Lab 6: Log/Process/Hashing

Details

Aim: To provide a foundation on how event logs are generated and to determine running processes, and to view and update logs. It also includes methods on using the hashing function.

Activities

1. .NET provides us with an excellent foundation in creating applications in which we can view and log events, as well as monitoring for processes. Another key feature is that it supports many encryption and authentication standards. If Visual Studio is installed on your machine, download the following solution:

http://www.dcs.napier.ac.uk/~bill/eventLog.zip

It has a Windows interface, such as:

Click on the tab buttons for the different programs in this tutorial.

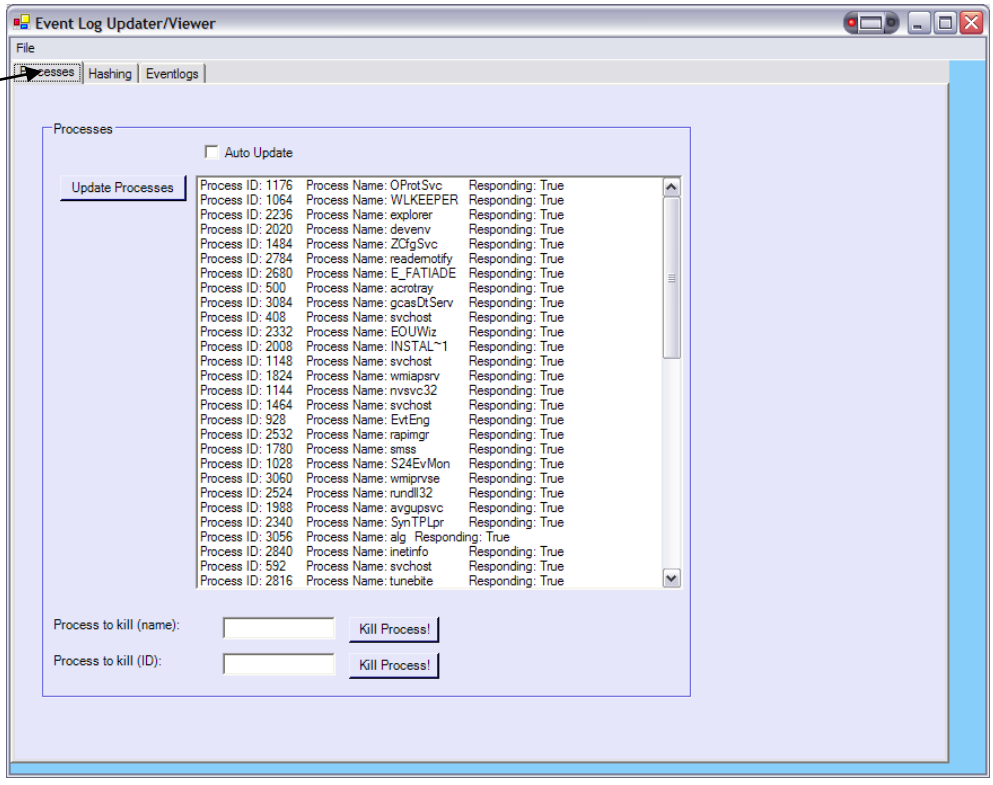

Figure 1: Processes

A Processes

The processes which run on a system are important, especially in monitoring for malicious processes, such as for spyware and trap-door programs, and also in creating

systems which provide audit facilities for event tracking. This part of the lab shows how a program can be written which monitors the programs which are running, and, possibly, kill them.

2. Run the program, and view the processes that are running on your machine.

A few of the processes running are:

ID:	Process Name:	Responding:
ID:	Process Name:	Responding:
ID:	Process Name:	Responding:
ID:	Process Name:	Responding:
ID:	Process Name:	Responding:

3. From the form, double click on the **Kill Process (name)** button, and add the highlighted code:

```
private void button7_Click(object sender, System.EventArgs e)
{
    System.Diagnostics.Process[] p =System.Diagnostics.Process.GetProcesses();

    for(int i=0 ;i<p.Length;i++)
    {
      if (p[i].ProcessName==tbKillProcess1.Text) p[i].Kill();
    }
}
```

4. From the form, double click on the **Kill Process (ID)** button, and add the highlighted code:

```
private void button9_Click(object sender, System.EventArgs e)
{
    System.Diagnostics.Process[] p =System.Diagnostics.Process.GetProcesses();

    for(int i=0 ;i<p.Length;i++)
    {
      if (p[i].Id==Convert.ToInt32(tbKillProcess2.Text)) p[i].Kill();
    }
}
```

6. Now startup up Notepad, and view that it is one of the processes. Now, using the

Kill Process (Name) button, kill the process running Notepad.

Did you see the process, and was it killed properly?

7. Now startup up Notepad, and view that it is one of the processes. Now, using the **Kill Process (ID)** button, kill the process running Notepad.

Did you see the process, and was it killed properly?

B Log files

A key feature in tracing the history of a computer is event log files. This part of the lab shows show to access the event logs on the system.

8. Select the **EventLogs** tab, and add the following code to the **List Application Log** button:

```
for (int i=0;i<listBox1.Items.Count;i++) listBox1.Items.RemoveAt(0);

foreach (System.Diagnostics.EventLogEntry ev in this.eventLogApplication.Entries)
{
   listBox1.Items.Add("Date: " + ev.TimeGenerated+"\tEvent ID: "+
                                ev.EventID+"\tMessage: "+ev.Message);
}
```

9. Add the code for the other buttons (such as **List Security Log** and **List System Log** with their logs). Run the program, and identify the last four logs for each of the event logs (Figure 2):

Last four events for Application log: **Last four events for Security log:** **Last four events for System log:**

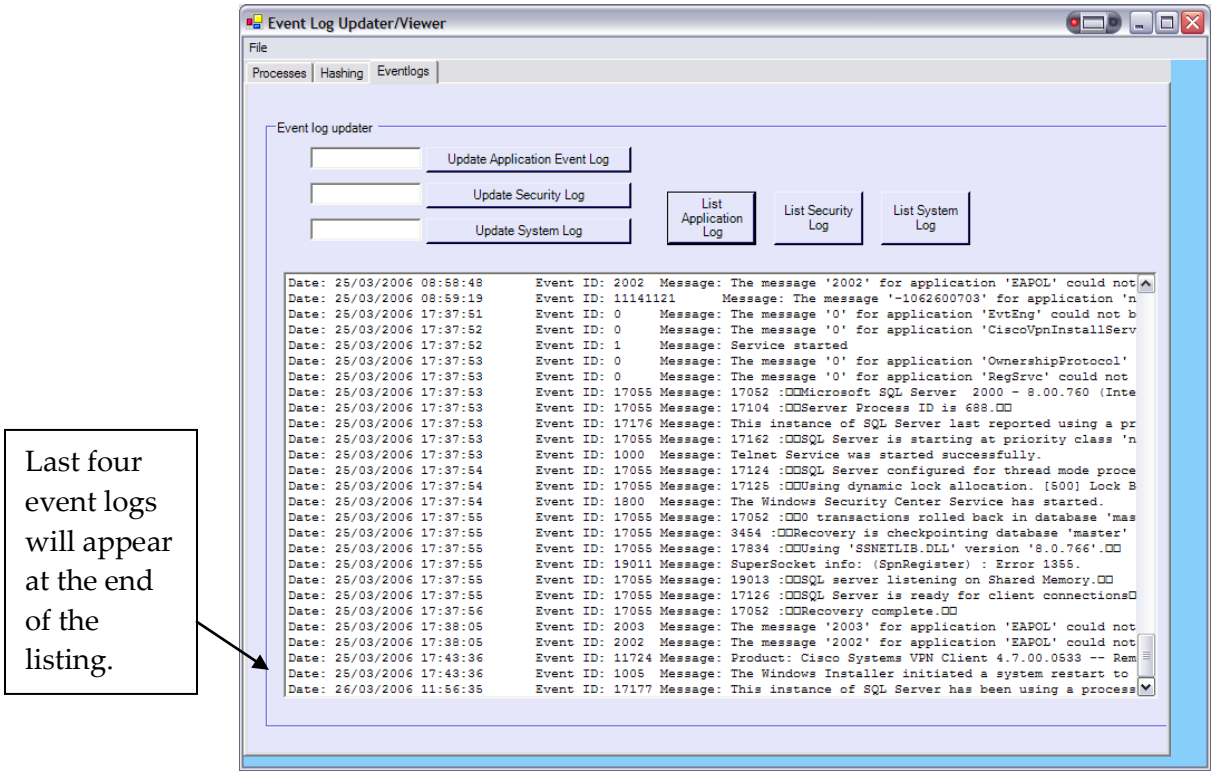

Figure 2: Event logs

10. Next add the following code to the **Update Application Log** button:

```
this.eventLogApplication.Source="My Application";
this.eventLogApplication.WriteEntry(textBox1.Text,EventLogEntryType.Warning);
```

11. Next add the following code to the **Update Security Log** button:

```
this.eventLogSecurity.Source="My Security";
this.eventLogSecurity.WriteEntry(textBox2.Text,EventLogEntryType.Warning);
```

12. Next add the following code to the **Update System Log** button:

```
this.eventLogSystem.Source="My System";
this.eventLogSystem.WriteEntry(textBox2.Text,EventLogEntryType.Warning);
```

13. Run the program, and add a message to each of the logs.

Did each of the logs update?

Verify that the message has been added to the Event Viewer logs [Control Panel->Admin Tools->Event Viewer (Figure 3) – right-click on My Computer and select Manage].

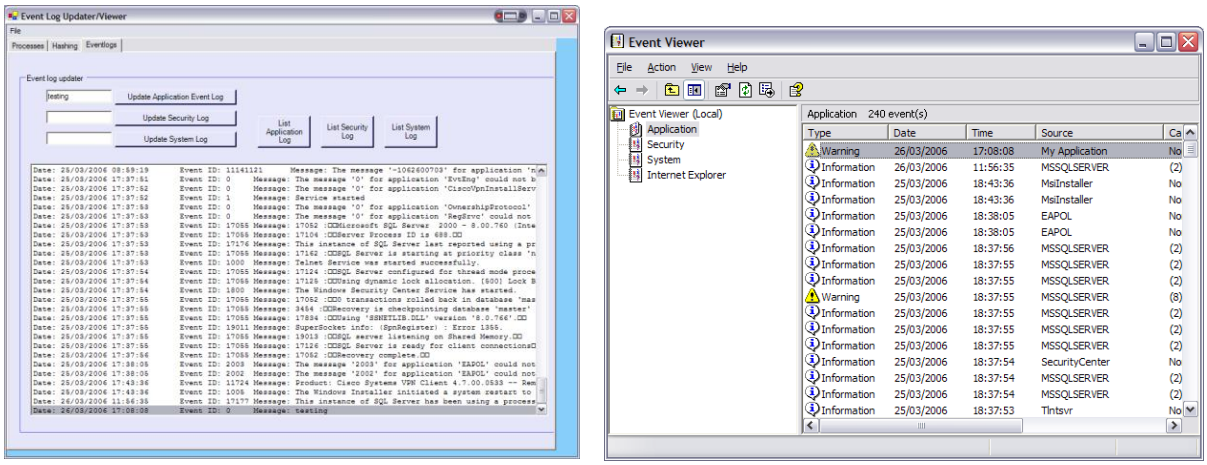

Figure 3: Event log

14. Determine the range of messages possible by modify the **EventLogEntryType** parameter:

```
EventLogEntryType.Warning
```

15. Update the program so that it shows an Error type, and also for Information type.

What is the icon used for an Error type:

What is the icon used for an Information type:

What is the icon used for a Warning type:

C Hash signatures

The hash signature is a key feature of creating dependable authentication for systems, especially for file signatures. In this part of the lab you will open a file, and generate a hash signature for it.

16. Select the **Hashing** tab, and add the following code to the **Open File** button:

```
byte [] buff = new byte[9999999]; // up to 9,999,999 bytes
string hashString="";

openFileDialog1.ShowDialog();
string fname = openFileDialog1.FileName;

tbFile.Text= fname;

FileStream  fs = File.OpenRead(fname);
BinaryReader br = new BinaryReader(fs);

int count = br.Read(buff,0,9999999);
```

```
MD5 md5 = new MD5CryptoServiceProvider();

byte[] result = md5.ComputeHash(buff,0,count);

for (int i=0;i<result.Length;i++)
{
  hashString+=result[i].ToString("X2"); // hexadecimal to string conversion
}
this.tbHash.Text = hashString;
```

17. Using Notepad, create a file named *YourMatric*.txt, and add the following text to it:

```
This is an example of generating a hash signature for a file.
```

18. Now run your program, and determine the hash signature.

Is the signature: 3e7baacc988a9077ddd1cd82bc6f0a04?

Now download an MD5 program (such as from http://ourworld.compuserve.com/homepages/pagrosse/hash.htm) and verify that the signature is correct.

19. Using Notepad, now modify the file to give the following:

```
This is an example of generating a Hash signature for a file.
```

Is the signature: 00B1A69FC8ED0D7D9195A423851E5427?

20. .NET also has an in-built SHA1 hash signature generator. Modify the program so that it now gives a SHA1 hash signature, such as with:

```
SHA1 sha1 = new SHA1CryptoServiceProvider();
```

How many characters does the SHA1 signature have:

21. Now generate a signature for SHA256, then SHA384, and finally SHA512, and note the number of characters in the signature:

SHA256 characters:
SHA384 characters:
SHA512 characters:

Which gives the more verifiable signature, and why?

Note

The event logs are easily added to the form by dragging the log from the Server Explorer window onto the form (see Figure 4).

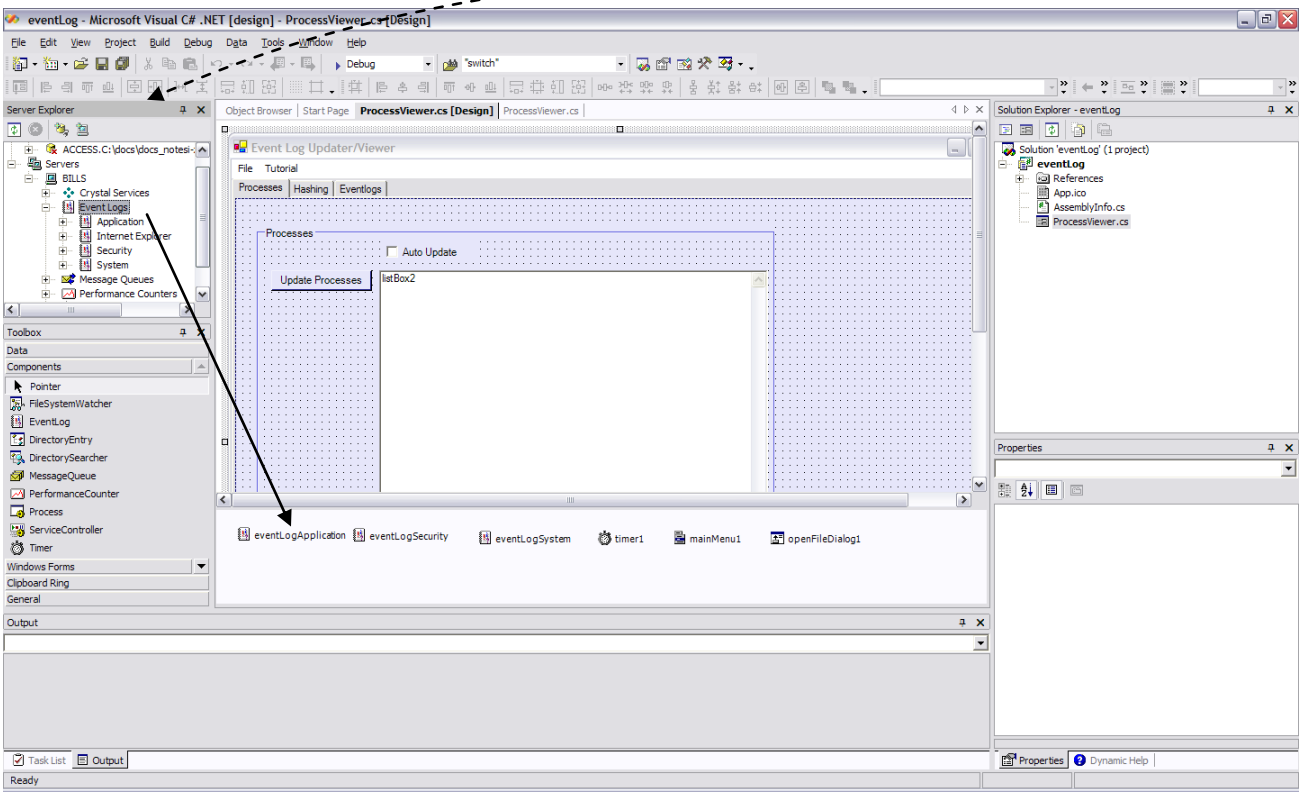

Figure 4: Adding an event log onto a form