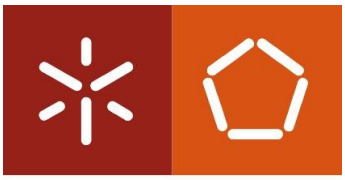

University of Minho

Informatics Department

Security Engineering
Project 1 - UC Secure Software Development Standard

Group 14
Karim Kousa E9590
Márcio Sousa A82400
Mário Sequeira PG39293

March 24, 2020

Contents

1 Introduction

Secure software development is a very important part of cyber security, as this is the first basis on guaranteeing the security of any software. Incorrectly implementing some part of the said software opens a possibility for exploitation that eventually some attacker will find and make use of. This problem is so common that almost every penetration testing report contains a chapter on exploiting "application weaknesses".

By establishing secure software development standards and respecting them, we will have as a final result a program in whose development stage special security related measures were taken in order to make the application as safe as possible, and by developing the project based on this security based architecture we will be preventing future problems that would most probably occur when attackers found the security breaches.

With this in mind we present here the Secure Software Development Standards from University of California, where we will provide an overview of the proposed security measurements on different parts of the development stage.

2 Requirements for Secure Software Development

Here is where we will provide an overview over the UC Secure Software Development Standards, and therefore we will list the topics we think are the most important.

2.1 Software Development Process

Secure software development includes integrating security in different phases of the software development lifecycle. And to do so, a few security elements must be introduced:

- Planning to meet security requirements and goals.
- Threat Modeling.
- Design to inlcude security and privacy concerns.
- The whole system architecture in order to be the less vulnerable possible.

When talking about code, especially code that deals with highly important information, there are also some work to do in order to prevent future exploitation:

- It is important to perform regular code reviews in order to reduce cyber risk by, for example, checking for common security mistakes.
- The review process should include someone with specific security experience, which will provide some extra safety considering the experience.
- As always, testing the code is extremely important, and to do so more efficiently, code testing tools should be used, like Jasmine for example, which is aimed at Javascript.

2.2 Input Validation

It is well known by now that if the user is able to provide input to the program or the website then most certainly someone not so well-intended will attempt to use that to push through with an attack, as we see with the good example of SQL Injection attacks. Therefore, if the software must take input from the user and not being possible to know if the user is well or not-so-well-intended, we must take precautions, which include but are not limited to:

- Validate/Sanitize user input before using it programmatically.
- Protect against buffer overflow attacks.

- Put all SQL code and commands in Server-sided code, and do not use concatenated database queries.
- Protect against URL query string manipulation attacks.

It also very important to now disclose any sensitive information in the URL/URI, like credentials, access tokens, serial numbers or record numbers.

2.3 Exception and Error Handling

It is of great importance to take care with Exceptions and Errors because, when incorrectly handled, the Exceptions/erros can be outputed to the screen or the browser, revealing information. So, the developer must take some precautions, such as:

- Configure runtime environments so the Exception/error messages are not outputed to the screen/browser. Set up custom error pages for the framework errors.
- All errors must be handled in the code. There should not be any empty catch blocks, all Exceptions must be dealt with to either stop the running of the program and exit safely, or appropriately handle the Exception.

By using these, we avoid the possibility of the attacker getting any useful information by forcing errors.

2.4 Insecure Direct Object References

When handling direct object references to any object, file, directory or database key, the developers must guarantee that these include an access control check or other protection.

2.5 Logging

In the log file topic, it is important to make sure that we log important events like authentication for example, but also assure that these don't disclosure any sensitive information in the details, and so it is recommended to:

- Log information related to important events like authentication, use of privileged functions, administration activities.
- Limit information in the log details, especially sensitive information.
- Ensure there is enough available storage to save all the eventual logs.

2.6 TLS and Secure APIs

The software developers involved in the project must make sure to:

- Force HTTPS and disable HTTP for all browser connections for safety purposes.
- Use the most recent version of TLS to protect machine-to-machine connections, for example the communication between the app server and the database server.
- Use strict transport security header.
- Encrypt Protection level 3 or higher using a separate and unique key.
- Authenticate and encrypt APIs, extract tools and service bus sessions.

No developer should write their own encryption as it is too easy to make errors, and should therefore use what already exists and is proven to work correctly and most efficiently.

2.7 Credentials/Passphrases

In order to maintain the security of a software, the IT workforce members must follow some rules and guidelines regarding Credentials and Passphrases, so that the software developed is as secure as possible. So developer should:

- Never store user passphrases.
- Protect service account credentials with established tools and techniques and use secure protocols for credential or other secret exchanges(e.g.,TLS 1.2 or later). They should also never hardcode credentials.
- Implement lockout after 5 failed authentication attempts, so that the risk of a dictionary attack is minimized.

2.8 Session and Logout

In terms of Sessions and Logouts, developers must be aware of user interactions, so that they cover as many possibilities of problems and risk as possible. If there is a session on without the proper authorization, it could lead to a breach in the software confidentiality. So IT Workforce members must:

- Ensure that session timeout is implemented.
- Not only making sure session tokens are random and long (GUID long), but also change (delete old, invalidate and create new) tokens at each logon and logout and each privilege escalation.
- Use TLS 1.2 or later for all session tokens and cookies.
- Provide a logout function. They should prominently display the logout function throughout the application so the user as an option and is aware of it.
- Properly close records, delete temporary objects and close operations on the server: As part of the logout process, after a set period of inactivity (e.g., less than one hour) or after a browser window close.

2.9 Federated Authentication / SAML / Shibboleth

Developers must use the Location-approved, single sign-on, SAML-based authentication or the CISO-approved method for authenticating users.

This means that the authentication process is secure. On improving user authentication is also helpful to:

- Assign permissions to user groups (not individual users)and assign users to user groups.
- Ensure the application has a user group with no/minimal privileges assigned and assign or default new users to said group. Also make assignment to a group explicit.
- Check/recheck authorization at the next logical operation (e.g., for web applications at the next page request).
- Log all actions that grant users or groups permissions or rights.

2.10 File Management

In this part of software development, IT Workforce Members must take on account the tries to elevate privileges by attackers.

So they have to ensure that the files of the software are blocked to users with the proper authentication. To accomplish that they must:

- Prevent/disable directory traversal/directory listing.
- Filter web requests to block access to files with non-approved extensions (e.g., .dll, .config, .java, .cs, etc.).
- For IT Resources with Availability Level 3 or higher, set a directory quota or similar control.
- Ensure shared volumes/file shares have appropriate access restrictions limited to the application service account.

2.11 Secure Configuration

In the topic of securing the software and its connections, is crucial that we specify how the connections are made, its protocols and which components can use it. It is also relevant to configure the privileges of each component. So developers must:

- Use secure configuration options for supporting technology, libraries, packages and tools.
- Perform or have performed a vulnerability scan of the entire solution and appropriately re-mediate findings based on risk before placing the solution into production.
- Secure the components used in an application and run them with the least privilege possible.
- Keep components patched and up-to-date.
- Execute SQL with the least privilege possible.
- Use vetted and tested hardening guides to ensure the correct application of options and settings.
- Define, implement and maintain secure settings (e.g., do not rely on default settings).
- Secure the software configuration used in an application.
- Set the configuration to define which HTTPS methods (e.g., Get or Post) the application will support and whether HTTPS methods will be handled differently in different pages of the application. Also set Header and Content-Security-Policy when possible.

2.12 Documentation

To better protect an application, software developers must create documentation in order to other developers and evaluating agencies can better understand how it works and the measure implemented in terms of security.

Developers working on software with Protection Level 3 or higher or Availability Level 3 or higher must create project documentation to:

- Record the security features and steps taken to protect: Confidentiality concerns. Integrity concerns. Availability concerns.
- List components used, the security state of the components, to track defects and its fixes.
- Note/record testing processes and how testing processes manage cyber risk.

2.13 Version Control

Version control is a key point while developing a software, as it allows developers to have full control on which features to implement, not to implement, or to know which software is in the testing phase and which version is ready to be launched or deployed.

Developers working on software with Protection Level 3 or higher or Availability Level 3 or higher must:

- Use a software version control system or repository and configure it to to prevent and detect unauthorized changes.
- Deploy test and production applications from the version control system repository.
- Tag/label versions so that they can be extracted at a later time.

3 Conclusion

There are many different precautions to take when developing software to be the safest possible, and it is important to note that although many are presented here, this is just an overview and therefore are many more ways in which one could improve the software being developed. Even though some fault may pass overlooked, by respecting these standards we ensure that our websites, applications and software in general, will be much harder to attack, providing a better experience to every client and avoid a great part of future problems that could occur.