Front Page

Contents

List of Tables

1 Introduction

Software development has evolved from something that is done in your garage after work, to the backbone of society. This growth has spawned numerous companies, and in turn increased the need for structure, planning and working methods that can provide results, regardless of project size. With the ever increasing market for computers and PCs in general, more and more people with malicious intents learned ways to steal and obtain information of high value. Hacking, creating computer worms and abusing loopholes became an ever increasing threat, in a market that seemed to grow endless.

Many companies started up businesses revolving solely around security. Norton [5], McAfee [4] & Avast [1] are some examples of companies that have created various softwares for security. Mostly virus scanning programs and firewall applications. But security related issues still haven't reached a significant part of the planning process of software development. Instead of repairing damages after an attack, why not simply prevent them from happening in the first place?

In this paper, we'll talk about a proposed software, designed to help plan development with security in mind. How can we assure that our users are secure using our application. What precautions do we need to take, and what issues are more critical than others?

2 Background

2.1 State Of The Art

As is, there are no applications out there that can analyse a software specification automatically. Most services are designed around practices while developing, like risk analysis or thread modelling. All which has to be done manually. The Microsoft Security Development Lifecycle (SDL) Process is a multi-step process for creating secure applications using both risk analysis and thread modelling to map security related issues. [10] One of the steps in this 7 step process is "Perform Dynamic Analysis", which includes running a software that monitors an application and checks for critical security problems. This step is performed with an already running version of an application, meaning it is of no use to someone who's still in the planning phase of software development.

Other possibilities for security assessment are hiring consultants that would either help with developing, or install their customized applications after development has been completed. [11] This obviously comes with a price, and hiring several would be out of reach for smaller companies or for sole proprietorship. A single consultant would cost around $200 and upwards. [12, 13, 14]

There's a clear need for a solution that is designed to handle security issues earlier in the development process, performs analysis automatically and comes with a price that would make it available to home developers as well as large companies.

3 Thesis Statement

How can we best design an application, hereby referred to as SecuritySoftware, to solve the needs we've highlighted so far? In short, our application has to answer the statement: *how can we create an application that would highlight security issues early in the planning phase, and integrate this as good as possible with already existing specification related methods.*

We will discuss in the success criterias how to implement this in a way that would let our application become an asset to the planning phase of software-related development projects.

3.1 Research Questions

We define the research questions for our thesis. We will use these questions in our thesis as a guideline for the direction of the project.

> *"How can security-issues related to software become a core part of the planning phase during development?"*
>
> *"How can we best implement an application to fit the current planning processes relevant to software- and specification-planning?"*

3.2 Success Criteria

3.2.1 Goal Statement

The goal for SecuritySoftware is to provide developers with a tool to help plan for features that are secure against external threats like attacks through security holes, viruses and poor programming solutions (an example would be using Statements [7] instead of PreparedStatement [6]).

3.2.2 General Criteria

In order to best realize our goal, we need to set some criteria for success; what do we define as a success, and how can we achieve it. For software developers, important functions for an application is responsiveness, cross-platform support and ease of use. Responsiveness is important, because when you're developing software, you do not want to be slowed down because the applications you are using is not working fluently, or keeping up with your speed. Cross-platform has become ever more important with the rise of Apples iMacs. More and more people use "Mac's" to develop software, but there's also a large userbase that works on Linux. Windows also have a large userbase, and supporting them is pretty much mandatory and this day and age.

3.2.3 SecuritySoftware Specific Criteria

For SecuritySoftware to be considered a success, the following criterias has to be met:

> *"The database of SecuritySoftware has to be easily updated for the software to stay relevant."*
>
> *"SecuritySoftware should require no former knowledge regarding security or programming."*

4 System breakdown

SecuritySoftwares two main components are the progam itself, and a database containing all the information regarding security issues and their corresponding solution (or lack thereof). The database will be located online, meaning SecuritySoftware will not be useful to someone without internet access. Possible feature to allow a user to download the database locally is a potential fix for this, but considering how often issues are discovered and how quick fixes for this can be discovered, this is a sub-par solution. What if a proposed solution already has been dated, and found not good enough? Users working offline would not be aware of this if the database is not updated regulary. In this day and age, it's almost impossible to develop software without an internet connection. Troubleshooting, version control and downloading dependencies are all done online. Version control is perhaps one of the most important aspects of development these days, as multiple people can work on the same files without having to worry about removing or changing their contributions. [8] GitHub, one of the largest websites related to development, have over 6 million people using their version control repositories. [9]

Use of SecuritySoftware will involve 4 different steps, each one dependant on the one before it (apart from step 1). The user will be guided through each step, and if needed, can jump back to a previous step to change, add or remove information.

4.1 Step By Step Explanation

Here's a rundown of each step encountered when running SecuritySoftware, and what is needed to complete each one. If needed, SecuritySoftware will allow backtracking to a previous step if changes has to be made. This will then change the final analysis.

4.1.1 Step 1 - Client Side Information

The first step a user of SecuritySoftware has to complete, is to list what kind of technology the client will be running to use the application that is going to be created. The reason for this is that each system has its own strengths and weaknesses when it comes to security, and certain flaws will only be present with a specific system. The user is presented by several checkboxes, each listing a system that they could choose to include in their analysis. Examples include iOS, Android, Web browsers on PCs.

4.1.2 Step 2 - Listing Features

The next step is perhaps the most crucial. For SecuritySoftware to be of any use, it needs to know what features the application will be providing. Without features, security wouldn't be an issue! The user is presented with a search box, along with a list of commonly used features. The search box will be updated while you type, meaning if the application offers a log in system, then it would show up while typing "auth" (short for authentication). Each feature will also be applied a "tag", meaning someone could search for "Log In" and they would be able to select "Authentication."

4.1.3 Step 3 - Server Side Information

The last step SecuritySoftware needs from the user, is information on how the server side will look like. Smaller systems might not need a database to handle authentication, and the issues would therefore be different. When adding a server to this step, SecuritySoftware will ask what this server will handle for each feature. If there's a database, and the application wants user credentials stored there, then the user of SecuritySoftware would list this after adding the database. This means that SecuritySoftware would know that the server database contains highly sensitive information, and it has to be stored safely in some way in case of a security breach.

4.1.4 Final Step - The Analysis

Until now, SecuritySoftware have not analysed anything. The first 3 steps have asked the user to list the information about how the system is constructed. The real meat of the application is when the analysis comes in.

SecuritySoftware relies on a database of information. This database is constructed with several tables. Each step has a seperate table, and there is a table containing issues and a table containing solutions. SecuritySoftware then creates an SQL request containing either a client, a feature, a server or 1 client and 1 feature. (An example would be (pseudocode): select from issue where client=client & feature=feature) Then it would fetch all corresponding solutions to this issue. After all clients have been analysed, it will do the same for every feature+server combination.

In short, SecuritySoftware will check all possible combinations, to see if there is a match in the "Issue" table with any of the 3 "primary tables". (Client, Feature, Server) If Client number 1 have an entry in the client-issue table, then that solution for that client would be given as a result. Several clients might have issues that are not connected to a specific feature, but just the client in general. This would then be picked up, because it would show up as an entry in the client-issue table.

Refer to Table 1 for a potential mockup of what SecuritySoftware's database could look like. Issues and solutions are seperate tables, because several issues could share the same solution, meaning the system would have an influx of redundant entires in its database.

Each column is a seperate table in the database.

Client		**Feature**		**Server**		**Issue**		**Solution**
iOS 7		Authentication		Database		SQL Injection		Prepared Statements

Table 1: SecuritySoftware Database Mockup

Version	**Additional Options**	**X For Yes**
iOS 7	Include mobile web-browsers?	**X**
iOS 6	Include mobile web-browsers?	**X**
Android 4.4 KitKat	Include mobile web-browsers?	**X**
PC Web Browsers		

Table 2: Step 1 - Client Information

4.2 Example Runthrough

The system is an information application to be used in hospitals, mainly targeting mobile operating systems, but also webbrowsers on PCs. It will support a chat system, letting you send instant messages to customer support if needed. This feature will require you to log in, meaning authentication has to be in place. This system is hereby reffered to as HospitalApplication.

4.2.1 Step 1 - Clients

HospitalApplication is a system that will be run on both Android systems, Apple systems and regular PCs.

The software will run on mobile devices, both cellphones and tablets/pads. In order to support Apples latest devices, iOS 7 is selected from the list. Most of the time, the latest version of a client is the correct choice, but in cases where it is not supported, the possibility of adding older versions are supported. This would be listed as a seperate entry in the client table.

Next up is Android. The latest version is 4.4.2 KitKat, and is selected from the list. This will allow the application to run on both cellphones and tablets supported by this version.

Both Apple and Android have mobile web browsers, and because the application is also accessable via web-browsers, SecuritySoftware needs to know this by checking an option that says “Include mobile web-browsers”.

Last, support for web-browsers running on a PC is added. See Table 2 for an overview of what this would look like.

4.2.2 Step 2 - Features

Next up, the user of SecuritySoftware needs to list the features HospitalApplication wants to support. As explained earlier, the ones supported for analysis all exist within SecuritySoftware. Popular choices show up in a list. Authentication is almost standard these days for mobile applications, so it shows up in the list of most used features. HospitalApplication also supports instant messaging and

Features	**Additional Information**	**X For Yes**
Authentication	Requires a username and a password	*Default*
Instant messaging	Require log in?	**X**
Chat log	Open for everyone to read?	**X**

Table 3: Step 2 - Feature Information

Server Information	Features	Description
Web Server	Authentication, Instant Messaging, Chat log	Request database
Database	Authentication, Instant Messaging, Chat log	Stores information

Table 4: Step 3 - Server Information

saving chat logs to a remote server. When the user adds Instant Messaging to the list of features, SecuritySoftware will ask if the feature requires a user to be logged in. HospitalApplication only allows logged in users to chat using the instant messaging service. The last feature is saving chat logs, and specifying wether or not this should be accessible for anyone to read.

List of features: Authentication (Log in, log out) via username and password, Instant Messaging, Chat logging of Instant Messaging. Please refer to table 2 for an overview of what this could look like.

4.2.3 Step 3 - Server

Step 3 shows how the client and server communicates for each feature. When a client wants to log in to HospitalApplication, he'll enter his credentials. This will then be sent to the server, which will ask the database if the given credentials match any of the entries found in the database.

When a client wants to use the instant messaging service, it will send a request and his message to the server, which will then be logged in the database. The server will then send the message to the correct recipient.

4.2.4 Final Step - Analysis

Now for the analysis, where SecuritySoftware finally does some work other than holding your own through the process. Please note that the issues and solutions highlighted in the analysis are for demonstrational purposes, and do not exist.

Clients: Step one is looking at the clients and see if there are any matches with the "Issue" table. There are no known issues with iOS 7 or iOS 6 right now, but Android 4.4 KitKat has a Java-issue where "ASyncTask" [2] can be intercepted by outsiders. Solution to this is to use a different means of communication when creating the Android application, namely the "Service" [3] class. There's no known security flaws with PC Web Browsers right now, so this means we've completed the client-issue step.

Client	Issue	Solution
iOS 7	No known issues	N/A
iOS 6	No known issues	N/A
Android 4.4 KitKat	Java: ASyncTask vulnerable	Use “Service” class instead.
PC Web Browsers	No known issues	N/A

Feature	Issue	Solution
Authentication	Vulnerable to stolen credentials	MD5-Crypt usernames and passwords
Instant messaging	No known issues	N/A
Chat log	No known issues	N/A

Table 5: Final Step - Client & feature analysis

Features: Next up, is checking the features. First feature is authentication. This is perhaps the most important issue revolving around security. Stolen identities and credentials is a big problem in this day and age, and whenever a solution for an issue is implemented, a new issue has already been discovered. SecuritySoftware's database is easily updated, and will therefore keep up with these changes as long as someone is maintaining it. For this simulation, we'll simply list the issue as “vulnerable to attacks” and list the solution as “md5-crypt on both usernames and passwords, so that in case of a leak it still has to be decrypted to be of any value.”

Instant messaging and chat logging has no known security issues for our simulation.

Servers: Because some systems will use databases to store information instead of directly on a server, there might be need of an extra layer of security in between the server and the database. This step in the analysis will provide that information, in correlation with the features. Chat logs will be open for everyone using HospitalApplication. It's therefore not needed to do anything special with these on our database, because all users can read it anyway.

Clients and features combined: SecuritySoftware have now completed the individual steps for client and feature, but the software also needs to check combinations of these. Maybe there are special conditions where authentication on iOS-devices are more vulnerable than Android? All combinations will therefore be checked.

When all these steps are completed, SecuritySoftware has finished analysing the application and the user can either change some of the information provided earlier, or add the results to a software specification.

5 Final Thoughts

SecuritySoftwares goal is to become a core part of the planning process of software development. The process has to be as smooth and easy as possible, while maintaining results that are relevant as the market changes. Problems can arise when SecuritySoftware tries to match clients with features or servers when trying to find issues. As development begins, these problems will be more clear and possible changes to how the analysis works are likely to happen. But in the end, the idea remains the same.

Additional features that might be relevant would be to include what programming language the system is to be implemented in, and increasing the information about the features a user of SecuritySoftware want to include. Should a specific feature require log in? Can anyone see it? Only accessable for certain users? These are all questions that are relevant to development, and will hopefully find it's way into SecuritySoftwares knowledge database.

In the end, SecuritySoftware will be an interesting project that will most likely serve as a prototype for future programs, or an idea to base new software upon.

References

[1] Avast.com AVAST 2014 | Last ned gratis antivirusprogramvare for virusbeskyttelse In-text: (Avast.com, 2014) Bibliography: Avast.com, (2014). AVAST 2014 | Last ned gratis antivirusprogramvare for virusbeskyttelse. [online] Available at: http://www.avast.com/no-no/ [Accessed 16 May. 2014].

[2] WebsiteAsyncTask | Android Developers Developer.android.com AsyncTask | Android Developers In-text: (Developer.android.com, 1918) Bibliography: Developer.android.com, (1918). AsyncTask | Android Developers. [online] Available at: http://developer.android.com/reference/android/os/AsyncTask.html [Accessed 16 May. 2014].

[3] WebsiteService (Java EE 6) Docs.oracle.com Service (Java EE 6) In-text: (Docs.oracle.com, 2009) Bibliography: Docs.oracle.com, (2009). Service (Java EE 6). [online] Available at: http://docs.oracle.com/javaee/6/api/javax/xml/ws/Service.html [Accessed 16 May. 2014].

[4] WebsiteMcAfee—Antivirus, Encryption, DLP, IPS, Firewall, Email Security, Web Security, SaaS, Risk & Compliance Solutions Mcafee.com McAfee—Antivirus, Encryption, DLP, IPS, Firewall, Email Security, Web Security, SaaS, Risk & Compliance Solutions In-text: (Mcafee.com, 2014) Bibliography: Mcafee.com, (2014). McAfee—Antivirus, Encryption, DLP, IPS, Firewall, Email Security, Web Security, SaaS, Risk & Compliance Solutions. [online] Available at: http://www.mcafee.com/no/ [Accessed 16 May. 2014].

[5] Website No.norton.com Norton: antivirus & anti spyware & backup In-text: (No.norton.com, 2014) Bibliography: No.norton.com, (2014).

Norton: antivirus & anti spyware & backup. [online] Available at: http://no.norton.com/ [Accessed 16 May. 2014].

[6] PreparedStatement (Java Platform SE 7) Docs.oracle.com PreparedStatement (Java Platform SE 7) In-text: (Docs.oracle.com, 1993) Bibliography: Docs.oracle.com, (1993). PreparedStatement (Java Platform SE 7). [online] Available at: http://docs.oracle.com/javase/7/docs/api/java/sql/PreparedStatement.html [Accessed 21 May. 2014].

[7] Docs.oracle.com Statement (Java Platform SE 7) In-text: (Docs.oracle.com, 1993) Bibliography: Docs.oracle.com, (1993). Statement (Java Platform SE 7). [online] Available at: http://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html [Accessed 21 May. 2014].

[8] Oss-watch.ac.uk What is version control? Why is it important for due diligence? In-text: (Oss-watch.ac.uk, 2013) Bibliography: Oss-watch.ac.uk, (2013). What is version control? Why is it important for due diligence?. [online] Available at: http://oss-watch.ac.uk/resources/versioncontrol [Accessed 22 May. 2014].

[9] Build software better, together In-text: (GitHub, 2014) Bibliography: GitHub, (2014). Build software better, together. [online] Available at: https://github.com/about [Accessed 22 May. 2014].

[10] Microsoft.com Microsoft Security Development Lifecycle In-text: (Microsoft.com, 2014) Bibliography: Microsoft.com, (2014). Microsoft Security Development Lifecycle. [online] Available at: https://www.microsoft.com/security/sdl/default.aspx [Accessed 23 May. 2014].

[11] https://www.appsecconsulting.com/Application-Security/application-security-program-development/menu-id-62.html

[12] How To Set Your Consulting Fees Forbes How To Set Your Consulting Fees In-text: (Forbes, 2006) Bibliography: Forbes, (2006). How To Set Your Consulting Fees. [online] Available at: http://www.forbes.com/2006/11/06/bostonconsulting-marsh-mckinsey-ent-fin-cx_mc_1106pricing.html [Accessed 23 May. 2014].

[13] Startmyconsultingbusiness.com How to set your hourly consulting rate In-text: (Startmyconsultingbusiness.com, 2014) Bibliography: Startmyconsultingbusiness.com, (2014). How to set your hourly consulting rate. [online] Available at: http://startmyconsultingbusiness.com/how-to-set-your-rate/ [Accessed 23 May. 2014].

[14] Forbes Low Consulting Rates Leave Accenture High And Dry In-text: (Forbes, 2011) Bibliography: Forbes, (2011). Low Consulting Rates Leave Accenture High And Dry. [online] Available

at: http://www.forbes.com/sites/greatspeculations/2011/02/14/low-consulting-rates-leave-accenture-high-and-dry/ [Accessed 23 May. 2014].