Top .NET Secure Coding Practices for a Team

Developing a software and web application is not a one man (developer) job; there is a team behind the success and failure of any product (software/application). The team is also responsible to develop secure software or vulnerable software. So, the developer is not the only one to blame, the developer's job is to write the code and create the program, but we can't ignore the other roles associated with this development. The general roles are:

The team work

- Assign a security advisor or make someone responsible to check the implementation of secure software development life cycle.
- Privacy of the product is very important; the project manager should implement the policies to make sure the privacy of the project.
- Define the security benchmark at start and direct the team members to achieve the established criteria.
- Create the security plan at the designing phase of the product; clearly define the roles and work process. In addition, allocate the required resources.
- Create the Final Security Review (FSR) outline, what to test, and who is responsible for the final test. Strictly follow the rules and do not let the software go live without the final approval of the security advisor or any role assigned for this task.
- Perform security risk assessment (SRA) at start to identify the core areas that might require the security review. Moreover, follow the guidelines of the assessment result during the development stage.
- Training plays a crucial role; every individual should understand the need of security. Introduce training & development sessions to have everyone on board, train them to maintain the confidentiality, integrity and availability of the software, data and functions.

- Get someone not associated with the team to test the security, this way you can minimize the biasness risk.

There are many standard work procedures have been created to counter the security issues; professionals have developed different framework, methodologies, tools, techniques, training and the vulnerability management systems. However, the ideal situation is to create/develop secure software/application that does not have vulnerabilities and bugs. Although, in real life scenario, software development goes through different phases and life cycle that introduces the loopholes in it.

This article discusses the techniques and secure coding practices to eliminate the coding errors and loopholes in the development stage. The smart developer should not let his/her software and web application go live without performing the necessary test to find the loopholes.

.Net framework is the renowned software development framework by Microsoft, Frame Class Library (FCL), which is the part of .NET framework, provides language interoperability and there are many other advantages to create a program by using .Net framework. In addition, implementing the secure coding practice is not the choice, but the requirement of S-SDLC (secure software development life cycle).

Every role is important and the team should follow the fundamental security practices such as:

Carefully Design the Class

OOP or object-oriented programming principle applies here; you can reduce or completely kill the assembly level attack by designing the secure classes. The Class represents the numbers & fields, while designing means the members (private and public). You should:

Reduce the member and class visibility:

The objective can be achieved by minimizing the entry points (public interfaces) in your code. Emphasized the private access modifier, protected access modifier can also be introduced if the members have to access the derived classes.

Code access security

The objective is to restrict the access to your code. Every method in a class is not for the general public use, you should restrict the access to your code where required. **Strong naming** is the right practice because it restricts the access of non-trusted or partially trusted callers. Alternatively, code access security permission should be imposed, so whenever the restrict caller tries to connect, it needs to have a security permit.

Input is not trusted

The common attacking methods (SQL-injection, XSS and other injection attacks) exploit the application by inserting their code. The practice is not to trust the input. Input validation is the practice to implement, validate the input by checking the range, format, length and type by using Regex Class and Regular Expression Validator Class. For example:

```
Regex reg = new Regex(@"^[a-zA-Z'.]{1,40}$");
```

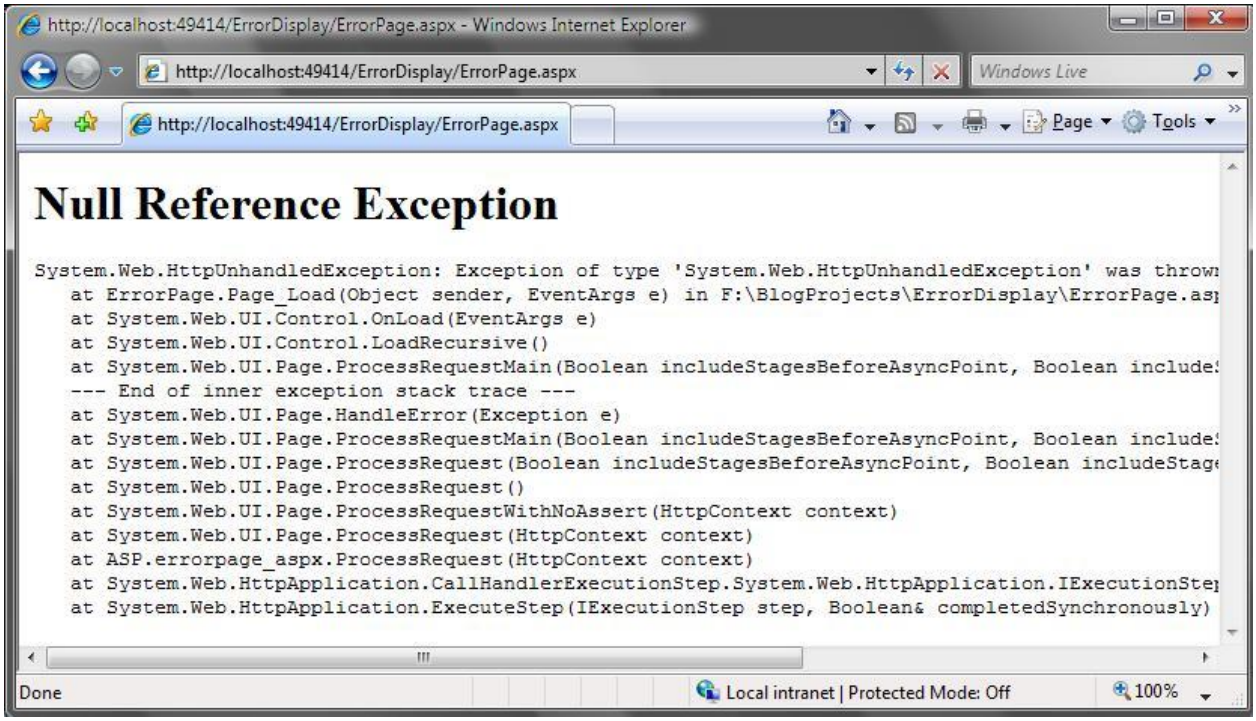
```
Response.Write(reg.IsMatch(txtName.Text));
```

Where, regex value above represents the accepted characters and range of the value.

Exception Management

Exception is the notification that appears if something not defined/programmed occur and stop the normal function of the program. It shows an unexpected event, this unusual situation or error can reveal some important information about the program, which should not pass to the client/user screen. Your software/application should be programmed in a manner that does not reveal the information of the system, software, version etc.

Exception can be used to achieve many objectives; for example, separating the codes (error and regular), creating groups of error and to not print back the internal information. The exception message sometime gives enough information to the user to plan an attack.



(Image source)

Protect system & application information

The exception message may include the OS details, SQL command, .Net framework version details, connection strings and other useful information that an attacker might use to exploit the system. The solution is to return the generic error message instead of giving all the details to the caller.

You should adopt an exception management system to improve your software/application. Microsoft also provides the complete guidelines and practices for exception management in .Net environment.

Data Access

Do not store connection string in your assembly

If your code has to communicate with the database, then do not hard code connection string in your assembly because an attacker can retrieve the information from the assembly. You can store the information in the configuration files that are not accessible.

Use parameterized SQL to prevent SQL-injection

Use "SqlParameterCollection" to validate the input and check for the type, length and values. Utilize the exception management techniques to handle the errors; the objective is not to publish the error on the user screen.

You should use the least privileged account to access the database, restrict the access of every individual user. This way, you can minimize the impact if database is compromised.

Communication & Encryption Management

Implement Transport level security (TLS) to counter sniffing, man-in-the-middle and tampering attack. System.Net.Security is the managed class available in .Net framework that provides network streams for secure communication between hosts. SSL (secure socket layer) should be implemented on both clients and user side via System.Net.Security.

Use XMLEncryption mechanism to encrypt the files with different keys and use HMACSHA1 to check and maintain the integrity of the data.

Encryption key size represents the security of a particular data or mode of communication, using strong algorithm and large key is not a choice, but a necessity. Use 2048-bit key for asymmetric algorithm (RSA) and use 128-bit key for symmetric algorithm (AES). Utilize GenerateKey method to generate randomly the symmetric algorithm key.

Use a strong algorithm such as SHA256 to store and protect the passwords. The passwords should be stored in a non-reversible hashed format; a random salt value can be generated by using the following string:

```
byte[] salt = new byte[32];

System.Security.Cryptography.RNGCryptoServiceProvider.Create().Ge
tBytes(salt);
```

General Guidelines

The secure coding practice strongly suggests that the developer should only use the approved tools, secure language subset and coding standard should maintain. **Static code analysis** has to be done here to find the coding errors and issues. Some common tools for .NET platform are:

- FxCop

- CAT.NET
- OWASP code crawler
- NDepend
- Parasoft

Static code analysis has its own advantages; for example, bugs can be detected in the development cycle. However, static code analysis is not the only one solution and it is not enough because you need to check your code in the run time environment and dynamic analysis is the solution for this. You should go ahead and do dynamic code analysis too. Dynamic code analysis is an act or methodology to test and evaluate the program by performing it. This step is essential because the developer has actually to perform the created software and how it is going to interact with the database like SQL, and how this created application will communicate with the web server. Dynamic analysis looks at the vast perspective and it is supposed to find the common vulnerability such as SQL-injection, it performs the input and output validation to test the validity. You should adopt both (static and dynamic) strategies; both strategies have their own merit and demerits. The performance of automated tools depends on how it programmed to work, and it gives false positive and negative response, meanwhile static analysis is a time-consuming process, but it can easily overcome the issue that dynamic code analysis has. Some tools for .NET platform are:

- dotTrace (.NET profiler)
- ANTS Performance profiler
- CLR profiler
- Yourkit .NET profiler

After developing the program, perform QA analysis. **Fuzz testing** and attack surface analysis should be done at this stage. Fuzz testing is the method to test the application by inserting the random data at input to understand the behavior of the software. You can find the errors, loopholes and the defects in your code by performing fuzzing. On the other hand, the **Attack surface analysis** is an act to check the entrants from where the attacker could get into the system. The following points should be checked:

HTTP headers & cookies	APIs	Database
Email and other communication	Login / authentication	Admin panel

Attack surface analyzer is created by Trustworthy computing security group and it is recommended by Microsoft. The tool is created to track the changes appear in attack surface, you can analyze the changes occur on firewall rules, file permission, registry and more.

Conclusion:

The discussed technique, if implemented properly can reduce the chances of error and any vulnerability. The few mistakes that happen during the .NET development cycle are:

- Focuses on design rather than logic
- The known vulnerabilities introduced due to the poor coding practice and taking the codes from non-verified online sources.

- Not identifying the security requirement at the start of the development

The risk can be eliminated by introducing training and development session for teams and to identify clearly the roles and associated risk.