Front Page

Contents

1 Introduction

Software development has evolved from something you'd do 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 Thesis Statement

In light of the background information, 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.

2.1 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?"*

2.2 Success Criterias

2.2.1 General Criterias

In order to best realize our goal, we need to set some criterias 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 easy to use. These are all general sucess criterias that are "self-explanatory". 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. Easy to use is subjective, but SecuritySoftware shouldn't require any form of extensive programming knowledge or certifications in security.

2.2.2 SecuritySoftware Specific Criterias

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

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

3 System breakdown

SecuritySoftware's two main components is 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. In this day and age, it's almost impossible to develop software without an internet connection. Therefore, requiring an internet connection should not be a big problem.

SecuritySoftware will consist of 4 different steps, each one dependant on the one beforehand. (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.

3.1 Step By Step Explanation

Below we'll list the rundown of each step you'll encounter when running SecuritySoftware, and what you need to do to complete each one. If you need to, SecuritySoftware will allow you to go back to a previous step if changes has to be made. This will then change the final analysis.

3.1.1 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.

3.1.2 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 you are planning to offer with your application. Without features, security wouldn't be an issue! Here the user will be presented with a search box, along with a list of commonly used features. The search box will be updated as you type, meaning if you're looking for authentication, it would show up while typing "auth". Each feature will also be applied a "tag", meaning someone could search for "Log In" and they would be able to select "Authentication."

3.1.3 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 you add a server to this step, you'll be asked to list what this server will handle for each feature. If you have a database, and you want your users credentials to be stored on the database, then the user would list this after adding the database. This means that SecuritySoftware would know that your database contains highly sensitive information, and you need to safely store it in some way in case of a security breach.

3.1.4 Final Step - The Analysis

Until now, SecuritySoftware hasn't really done anything. It's just 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.

So how does the analysis work? SecuritySoftware relies on a database of information. This database is constructed with several tables. There's a table

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

for each step, 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's 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.

3.2 Example Runthrough

To show what a runthrough with SecuritySoftware would look like, we'll guide you through it step-by-step. 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.

3.2.1 Step 1 - Clients

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

We start with Apple first. Our software will run on mobile devices, both cellphones and tablets/pads. This means we'll choose iOS 7 from the list. Most of the time, you'll always choose the latest version of a client, but in some cases you might want to target phones that can't run the latest versions, meaning if you want to support more, you have to intentionally release it on lower versions. We would therefore list this aswell.

Next up is Android. To make it easier to follow, we'll simply add support for the latest version, which is 4.4.2 KitKat. This will allow our application to

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

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

run on both cellphones and tablets supported by this version.

Both Apple and Android have mobile web browsers, and because we're creating an application that's also accessable via web-browsers, we need to let SecuritySoftware know this by checking an option that says "Include mobile web-browsers".

Last we add support for web-browsers run from PCs. See Table 2 for an overview of what this would look like.

3.2.2 Step 2 - Features

Next up we need to list the features we want our program 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 we want these conversations to be logged, so we need to add a feature for storing logs somewhere. When the user adds Instant Messaging to the list of features, SecuritySoftware will ask if the feature requires a user to be logged in. Because we want to log our chats, we need this to be set to true. The next step, we'll choose where to store our logs and how these features will communicate from the server and to the client.

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.

3.2.3 Step 3 - Server

We've reached the step where we list how the client and server will communicate with each other 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 in the database.

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

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.

3.2.4 Final Step - Analysis

Now for the analysis, where SecuritySoftware finally does some work other than holding your own through the process. All issues and solutions proposed here are not real life situations, and are simply created for simulation purposes.

Clients: First, we look at the clients and see if there's any matches with the issue table. There's no known issues with either iOS clients 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.

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 contain their information, 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 log will be open for everyone in our HospitalApplication. It's therefore not needed to do anything special with these on our database, because you can read it anyway.

Clients and features combined: We've now completed the individual steps for client and feature, but we also need to check combinations of these. Maybe

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

there's 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 he provided earlier, or add the results to his potential software specification.

4 Final Thoughts

SecuritySoftware's 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 work are likelyto happen. But in the end, the idea will remain 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 you 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 some way or the other.

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.

5 Table List

List of Tables

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].