Developing an online banking application

Finance, Banks

This report details the importance of securely developing a software and the best practices to implement throughout the development lifecycle. Using the Microsoft Secure Development Lifecycle Model, a software can be developed with sufficient security measures throughout each stage from the beginning of development until it’s eventual release and even responding to incidents that may follow it’s release.

Creating an online banking application without thoroughly considering the security of the bank’s assets and customer’s information would be virtually impossible. Due to the vital importance of the assets a bank contains, large security measures while developing any aspect of its services must always be implemented. Developing this online banking application must include various steps as can be seen in the Microsoft Security Development Lifecycle (Such as Security Requirements, Risk Assessment and Threat Modelling).

Banks and financial businesses are large targets for malicious attackers who target the online services provided by these companies. It is for this reason that the threats posed to a bank with an online banking service are vast and development of such an application should be treated as such.

Considering the OWASP Top 10 is a good initial security measure as mitigating the threats of the top 10 most common vulnerabilities found in web applications will give a good foundation in avoiding attacks.

The application works by having the user access the website through their browser, navigating through the two step authentication and then gaining access to various options relating to their account such as viewing

statements, transferring money to other accounts and viewing the amount currently in their account.

The first of the two step verification is an 8 digit pin that the user will have decided upon earlier when first creating their account for their online banking service. The second step verification will either be the user's date of birth or occasionally it will be the user's contact number. This second step verification will change randomly so as to avoid use of an automated tool attempting to access a user's account.

When the user creates an online banking account, they will be required to give their home address and account number. A letter will then be sent to the user giving them a code that is specific to them which they can then use to verify their identity on their first use of the online banking application and complete creating their account. This means that the only people who can use the service are those who already have full access to the user's account details and their post. This is an effective security measure as implementing security into a software that can be compromised simply by having any person impersonate another user signing up for the service would be redundant.

Another way that the login process will be secured is by using a counter in which if a user enters details incorrectly three consecutive times then they will be unable to make another attempt for a short period of time.

The reason behind this two step verification process is to hinder the use of tools that would continuously attempt to crack the login system, possibly

with the use of a tool such as John the Ripper or THC Hydra. The limited amount of login attempts is also used to avoid brute-force attacks from occurring.

Having already been authenticated, a user will then have access to their account details including their balance, their previous statements and also they will be able to transfer funds from their account. All of this information will be stored in a database which will be encrypted and salted meaning that a leak of this information should not cause for the information to be decipherable by an attacker.

The Secure SDL (Software Development Lifecycle) as implemented by Microsoft is a development process which assists developers in creating secure software and looks at complying with security requirements whilst reducing the overall development cost.

The Lifecycle is separated into 7 different SDL practices as can be seen in the figure below. These practices are used to highlight security implementations in the various stages of a software’s development. For example, in the designing of a developing software, it is necessary to create accurate threat models which can be used to easily locate different possible vulnerabilities that the software may be subject to.

(stan. gr, 2012).

(Microsoft, 2016).

Establishing Security Requirements

One of the first steps to be taken in developing the banking software is to establish what security and privacy requirements will be implemented in the software. This will make it easier to identify the direction of the development and assist in keeping to the schedule. The team developing the banking software will primarily look at the OWASP Top 10 as the main vulnerabilities that may occur in the application and attempt to secure against these.

One of the security requirements that will be present in the software is to secure the software against Injection. As the information that is shown when a user logs in is sensitive, the software must protect against malicious users attempting to login by using injection. In order to avoid SQL injection, the software will be developed using prepared statements in order to sanitise the input of the user.

Validation methods will be included in the software to ensure that each user has the correct authority to use the functions that they attempt to use and that all inputs that are entered into the application will be acceptable so as to avoid cross site scripting and other such threats.

Create Quality Gates / Bug Bars

In the early stages of development, deciding what the minimum acceptable level of quality should be present in the security of the software is vital. Without this step, oversights may exist such as user's private information not being totally secure as the development team did not focus on protecting this over a different area.

Having a minimum acceptance level also helps the development team to correct security bugs as they are to follow the standard set and will be given some concept as to what risks are associated with various issues.

For this software, it will not be acceptable that any bug that could be related to the leaking of information may be present. Strict security measures will be put in place to ensure that the privacy of the bank’s customers will be protected.

Security & Privacy Risk Assessment

This stage of the development will involve examining the software design and locating areas that are potentially prone to more threats or perhaps possess more risks than other areas. For example, the database being protected, as it contains vital information, is of higher risk of a malicious attack than the website hosting the application. Identifying these risks and what they are susceptible to will improve the security of the software. This will be further developed in the threat modelling step as this step determines which parts of the project will require threat modelling.

This stage is vital in the development process as the likelihood of protecting against a risk that has been overlooked in the development of the software is far less than if it had been analyzed throughout the development.

Design

(Microsoft, 2016).

Establish Design Requirements

Establishing the Design Requirements will ensure that the software will function in the intended way while also allowing to minimise cost and improve security throughout the development. This stage will guarantee that the software will be user friendly and will also assist in ensuring that there is no way that a user may accidentally gain access to information that they are not authorised to do so.

Analyze Attack Surface

This step involves analyzing which parts of the software presents opportunities for attackers and can assist developers in reducing these vulnerabilities. This may involve disabling or restricting certain access to services. This stage is another stage that will be a large part of the threat modeling stage in that it will allow the developers to identify aspects of the software that are viable to be attack targets.

Threat Modeling

This step will allow the developers to look at exactly what happens when a user is using the service and to anticipate what aspects are vulnerable to threats. From here, developers can decide the feasibility of reducing these threats and how this may be achieved. This can be done by identifying vulnerable areas and ensuring that they are secured against the attacks that they are susceptible to. The importance of this stage is highlighted by the importance of protecting the sensitive information that the application will be using.

The figure below shows a threat model created with the *' Microsoft Threat Modelling Tool 2016'* in regards to the online banking service.

(Microsoft, 2016).

Use Approved Tools

Using approved tools throughout the development process will assist in ensuring that correct security procedures will be used in the software. This includes using a compiler which will flag security warnings if the software is being compiled and contains a known security risk. These tools may include the IDE (Integrated Development Environment) for the developers to programme the software on, such as Eclipse.

Deprecate Unsafe Functions

Banning functions that are deemed to be unsafe will reduce potential bugs in the software. Detecting these can be done by using automated tools or manually checking the code and ensuring that none of the functions are present on the banned list which can be found at *< https://msdn. microsoft. com/en-us/library/bb288454. aspx >*.

Static Analysis

Analyzing the source code before compiling it is a good way of ensuring that the code has been developed in a secure manner. This stage will involve the developers to look at the code and check that the correct security protocols have been put in place such as prepared statements and sanitisation of inputs.

(Microsoft, 2016).

This stage of the Software Development Lifecycle involves testing the software to ensure that the software is functioning as it is intended and also allows for web application penetration testing to be carried out in order to confirm that the security functions put in place are working correctly. This penetration testing can be done by the business if they have their own department or it can be outsourced to an outside specialist company such as Offensive Security.

Offensive Security offers “ more accurately simulate real-world hacking situations to audit network, web, and application security programs” (Offensive Security, 2016).

(Microsoft, 2016).

Perform Dynamic Analysis

Using various tools to monitor things such as user privilege issues will assist in verifying how secure the software is when being used. It is at this stage that the software can be looked at for any possible security oversights. This stage is similar to the testing stage and can be used to verify what devices the web application works on and also if there are any errors with how to application performs. An example of this would be that the application may work as intended on a Firefox browser from an android device but may not work entirely as intended on Safari on an iOS device.

Fuzz Testing

This step involves attempting to make the program fail by introducing random data. This testing is used to verify how the software handles errors and if there is any weakness in the security of how the software does this. This may involve an error occurring which gives sensitive data about the software's database. This testing will ensure that the sanitisation of the user input's is working correctly by handling these errors rather than executing code that is input.

Attack Surface Review

Reviewing the attack surface when the code has been completed will help ensure that any future changes to the design or functionality of the software has been considered and that these changes will not compromise the security of the software. An example of this could be that considering making the web application into a mobile device application may present difficulties as different vulnerabilities may be present.

(Microsoft, 2016).

Create an Incident Response Plan

Creating an Incident Response Plan is crucial in order to combat any threats that may appear over the software's lifecycle. It involves identifying security emergency contacts in the event that a security breach occurs. The incident response plan can be broken down into six phases:

Preparation

Detection

Containment

Investigation

Remediation

Recovery

The Preparation phase involves having implemented the correct controls in order to recover following an incident. It states the policies, tools and contact information that is necessary in order to respond efficiently to an incident.

Detection is a phase which involves the discovery of the incident. This can be through use of logging or may come in the form of a consumer alerting the business. In this phase, the incident will be declared and the severity of it will be determined.

The containment phase will be where the affected part of the software will be isolated or mitigated if possible. If the incident affects the software in it’s entirety, it must be determined whether or not the entire software is to be taken offline so as to avoid any more users to be affected by it.

The investigation phase will involve looking at the incident and attempting to identify the source, the scope and the priority of the incident.

The remediation phase will be where it is decided which parties to inform about the incident and will confirm that the threat has in fact been contained.

The recovery phase will be the phase in which it is determined how the software will ensure that the incident does not happen again and will confirm whether it is necessary to review any of the software's policies. (Raderman, L. 2015)

Conduct Final Security Review

Reviewing all of the security checks and measures prior, throughout and post release of the software helps to ensure that they were carried out correctly and that none had been left out. This step can be assisted by using an automated tool such as Vega to scan the application and determine if any known vulnerabilities have been overlooked.

Ensuring that the utmost has been done to protect the security and privacy of it's users should be one of the bank's largest priorities in developing this software as without the trust provided by this, the bank will surely suffer with a loss of assets in the form of customers and finances.

Certify Release and Archive

Certifying the software before it is released will help to ensure that all of the correct security requirements were met. Archiving the data will allow the developers to do roll backs and to review any future security or privacy breaches in relation to the original software. Without certifying the software upon it's full release, the credibility of the software may be questioned and it may cause negative public relations for the business. As a bank, it is important that customers are confident in the security and privacy provided by the business.

(Microsoft, 2016).

Execute Incident Response Plan

The capability of implementing the Incident Response Plan from the Release step will assist in helping users to avoid severe security and privacy breaches and allow for the company to have a quicker response to any exploits that may arise. This step is important as users should feel confident that the bank has their best interests in mind and will ensure that their security, being one of the business' key assets, is being frequently and effectively protected.

Developing an online application for a bank could prove extremely beneficial and convenient for it's customers. However, the importance of the information that a bank retains in regards to it's customers and their finances is high and with the nature of cyber security and its ever increasing attacks, especially to a high profile target such as a bank, the development of such an application should be assessed with security in mind throughout the process. Following the Microsoft Secure Development Lifecycle is a very effective way of ensuring that a software is thoroughly analyzed for security threats and vulnerabilities and ensures that a business will have reasonable plans in place in the event that any breach of security may happen. It is also beneficial when developing a software to be secure, to refer to the OWASP Top 10 vulnerabilities and ensure that the software is as secure against these vulnerabilities as possible.

Microsoft (2011) Security Development Lifecycle (SDL) Banned Function Calls [online] available from: A https://msdn. microsoft. com/en-us/library/bb288454. aspx [accessed 27th December 2016].

Microsoft (2016) What is the Security Development Lifecycle? [online]

available from: https://www. microsoft. com/en-us/sdl/ [accessed 27th December 2016].

Offensive Security (2016) Advanced Penetration Testing Services [online]

available from:

https://www. offensive-security. com/offensive-security-solutions/penetration-testing-services/ [accessed 2nd January 2017].

OWASP. org (2015) Top 10 2013-Top 10 [online]

available from: https://www. owasp. org/index. php/Top_10_2013-Top_10 [accessed 27th December 2016].

Raderman, L. (2015) Computer Security Incident Response Plan. Carnegie Mellon Information Security Office[online], 13th Febuary 2015, (pg 8-9),

available from:

[accessed 2nd January 2017].

The phases of Microsoft SDL.(2012) [online image] available from:

,[accessed 27th December 2016].