; padding: 5px;"><%# Eval("CreatedDate") %></td>
</tr>
</ItemTemplate>
<AlternatingItemTemplate>
<tr>
<td style="border: 1px solid #E2E2E2;background-color: #ECF5FA; height: 40px; padding: 5px;"><%#
Eval("BufferID") %> </td>

<td style="border: 1px solid #E2E2E2;background-color: #ECF5FA; height: 40px; padding: 5px;"><%#
Eval("URL") %></td>
<td style="border: 1px solid #E2E2E2;background-color: #ECF5FA; height: 40px; padding: 5px;"><%#
Eval("CreatedDate") %></td>
</tr></AlternatingItemTemplate>
<FooterTemplate></table>
</FooterTemplate></asp:DataList>
```

```
</div></asp:Panel>
</asp:Content>
```

URLDecoder.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="URLDecoder.aspx.cs"
Inherits="Sigfree.URLDecoder" MasterPageFile="~/HomeMaster.Master" %>
<asp:Content ID="cnt" ContentPlaceHolderID="mainContent" runat="server">
<div>
<asp:Panel ID="pnlDecoder" runat="server" BorderColor="Blue" BorderStyle="Solid" BorderWidth="2px"
style="padding:10px;">
<div align="center">
<h3>URL Decoder</h3>
</div>
<div style="padding:5px;" align="center">
<asp:Label ID="lblUrl" runat="server" Text="URL :" style="width:120px;padding-left: 20px;" />
<asp:TextBox ID="txtUrl" runat="server"  style="border:1px solid blue; width:130px;"/>
</div>
<div style="padding:5px;"align="center">
<asp:Label ID="lblEncoder" runat="server" Text="Encoder :" style="width:120px;" /> <asp:TextBox
ID="txtEncoder" runat="server"  style="border:1px solid blue;width:130px;" />
</div>
<div style="padding:5px;"align="center">
<asp:Label ID="lblDecoder" runat="server" Text="Decoder :" style="width:120px;" /> <asp:TextBox
ID="txtDecoder" runat="server"  style="border:1px solid blue;width:130px;" />
</div>
<div align="center">
<asp:Button ID="btnNext" runat="server" Text="Next" OnClick="btnNextClick" />
</div>
</asp:Panel>
</div>
</asp:Content>
```

V System Testing

Software Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding, Testing presents an interesting anomaly for the software engineer. During testing, the program to be tested is executed with a set of test cases and the output of the program is performing as it is expected.

5.1 TYPES OF TESTS

5.1.1 Unit Testing:

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

5.2.1 Integration Testing:

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

5.2.2 Functional Testing:

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

Functional testing is centered on the following items:
Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems/Procedures: interfacing systems or procedures must be invoked.

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

5.2.3 System Testing:

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

White Box Testing

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

Black Box Testing

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

Unit Testing:

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

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

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

Features to be tested

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

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

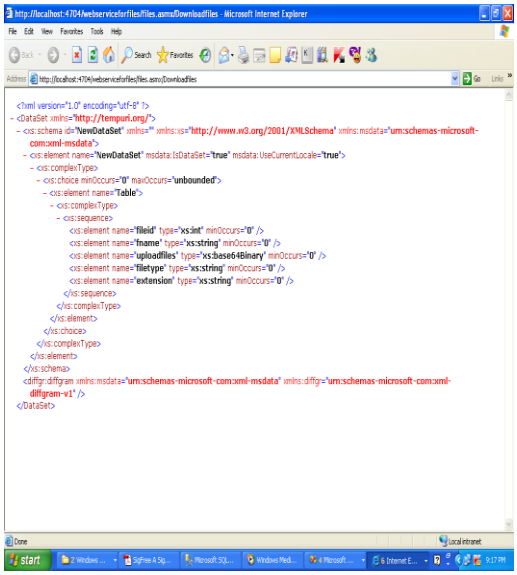
5.2.4 Acceptance Testing:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

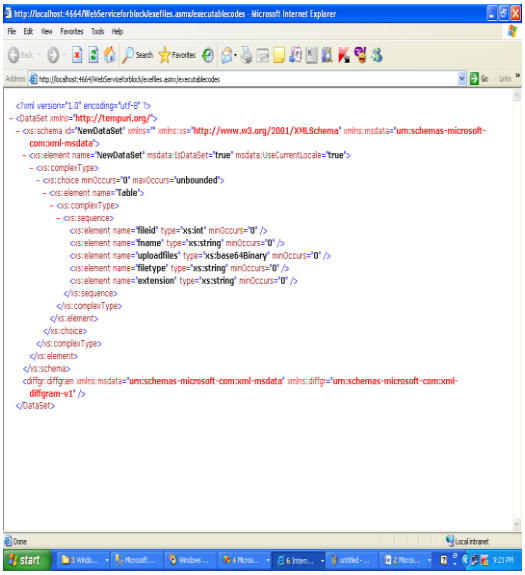
Test Results: All the test cases mentioned above passed successfully. No defects encountered

Screenshots:

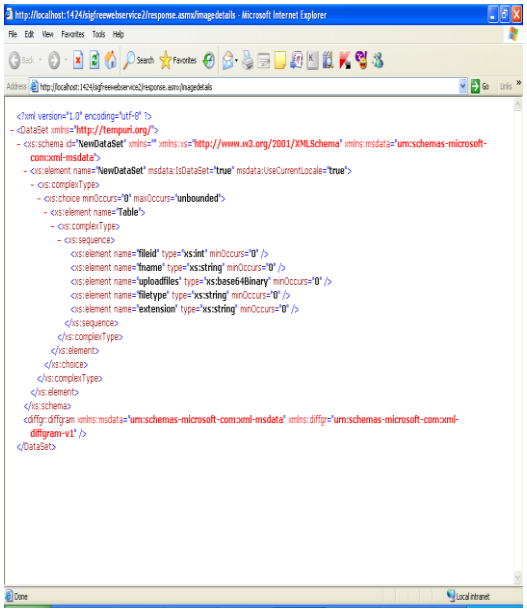
The following pictures screen shorts taken from the system monitor while executing the programme of "SIGFREE-A SIGNATURE-FREE BUFFER OVERFLOW ATTACK BLOCKER" for the better understands of the programming in stepwise.



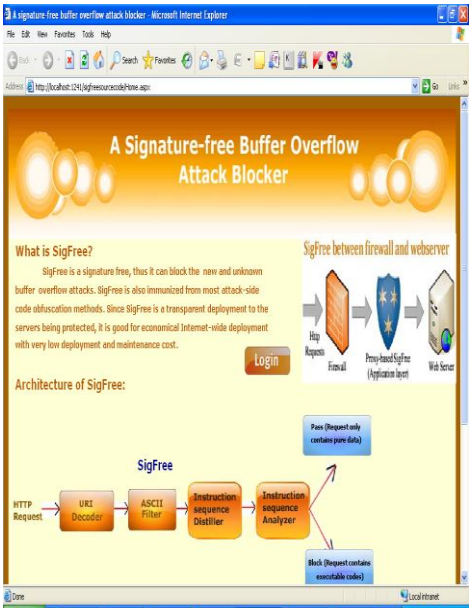
RUNNING FIRST WEB SERVICE:



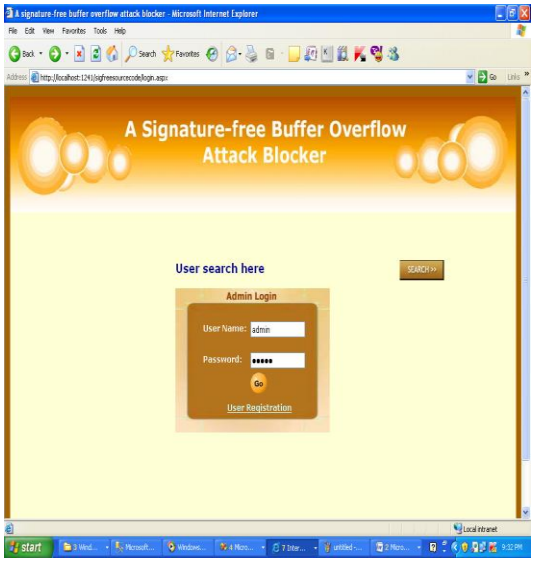
RUNNING SECOND WEBSERVICE



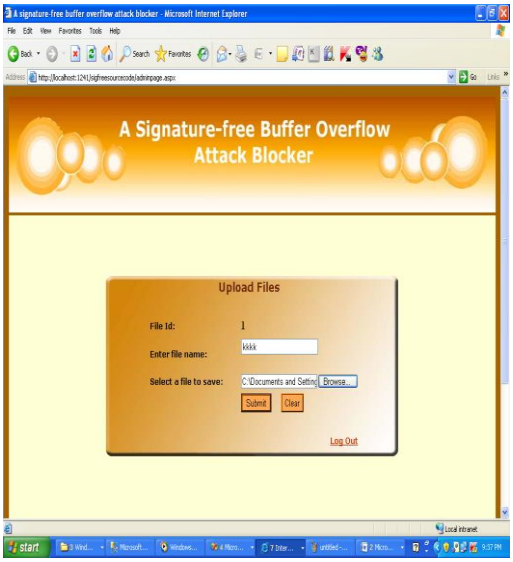
RUNNING THIRD WEBSERVICE:



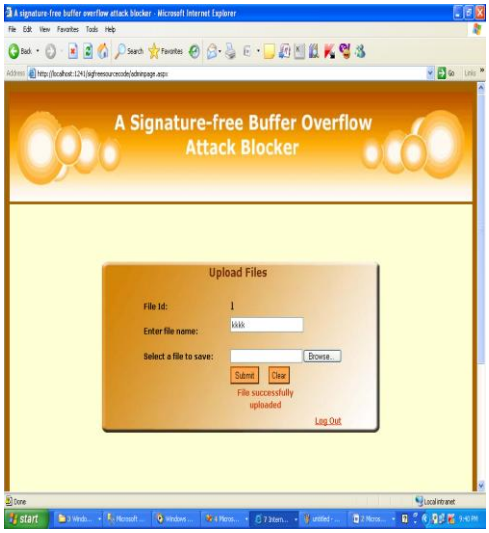
RUNNING THE SOURCE CODE –LOGIN PAGE:



ADMIN LOGIN:

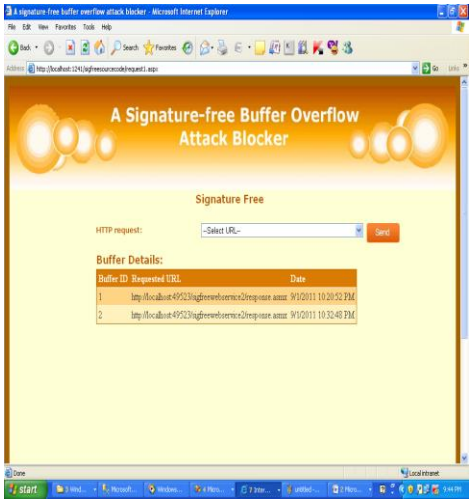


UPLOADING FILES

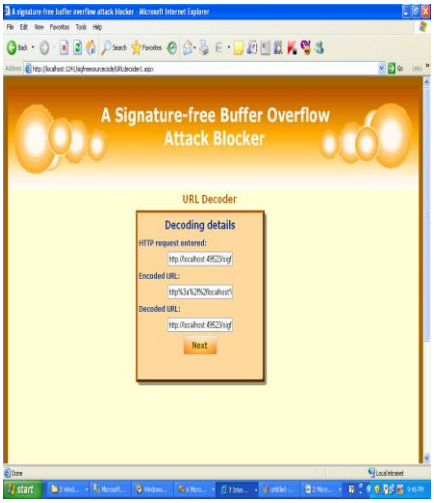


AFTER SUCCESFUL UPLOADING:

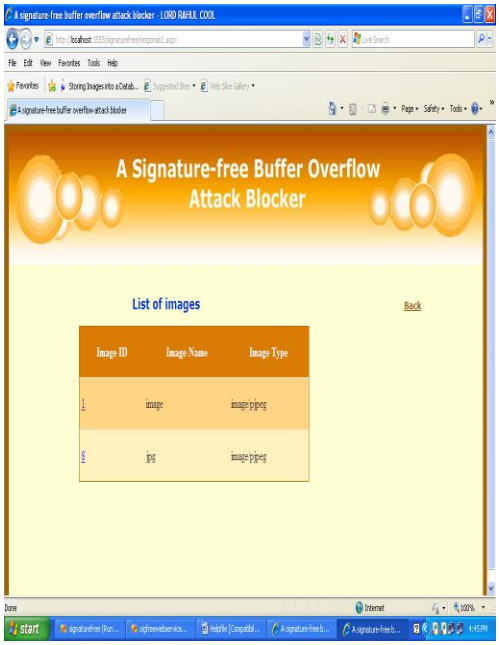
USER SEARCH:



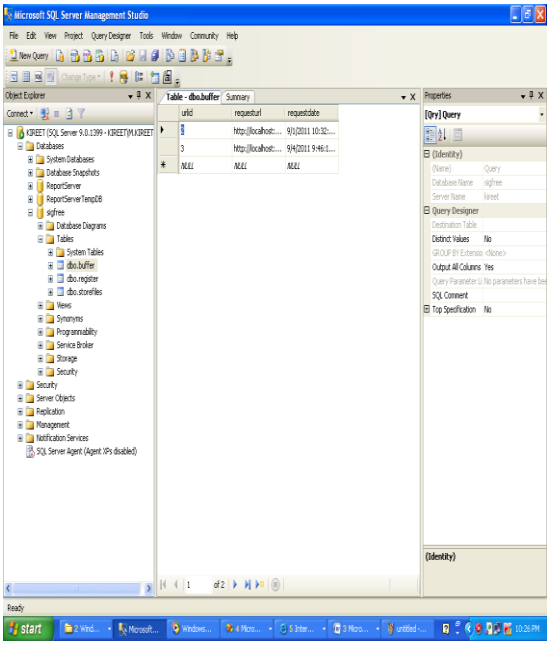
HTTP REQUEST:



URL DECODER CONVERSION:



INSTRUCTION SEQUENCE ANALYSER:



BUFFER:

VI Conclusion

SigFree, a realtime, signature free, out-of- the-box blocker that can filter code-injection buffer overflow attack messages, one of the most serious cyber security threats, to various Internet services. SigFree does not require any signatures, thus it can block new, Unknown attacks. SigFree is immunized from most attack-side code obfuscation methods, good for economical Internet wide deployment with little maintenance cost and negligible throughput degradation, and can also handle encrypted SSL messages.

References

[1]. Buffer overrun in jpeg processing (gdi+) could allow code execution 833987
[2]. http://www.microsoft.com/technet/security/bulletin/ MS04-028.mspx
[3]. Fnord snort preprocessor. http://www.cansecwest.com/spp fnord.c.
[4]. Intel ia-32 architecture software developer's manual volume 1: Basic architecture.
[5]. Metasploit project. http://www.metasploit.com.
[6]. Security advisory: Acrobat and adobe reader plug-in buffer overflow.
[7]. http://www.adobe.com/support/techdocs/321644.html.
[8]. Stunnel – universal ssl wrapper. http://www.stunnel.org.
[9]. Symantec security response: backdoor.hesive.
[10]. http://securityresponse.symantec.com/avcenter/venc/data/backdoor.hesive.html
[11]. Winamp3bufferoverflow. http://www.securityspace.com/smysecure/catid.html?id=11530.
[12]. Pax documentation. http://pax.grsecurity.net/docs/pax.txt, November 2003.
[13]. BARATLOO, A., SINGH, N., AND TSAI, T. Transparent run-time defense against stack
[14]. smashing attacks. In Proc. 2000 USENIX Technical Conference (June 2000).