A SIGNATURE-FREE BUFFER OVERFLOW ATTACK BLOCKER A JAVA .NET PROJECT REPORT(.NET)-2012(FINAL PROJECTS 2030)

Description:

A Signature-free Buffer Overflow Attack Blocker a Java .Net 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 Application

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. In existing system Detection of Data Flow Anomalies are in static or dynamic methods, to detect data flow anomalies in the software reliability and testing field.

```
import javax.servlet.*;

import javax.servlet.http.*;

import java.io.*;

import java.util.*;

import java.sql.*;

public class clientlist extends HttpServlet

{

  public void doGet(HttpServletRequest req,HttpServletResponse res) throws IOException,ServletException

  {

     res.setContentType(“text/html”);

    PrintWriter out=res.getWriter();

    Connection con;

    Statement st;

    ResultSet rs;
```

```
    out.println(“&lt;html&gt;”);

    out.println(“&lt;head&gt;”);

      out.println(“&lt;title&gt; Server: Conference list
&lt;/title&gt;”);

    out.println(“&lt;/head&gt;”);

    out.println(“&lt;body&gt;”);

    out.println(“&lt;form&gt;”);

    out.println(“&lt;br&gt;”);

    out.println(“&lt;br&gt;”);

    out.println(“&lt;h1&gt; &lt;center&gt;List Of Conferencing
Persons &lt;/center&gt; &lt;/h1&gt;” );

    out.println(“&lt;br&gt;”);

    out.println(“&lt;br&gt;”);

      out.println(“&lt;table border=0&gt;”);

    out.println(“&lt;tr valign=center&gt;”);

      out.println(“&lt;td align=center&gt;”);

     out.println(“Emp names”);

      out.println(“&lt;/td&gt; “);

      out.println(“&lt;td align=center&gt;”);

      out.println(“&lt;select name=Empname multiple&gt;”);

    try

    {

      Class.forName(“sun.jdbc.odbc.JdbcOdbcDriver”);
```

```
con=DriverManager.getConnection(“jdbc:odbc:netConference”);
      st=con.createStatement();
      rs=st.executeQuery(“select * from necdata2”);
    while(rs.next())
    {
      String str=rs.getString(1);
      out.println(“&lt;option value=”+str+”&gt;”);
      out.println(str);
    }
    out.println(“&lt;/option&gt;”);
    }
    catch(Exception e)
    {out.println(“Error in…..”);
    }
    out.println(“&lt;/select&gt;”);
    out.println(“&lt;/td&gt;”);
    out.println(“&lt;/tr&gt;”);
    out.println(“&lt;/table&gt;”);
                                        out.println(“&lt;A
HREF=’http://localhost:8080/servlet/conferencescreen2’&gt;
Ready To Paricipate in conference &lt;/A&gt;”);
    out.println(“&lt;/form&gt;”);
```

```
    out.println(“&lt;/body&gt;”);

    out.println(“&lt;/html&gt;”);

}

}
```

Conclusion:

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. In our proposed project this SigFree, a real-time, 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. It 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.