C# code auditing

by Ilja van Sprundel
Principal Security Consultant at IOActive, Inc

OWASP London – July 2009

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Using our past to secure your future

London | Chicago | San Francisco | Seattle

www.ioactive.co.uk

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Who am I

- Ilja van Sprundel
- Principal Security Consultant at IOActive Europe
- Top Security Bug Finder on the Final Security Review for Microsoft Vista and Microsoft Windows 2008
- Presented at BlackHat, EuSec, Ruxcon, CCC Congress, BlueHat, Cansec, Pacsec, IT Underground, HiTB, What the Hack
- Published security vulnerabilities on OpenBSD, FreeBSD, Linux, MacOSX, Windows, NetBSD, complete with remote code execution on networking stack
- Netric
- Blog: http://blogs.23.nu/ilja

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

intro

- Strong background in unmanaged languages
- Performed managed code review on behalf of Microsoft for Geneva Framework
- Will present what I learned about c# code auditing
- Some emphasis on webapp c# development
- With asp.net
- Most of this can apply to non-webapps written in c# aswell

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

agenda

- Entrypoints
- Xss
- Arbitrary redirects
- Sql
- Xml
- Unsafe/unmanaged/Pinvoke/Object
- Exception handling
- GC issues
- Integer types/rules

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Entry points

- Webapps in c# are usually stored in 2 files
  - File.aspx
    - Contains the html code and controls that you can steer from the .cs file
    - Can include code in there by doing <% … code … %>
  - File.aspx.cs
    - Code behind the html in .aspx file
    - Contains functions that are entrypoints

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Entry points

- functions
  - Page_Init
    - Where initialization is done
    - This is before the page actually gets loaded
  - Page_Load
    - This is where you're supposed to do most of your work
    - Executes when the requested page is loaded
  - Page_PreRender
    - Is called just before controls are rendered
    - Should only use this to do some last-minute control changes

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Entry points

- Standard functions visual studio declares and you can use
- You can register your own
- By doing something like:

```
this.Load += new
System.EventHandler(this.my_handler);
```

- In Page_Init() for example

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Entry points

- They look like:

```
private void Page_Load(object sender, System.EventArgs e)
private void Page_Init(object sender, EventArgs e)
private void Page_PreRender(object sender,
System.EventArgs e)
```

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Entry points

- Variables
  - Request.QueryString[“param”]
  - fields in .aspx file:

```
Aspx file:
<asp:Label id=“MyLabel" runat="server"></
asp:Label>
Cs file:
MyLabel.Text = QueryString[“blah”];
```

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

xss

- Assume you know what xss is and what you can do with it
- asp.NET has a default xss filter
- Isn't really input validation
- It's exploit mitigation
- Seems like a small difference
- But it's not

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

xss

- Exploit mitigation tries to make exploitation harder
- Input validation should validate all input for correct input.
- asp.NET has no idea what kinda input your app wants, hence it can't do input validation properly
- All it does is try to detect html injection (without breaking functionality) and stop it if it detects html

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

xss

- Sometimes it does break functionality
- If you have a webapp that sends xml data back and forth between client and server
- Need to turn off xss filter for that
- <%@ Page validateRequest="false" %>

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

xss

- Fairly trivial to break
- it looks for “<“
- Followed by a-z, A-Z or space
- Find anything else to inject after the < but before the tagname
- That a browser will still accept
- Bypasses the input filter

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

xss

- Can have input validators bound to a single variable
- Regex's are quite difficult to get right
- Chances are the regex is too tolerant

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

xss

- HtmlEncode()
- HtmlAttributeEncode()
  - much more lightweight than HtmlEncode(), only use this on html attributes !)
- UrlEncode()

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

xss

- Anti-XSS
  - Same one's as asp.net offers
  - XmlEncode()
  - XmlAttributeEncode()
    - Only use on xml attributes, should be more lightweight than XmlEncode()
  - JavaScriptEncode()
    - Only encodes strings! Does not work if you want to put some piece of userdata in a variable name
  - VisualBasicScriptEncode()
    - Only encodes strings! Does not work if you want to put some piece of userdata in a variable name

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Arbitrary redirects

- Redirection looks like:

```
Response.Redirect(Request.QueryString["redir"]);
```

- Arbitrary redirect
- Could be used to spam search engines for example
- Need to do input validation so it only points to your webapp

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Sql injection

- Sql injection usually looks something like

```
sqlq="select * from bands where id=" +
Request.QueryString["id"];
rs=conn.execute(sqlq);
```

- A hacky solution is to do input validation for this yourself.
- not really recommended, too easy to make mistakes

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Sql injection

- A better solution is to use parameterized queries:

```
sql = "select * from bands where id =@id";
cmd.Parameters.Add(New SQLParameter("@id", Request.QueryString["id"]));
```

- This is usually a lot safer
- It's still possible to screw things up

```
sql = "select * from bands where id =@id and id2 =“ + Request.QueryString["id"];
cmd.Parameters.Add(New SQLParameter("@id", Request.QueryString["id"]));
```

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

xml

- XmlTextReader (get pwned)
  - Resolves external entitities
  - DTD, schema
  - Can handle local file://
  - Has no length limits
- XmlReader (DoS)
  - No length limits in place
- XmlDictionaryReader
  - This one is sane. Should be used iso other 2

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Unsafe/unmanaged code

- Marshal class is used to do unmanaged things in c# code.
- Such as manual memory allocation
- Suffers from same security issues as c/c++

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Unsafe/unmanaged code

```
IntPtr hglobal = Marshal.AllocHGlobal(100);
int size;
Marshal.copy(inputfromnetwork, 0, hglobal, 100);
size = Marshal.ReadInt32(hglobal);
IntPtr data = Marshal.AllocHGlobal(size + 2);
Marshal.copy(inputfromnetwork, 4, data, size);
```

- Classic integer overflow in memory allocation
- Leads to heap overflow

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Unsafe/unmanaged code

- Blocks of code can be marked as unsafe
- Scoping then unsafe
- Basically allows you to include something that looks very much like c/c++
- Allows for unmanaged memory allocations, pointers, ….

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Unsafe/unmanaged code

```
int to = 0x41414141;
byte what = 0x42;
unsafe {
        byte *ptr = (byte *) to;
        *ptr++ = what;
}
```

- Assigns 0x41414141 to ptr;
- Derefs pointer and writes 0x42 to it !
- .net does nothing for you when using unsafe
- It all looks like c/c++ again !

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

PInvoke

- PInvoke is used to call functions in dll's
- Usually code around these exported dll's will use unmanaged or unsafe code
- Only way to really use functions in dll's (assuming .net doesn't export the functionality you want)

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Object

- Server.CreateObject()
- Is used to access com objects from within asp(.net)
- A lot of 3rd party IIS com objects (email, file download/upload, ….)
- Using it for these 3rd party objects is usually not a great idea
- 3rd party com objects are usually not really secure at all

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Exception handling issues

- Almost everything in c# uses exceptions
- Exceptions make it way too easy to leak stuff
- It's all to common to see a buch of code wrapped under 1 single try, even those more than one api can throw a multitude of exceptions

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Exception handling issues

- One of two things usually go wrong
  - You miss an exception (and app dies for example)
  - You don't but forget to clean something up in the exception handler

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Exception handling issues

```
...
myclass b = new myclass; // need to call dispose on this one
try {
        byte a[] = new byte[100];
        ...
} catch () {
        bailout(); // throw some exception
}
myclass.dispose;
...
```

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

GC issues

- C#'s GC (garbage collector) kinda sucks at times
- It can be quite slow
- It can take minutes (or even hours!) before it cleans something up
- Just not acceptable in some cases
- Mysql example

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

GC issues: mysql example

```
System.Data.OleDb.OleDbConnection con;
con=new System.Data.OleDb.OleDbConnection("");
con.ConnectionString="Provider=MySQLProv;Data
Source=mysql;";
Try {
        con.Open();
        ... do stuff ...;
        con.Close();
} catch(Exception ex) {
        bailout();
}
```

- Con.Close() not called in case exception is thrown

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

GC issues: mysql example

- GC will call con.Close() when con is garbage collected
- Can take minutes (hours sometimes !)
- Meanwhile connection is still open !
- Now assume you have a very busy webapp
- Let's say 200 connections a minute
- That's 1000 connections in 5 minutes that still need to get GC'ed !
- Totally screws up mysql's connection pool !

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Integer types

- Long is always 64 bits long (unlike vc++ where they're always 32 bits long)
- there is an int64 type in c#
- No unsigned int64 !

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Integer rules

- Integer rules are slightly different from c/c++
- Most are the same, however
- There is no int promotion when doing comparisons

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Integer rules

```
unsigned int maxlen = 256;
BinaryReader binReader =
    new BinaryReader(File.Open(fileName,
FileMode.Open));
int len = binReader.ReadInt32();
if (len > maxlen) {
        bailout();
}
byte a[] = new byte[len];
```

- No int promotion to unsigned done !
- Signed comparison, bypasses maxlen check
- Does a massive allocation (since new_TYPE_ see it as unsigned)

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Integer rules

- Int overflows can throw an exception
- Not on by default
- /checked compiler option
- Can also use checked() to make it throw an exception

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Integer rules

```
size = Marshal.ReadInt32(hglobal);
try {
        IntPtr data = Marshal.AllocHGlobal(checked(size + 2));
}
catch (System.OverflowException e)  {
        bailout();
}
```

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Questions ?