

Defensive Programming 101

How to be secure and cope with idiots

No matter how well you code they will break it

- Murphy's Law (he was an optimist really!)

Programmers

95% of programmers are

- Smart
- Clever
- Trusting

The other 5% are devious or become management!

Programmers are not natively security conscious

The reason

- Its take a lot longer to write!!

Its not always about code

- Need to know your platform.
- Solid house, bad foundations.

Users

They are the known objective in programming

They are who you should code for

Advanced Users & Hackers

Are just smarter, more curious and sometimes malicious users

Attempt to open the unlocked door

They are what you secure your code for.

Programmers Again!

10: Programmers are the problem

20: You are a programmer

30: GOTO 10

Issue # 10 Leaving admin info

Mainly applies to web apps.

Leaving admin info systems on the server to be accessed

You can use Google to find this info

You can find password files, office data files (PST) etc

Old files are possible especially you rename in the same directory. Then possible to download source code from your site.

Sample: `intitle:index.of outlook pst`

Also leaving trace output with `<trace enabled="true" and localOnly="false"> ..`

Allows access to `trace.axd`

Issue # 9 Passwords in plain text

Following issue #10

Username and passwords should be encrypted.

Sensitive data should be in encrypted

Don't write your own Crypto protocols.

Can also use google code to find these (especially if you leave personal ones there!!!)

<http://google.com/codesearch?hl=en&lr=&q=sa+connectionstring+file%3Aweb.config&btn=Search>

Issue # 8 Not patching

One of the easiest ways to get caught

Vulnerability is not in your code but on the system

Especially painful on web servers

Google can be used to find vulnerable web servers

Requires you most of the time to pester the local sys admin

Issue # 7 Client side validation

Shouldn't be the only thing that sanitizes your input

Consider you have a javascript function to see if the number is valid

User views source page and sends you the variables

Do validation on both sides to be sure, but definitely server side at least.

Validation best practices

Validate all inputs at the server even if client validated

Use a central validation source

Use white lists rather than blacklists

Escape special characters

Validate against RFC rules

Validate XML against the schema

Issue # 6 Error messages

You should never show a detailed error message on a production web site.

Use CustomErrors in the web.config

Either RemoteOnly or On

Again also turn off Trace and set Debug="false"

Issue # 5 – Incorrect Permissions

SQL connection using SA or SysAdm level permissions

Requiring Administrator permissions on the web server!!!!

Requiring Admin privileges for a windows app

Issue # 4 – Directory Traversal

Consider `default.aspx?download=filestore/file.exe` using `BinaryWrite`

Change the download variable

Now `default.aspx?download=web.config`

Page will display the incorrect file and give ideas about what way the machine is configured and possibly access to a lot more.

How to prevent it

Validate your input

- Checking for ../ usually wont work due to URLEncode
- Strong checking of input

Placing web apps on separate partitions to system files

Correct permissions

Web server fully patched

Using scanner tools to validate the web server

- IIS Lockdown
- URL Scan

Issue # 3 XSS – Cross Site Scripting

HTML & Script Injection

3 Main types

- DOM
- Non Persistent
- Persistent

Non persistent is the most common, and persistent is the most dangerous.

Certain CMS are vuln, as well as pages taking input and displaying that input back.

Other variations include HTTP response splitting, HTTP header injection, remote file inclusion

Remote File Inclusion

Particularly nasty

More common with scripting languages such as ASP and PHP

Allows you to insert your own file to be run

Not as relevant to .NET but still can cause a problem

Example

<http://server/file.aspx?redir=page.aspx>

<http://server/file.aspx?redir=http://badplace/haha.aspx?>

Imagine that with a login and similar look of your own site

Mind your cookies!

Make cookies only accessible to server side code

```
<httpCookies httpOnlyCookies="true">
```

Use cookie based session state to stop session hijacking

```
<sessionState cookieless="UseCookies">
```

Where possible use SSL for authentication cookies

Use unique forms name when using multiple sites with forms auth.

How to avoid

Use `HtmlEncode` to disable special chars

Make sure on redirect its only going to where you expect it to be going

Sanitize your input

Mind your cookies and evaluate `web.configs` above the web app for vulns

Issue # 2 SQL Injection

Allowing straight input to your database

Consider

- `SELECT * FROM tbl WHERE (Email='RequestData') AND (PASSWORD='OtherData')`

Now consider the inputs `" ` OR '1'='1' "`

`SELECT * FROM tbl WHERE (Email=" ` OR '1'='1' ") AND (PASSWORD=" ` OR '1'='1' ")`

Worse

- `UPDATE tbl WHERE ID=RequestData`
- `RequestData = 1;DELETE FROM tbl;`

Worst!

- `RequestData = 1;DROP tbl;`

How to avoid

Sanitize your input

Dont blindly allow access to the database from the front end

Use only the permissions required for the option

Consider two level database access

- Reader
- Writer

With SQL Server reduce your permissions to execute only if you are using stored procs

Issue # 1 – Being Trusting!

Trusting your users!!!

Sanitize your input

If you don't check it, be prepared to deal with the consequences. See issues 2 & 3!

Famous examples: Amazon & Komplet

Resources

- <http://www.security.nnov.ru/>
- <http://www.devx.com/dotnet/Article/32493/1763/page/1>
- <http://cwe.mitre.org/top25/#Brief>
- <http://msdn.microsoft.com/en-us/library/ms998274.aspx>
- <http://msdn.microsoft.com/en-us/library/ms998271.aspx>
- <http://www.microsoft.com/downloadS/details.aspx?familyid=58A7C46E-A599-4FCB-9AB4-A4334146B6BA&displaylang=en>
- <http://msdn.microsoft.com/en-us/security/aa973814.aspx>

Contact Information

Blog: <http://www.certsandprogs.com>

Twitter: @nmerrigan

Email: through the blog