Henry Post

ITMS 443

Vulnerability Assessment Report

Contents

[Target 2](#_Toc527905076)

[Information 3](#_Toc527905077)

[DNS Info 3](#_Toc527905078)

[IP Addresses 3](#_Toc527905079)

[Open ports 3](#_Toc527905080)

[Vulnerabilities 4](#_Toc527905081)

[Determining if a user exists or not 4](#_Toc527905082)

[URL Manipulation & Filesystem Traversal 4](#_Toc527905083)

[Determining if a folder exists 5](#_Toc527905084)

[Determining if a file exists and getting its contents 5](#_Toc527905085)

[Getting the contents of any file anywhere via ‘Null Byte Injection’ 5](#_Toc527905086)

[Getting source code (uh-oh!) of the website 5](#_Toc527905087)

[Exploiting this programmatically (NullByteMap, anyone?) 5](#_Toc527905088)

[Brute force vulnerabilities 6](#_Toc527905089)

[SQL-injectable forms 6](#_Toc527905090)

[Other points of interest 6](#_Toc527905091)

[Use of Regular Expressions in comment storing module 6](#_Toc527905092)

[Developer Comments 6](#_Toc527905093)

[Etc 6](#_Toc527905094)

[Solutions 7](#_Toc527905095)

[Null Byte injection 7](#_Toc527905096)

[Developer comments 7](#_Toc527905097)

[SQL Injection 7](#_Toc527905098)

[Brute force vulnerabilities 7](#_Toc527905099)

# Target

The target, initially, was a single URL:

<http://demo.testfire.net/>

# Information

## DNS Info

* testfire.net
  + demo.testfire.net
  + ftp.testfire.net
  + www.testfire.net

## IP Addresses

* 65.61.137.117

## Open ports

* 80
  + TCP
  + Web server
* 443
  + TCP
  + Microsoft IIS
* 1863
* 8080
  + TCP
  + HTTP Proxy.

# Vulnerabilities

## Determining if a user exists or not

When you login at <http://www.testfire.net/bank/login.aspx> using a random username and password, you get the following error:

**Login Failed: We're sorry, but this username was not found in our system. Please try again.**

Top of Form

Bottom of Form

Now, if I login with a username that exists, such as admin, I get a DIFFERENT error:

**Login Failed: Your password appears to be invalid. Please re-enter your password carefully.**

Top of Form

Bottom of Form

## URL Manipulation & Filesystem Traversal

While viewing the source of the homepage, I noticed how the hyperlinks to different pages were structured.

**<li><a** id="\_ctl0\_\_ctl0\_Content\_MenuHyperLink1" href="default.aspx?content=personal\_deposit.htm"**>**Deposit Product**</a></li>**

In this **<li>** tag, we can see that there is a parameter called content with a value of personal\_deposit.htm. This is sent via an HTTP GET request, and can be easily modified by even just end users.

Curious, I modified it to be a different value, like “cheese.txt”.

I received a developer error page with some pretty juicy information!

**An Error Has Occurred**

**Summary:**

**Could not find file 'C:\downloads\AltoroMutual\_v6\website\static\cheese.txt'.**

**Error Message:**

System.IO.FileNotFoundException: Could not find file 'C:\downloads\AltoroMutual\_v6\website\static\cheese.txt'. File name: 'C:\downloads\AltoroMutual\_v6\website\static\cheese.txt' at System.IO.\_\_Error.WinIOError(Int32 errorCode, String maybeFullPath) at System.IO.FileStream.Init(String path, FileMode mode, FileAccess access, Int32 rights, Boolean useRights, FileShare share, Int32 bufferSize, FileOptions options, SECURITY\_ATTRIBUTES secAttrs, String msgPath, Boolean bFromProxy) at System.IO.FileStream..ctor(String path, FileMode mode, FileAccess access, FileShare share, Int32 bufferSize, FileOptions options) at System.IO.StreamReader..ctor(String path, Encoding encoding, Boolean detectEncodingFromByteOrderMarks, Int32 bufferSize) at System.IO.StreamReader..ctor(String path) at System.IO.File.OpenText(String path) at Altoro.Default.LoadFile(String myFile) in c:\downloads\AltoroMutual\_v6\website\default.aspx.cs:line 42 at Altoro.Default.Page\_Load(Object sender, EventArgs e) in c:\downloads\AltoroMutual\_v6\website\default.aspx.cs:line 70 at System.Web.Util.CalliHelper.EventArgFunctionCaller(IntPtr fp, Object o, Object t, EventArgs e) at System.Web.Util.CalliEventHandlerDelegateProxy.Callback(Object sender, EventArgs e) at System.Web.UI.Control.OnLoad(EventArgs e) at System.Web.UI.Control.LoadRecursive() at System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolean includeStagesAfterAsyncPoint)

The first thing I noticed was, well, I now know where the webserver resides on whatever system is hosting it, which is C:\downloads\AltoroMutual\_v6\website.

I also know that it’s written in C#, likely ASP.NET, and on a Windows box.

### Determining if a folder exists

Because we have a different error page for folders which DO exist (with nonexistent files):

**Could not find file 'C:\downloads\AltoroMutual\_v6\website\text.txt'.**

And ones that DO NOT exist:

**Could not find a part of the path 'C:\downloads\AltoroMutual\_v6\website\folderthatdoesntexist\text.txt'.**

We can essentially ask the website whether or not any folder exists.

### Determining if a file exists and getting its contents

Now, if we know the location of a specific file, we can ask the server for it.

However, it can only end in .htm or .txt, according to the server.

<http://www.testfire.net/default.aspx?content=../../../../Windows/System32/drivers/gmreadme.txt>

If a file exists, like the one above, we’ll be given its contents.

And if it doesn’t, we simply get the error similar to our ‘test.txt’ file above.

### Getting the contents of any file anywhere via ‘Null Byte Injection’

If we embed a null character, aka the numerical value (NOT ASCII) zero aka 0b0, aka 0x0, into the URL, we can do some fun stuff.

<http://www.testfire.net/default.aspx?content=../../../../Windows/System32/drivers/etc/hosts%00.txt>

The above URL will yield the Windows hosts file. This is quite a sensitive location to be reading from!

### Getting source code (uh-oh!) of the website

Using our previous knowledge of the site, its technologies (c#, ASPX, ASP.NET) combined with this Null Byte Injection vulnerability, we can view the source code of the site!

Below is a page which displays part of its own source code. An unintentional [quine](https://en.wikipedia.org/wiki/Quine_(computing))!

<http://www.testfire.net/default.aspx?content=../default.aspx.cs%00.txt>

### Exploiting this programmatically (NullByteMap[[1]](#footnote-1), anyone?)

I have included a Python 3 script that will return files using this exploit.

Run it for yourself!

## Brute force vulnerabilities

There’s multiple web forms, most notably [/bank/login.aspx](http://www.testfire.net/bank/login.aspx), that are easily brute-forceable.

There’s a hydra script that attempts to brute-force this login page.

No lockout procedures or IP banning is implemented as far as I can tell.

## SQL-injectable forms

There are a couple of web forms that seem like good targets for SQL injection, given how much of the site is custom code (with bugs!)

## Other points of interest

### Use of Regular Expressions in comment storing module

When passing malformed/invalid parameters to /comment.aspx, I discovered that I could get the following error:

**An Error Has Occurred**

**Summary:**

**Value cannot be null.**

**Error Message:**

System.ArgumentNullException: Value cannot be null. Parameter name: input at System.Text.RegularExpressions.Regex.IsMatch(String input) at System.Text.RegularExpressions.Regex.IsMatch(String input, String pattern) at Altoro.comment.writeToFile(String file, String name, String email\_addr, String subject, String comments) in c:\downloads\AltoroMutual\_v6\website\comment.aspx.cs:line 31 at Altoro.comment.Page\_Load(Object sender, EventArgs e) in c:\downloads\AltoroMutual\_v6\website\comment.aspx.cs:line 27 at System.Web.Util.CalliHelper.EventArgFunctionCaller(IntPtr fp, Object o, Object t, EventArgs e) at System.Web.Util.CalliEventHandlerDelegateProxy.Callback(Object sender, EventArgs e) at System.Web.UI.Control.OnLoad(EventArgs e) at System.Web.UI.Control.LoadRecursive() at System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolean includeStagesAfterAsyncPoint)

From the GET request below:

<http://testfire.net/comment.aspx?name=asdf&email_addr=asdf&subject=asdf&comments=asdf&submit=+Submit>+

This is most definitely a good further vulnerability to explore.

### Developer Comments

#### L79 of [bank/login.aspx](http://www.testfire.net/bank/login.aspx)

*<!-- To get the latest admin login, please contact SiteOps at 415-555-6159 -->*

#### L87:89 of [feedback.aspx](http://www.testfire.net/feedback.aspx)

*<!--- Dave- Hard code this into the final script - Possible security problem.*

*Re-generated every Tuesday and old files are saved to .bak format at L:\backup\website\oldfiles --->*

**<input** type="hidden" name="cfile" value="comments.txt"**>**

### Etc

* <http://testfire.net/robots.txt>

# Solutions

## Null Byte injection

* Scrub all strings, data, numbers, etc. that come from ANY user
* Restrict access to directories by the webserver
  + Possibly create separate user with only READ ACESS to the webserver root
* Use existing known-good solution for serving .TXT or .HTM files to the user

## Developer comments

* Just don’t put sensitive stuff inside the code!
  + I’m looking at you, Dave!
* All code and HTML should be scrubbed of comments and unnecessary metadata before being served to remove the chance of this happening
  + This also makes the site load faster.

## SQL Injection

* Use prepared statements
* Isolate database users’ privileges from one another

## Brute force vulnerabilities

* Lock out users who are targeted by people trying passwords.
* Have limits on IP login attempts.
* Force users to satisfy a password strength criterion.

1. Refers to SQLMap. Intended to be funny. [↑](#footnote-ref-1)