NTNU
Kunnskap for en bedre verden

Bacheloroppgave

IE303612 - Bacheloroppgave

Peephole - ASP.NET Information Security Teaching Tool

130201, 020161, 130204

Totalt antall sider inkludert forsiden: 180

Innlevert Ålesund,

Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. **Manglende erklæring fritar ikke studentene fra sitt ansvar**.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	**Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.**	☒
2.	**Jeg/vi erklærer videre at denne besvarelsen:** • ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • ikke refererer til andres arbeid uten at det er oppgitt. • ikke refererer til eget tidligere arbeid uten at det er oppgitt. • har alle referansene oppgitt i litteraturlisten. • ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	☒
3.	**Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen.**	☒
4.	**Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert i Ephorus, se Retningslinjer for elektronisk innlevering og publisering av studiepoenggivende studentoppgaver**	☒
5.	**Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter NTNUs studieforskrift.**	☒
6.	**Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider**	☒

Publiseringsavtale

Studiepoeng: 20

Veileder: Girts Strazdins

Fullmakt til elektronisk publisering av oppgaven

Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven §2).
Alle oppgaver som fyller kriteriene vil bli registrert og publisert i Brage med forfatter(ne)s godkjennelse.
Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

Jeg/vi gir herved NTNU i Ålesund en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering: ☒ja ☐nei

Er oppgaven båndlagt (konfidensiell)? ☐ja ☒nei
(Båndleggingsavtale må fylles ut)
- Hvis ja:
Kan oppgaven publiseres når båndleggingsperioden er over? ☐ja ☐nei

Er oppgaven unntatt offentlighet? ☐ja ☒nei
(inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)

Dato: 26.05.2016

PREFACE

Society is moving towards digitalization of information which leads to an increased focus on security. Statistics show that there are a lot of vulnerabilities in online applications potentially exposing the users' information. We address this issue by making a teaching/learning tool for information security targeting the Windows environment. We found a gap between the existing solutions and the present vulnerabilities, motivating the bachelor thesis. Throughout this report we will explain the process of making the web application, discuss why our application is needed and how it stands out among other similar applications.

The project group consists of Ole-Martin Bratteberg, Per-Olav S. Eikrem and Jens Vingen.

We would like to thank Juan J. Güelfo, our assigner and security expert at Encripto AS, for giving us the opportunity to choose this thesis and for useful feedback during the development. We would also like to thank Girts Strazdins for being a great supervisor helping us with decision making, planning and knowledge. Huiyun Ge, a student for Master (M.Sc.) in NFC based mobile payments security, has been most helpful testing the application and provided us with valuable feedback.

CONTENT

Contents

SUMMARY

With an ever increasing amount of information online, the need for secure web applications is becoming more and more important. A great way to learn is by practice. An up-to-date, easily extendible teaching tool for ASP.NET facilitating learning by practice is non-existent. With focus om modularity and extendibility, we have created Peephole to fill this gap. An additional goal to this project was for us to learn ASP.NET, MVC and C# which was not a part of our education. Research studies was made to map how to fulfill the requirements towards extendibility and functionality. The development was conducted using Scrum methodology and resulted in Peephole solution that contains two projects, Frontpage and Bank. The Frontpage is a hub supplying an overview of vulnerable projects within the solution. The Bank project is a vulnerable web application ready for exploitation. Additional projects can easily be created in the solution and added to Frontpage.

The final product has been reviewed and received great feedback. The feedback is consistent with our personal conclusion that the requirements have been fulfilled and Peephole standing out as a product that can be expanded and become a comprehensive tool.

> Peephole is an excellent project that takes web security training to a next level, since it tackles many of the unsolved challenges seen until now.
> - *Juan J. Güelfo, CEO & Lead IT Security Consultant at Encripto AS*

> When I first see the Peephole I feel the UI is very comfortable and many modules.
> - *Huiyun Ge, Master Student at Guangdong University of Technology (GDUT), China*

TERMINOLOGY

Definitions

What	Definition
C#	C# is a modern programming language created by Microsoft. C# was developed to be used in the .NET framework and is based on Java and C++. C# is simple, powerful, type-safe, and object-oriented. [1]
.NET	.NET is a cross-platform software framework developed by Microsoft which is mainly used on the Windows platform. The advantage of .NET is the possibility to use several programming languages with the CLR (Common language Runtime) and that it's using software virtualization. [2]
Agile Methodology	A development methodology that embrace evolutionary development end flexibility towards changes. [3]
Scrum	Scrum is an agile software development methodology for managing product development. Scrums goal is having a development team work as a unit to reach a common goal. This includes physical meetings at a physical or close online collaboration of all team members, as well as daily face-to-face communication across the project. An important part of the Scrum methodology is that the customers can change their minds about what they want and what they need, though this could make it more unpredictable for the developers. [4]
VSTS	VSTS is a cloud collaboration tools made by Microsoft. This tool works with most IDEs and includes version control, tools for agile collaboration, performance testing and integration with other big cloud applications. This tool supports many programming languages, including C# which is the main programming language we have been using.
Integrated development environment	An IDE is a software that enables programmers to develop applications and other software. The software usually implements components like a source code editor, debugging functionality and tools to automate and streamline the development. Intelligent code completion, which helps the programmer completing code snippets, is also a main functionality of an IDE.

	Microsoft's IDE Visual Studio has an intelligent code completion functionality named IntelliSense. [5]
Bootstrap	Bootstrap is a collection of tools for creating websites and web applications that is free and open source. It includes HTML and CSS design templates for most interface components, as well as JavaScript extensions. It is a tool meant to make development of dynamic websites and/or applications a lot easier. Bootstrap is a front end framework. [6]
MVC (Model view controller)	The MVC is a popular structural file architecture framework/pattern. This framework is used for building web applications containing: Model: the application core (usually contains some kind of database) View: responsible of displaying the data Controller: responsible for the inputs, which means it is reads the data from view, controls the user input and then send the data to the model and then is arranging all that data to the user of the application. [7]
Modular programming	Modular programming is dividing the application into separate sections where these sections each have their own functions and routines. These sections have minimal to none interaction between them making it easier to read, understand and extend the application. [8]
Atlassian JIRA	Project and issue tracking. A great tool to plan and track work for agile teams. [9]
Atlassian Confluence	A system to create, collaborate on and gather documentation, notes and requirements. [10]
Atlassian Bitbucket	A service to collaborate on code utilizing Git. [11]
Site.master	View pages can include Site.master in order to include its layout. The master file has the functionality of a template in order to standardize the layout for a group of pages. [12]
DVWA	DVWA is a PHP / MySQL webapplication that is intentionally vulnerable to aid in the process of teaching / learning / testing skills and tools towards web application security. [13]

Abbreviations

ASP.NET	Active Server Pages Microsoft framework
C#	C-sharp (programming language)
CDN	Content Discovery Networks
CLI	Command line interface
CSRF	Cross-site Request Forgery
CSS	Cascading Style Sheets
DLL	Dynamic-link Library
DOM	Document Object Model
DVWA	Damn Vulnerable Web App
EF	Entity Framework
GUI	Graphical user interface
HMI	Human-Machine Interaction
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
LINQ	Language-Integrated Query
MVC	Model-View-Controller
MVP	Model-View-Presenter
NISK	Norwegian Information Security Conference
ORM	Object-Relational Mapping
OWASP	Open Web Application Security Project
SQL	Structured Query Language
UML	Unified Modelling Language
VS	Visual Studio
VSTS	Visual Studio Team Services
XSS	Cross-Site Scripting

1. INTRODUCTION

In our fifth semester of studying Bachelor of Engineering in Computer Science we had a course about Information Security. The teacher Juan J. Güelfo at Encripto AS expressed the need for a teaching tool that demonstrated the vulnerabilities for the ASP.NET framework.

When learning information security, it is important to understand the underlying technologies that the vulnerabilities are a product of. Real world applications are not suitable teaching information security as one of the best ways to learn is to actually exploit the application and look at the code to understand how it is all connected.

There are existing vulnerable web applications that provide a suitable learning environment. What these existing solutions have in common is the focus on open source technologies such as PHP and MySQL, they are outdated or simply not functional.

ASP.NET stands for 15.8 % [14] of all known server-side languages which makes up a significant number of systems running the technology amplifying the demand for Peephole developed with and for the Microsoft platform. The technology

ASP.NET and C# is currently not taught in any courses at NTNU Ålesund. We chose The Peephole project also as a means to learn ASP.NET/C#. By making available the thesis report together with documentation and the source code, this project also aims to be of guidance for other students developing ASP.NET applications.

1.1 Problem to be addressed

To ensure best possible security for web applications, education for the developers plays a major part and education needs practice. With a threat picture that is constantly changing, how do we best create the most comprehensive, useful and over time current information security teaching tool for ASP.NET.

To achieve the desired result, these are the most important aspects to consider:

- Distribution
- Architecture
- Extensibility
- Implemented vulnerabilities

1.2 Constraints

The Peephole solution will not provide a complete teaching environment with a full set of vulnerabilities nor information about such vulnerabilities. Instead, we focus on providing the best possible foundation for worldwide cooperation that over time could result in the most comprehensive and current information security teaching tool for ASP.NET.

1.3 Text Formats

Code-snippets in the thesis will be formatted like shown in Figure 1.1.

```
<div class="code-snippet">
This is an example. As you can see code will have a border at the left to
make it easier to difference text from code
</div>
```

Figure 1.1: Code-Snippet Format Example

Longer citations in the thesis will be formatted like shown in Figure 1.2.

> This is a paragraph with
> text cited from a source.
> Lorem ipsum color sit amet

Figure 1.2: Citation from person or online source example

2 THEORETICAL BASIS

This chapter will elaborate on project management, agile methodology, common vulnerabilities, applications similar to ours and other system development topics.

2.1 Project management

Association for project management defines project management like this:

> Project management is the application of processes, methods, knowledge, skills and experience to achieve the project objective [15]

Considering that the management should cover everything from concept to realization, there are many points that must be covered, including:

- Managing the risks, issues and changes on the project
- Defining the reason why the project is necessary
- Capturing project requirements.

2.2 Agile methodology

There are many companies, organisations and people developing software every day and they all need a way to manage project tasks and communication. In the recent decades “agile methodology” have emerged.

The Agile manifesto attempts to pinpoint what Agile software development is all about, lightweight development methods:

> Individuals and interactions over processes and tools
> Working software over comprehensive documentation
> Customer collaboration over contract negotiation
> Responding to change over following a plan [1]

There are several different agile methodologies, for instance eXtreme Programming, Crystal Clear, Spiral model and Scrum. Scrum is the most common methodology [16]. More than 1000 books have been published on Scrum which has been successfully applied in several segment like manufacturing, marketing, operating and education.

2.2.1 Scrum

The Scrum methodology consists of three main elements [17]:

- Roles
- Artifacts
- Meetings

[1] http://www.agilemanifesto.org

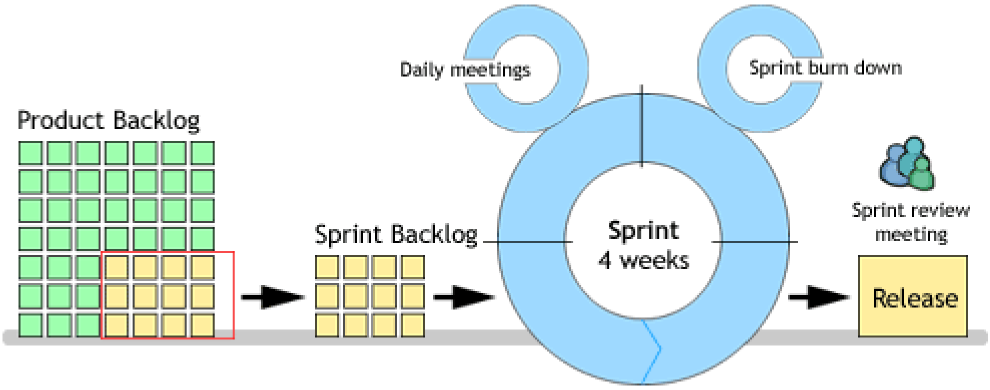


Figure 2.1: The Scrum Development Process [17]

Roles

The roles in a Scrum based development cycle is the product owner, the Scrum development team and the Scrum master.

The product owner is the person(s) that "owns" the product and who prioritizes the backlog of functionality to be implemented. This person is supposed to focus on the *what* not the *how*.

The Scrum development team is a collaborating group of people with (possibly) different experience and knowledge that attempts to build a shippable product during each sprint. They have no leader. In Scrum the leadership evolve from the team and together they delegate features from the backlog to be implemented in the next sprint.

The Scrum master is the third and last role in the Scrum methodology and this role is the most misunderstood role in Scrum. This is because this role does not include any management authority and neither does he have a manager role. The Scrum master is a facilitator, meaning that he is in charge of three aspects; processes, roles and the artifacts. A proficient Scrum master will make him/her-self superfluous.

Artifacts

The artifacts encompass the product backlog and the sprint backlog. The product backlog is a prioritized list of features, bugs and other tasks. It is supposed to describe the *what* and not the *how*, but it says nothing when it is supposed to be done. This list is owned and edited by the product owner.

> … if not in the backlog, it does not exist... [16]

The sprint backlog is the *what* from the product backlog. One feature from the product backlog can have several tasks. The development team selects features from the prioritized queue in the backlog, decide on which tasks must be conducted to create the feature and adds the tasks to the sprint backlog. Figure 2.1 illustrates how the sprint backlog is extracted from the product backlog and the rest of the process.

Meetings

The last element in Scrum is meetings. Meetings is an important factor as to how the methodology has gained its success and popularity.

Types of meetings within Scrum:

- Sprint planning meeting
- Sprint retrospective meeting
- Sprint review meeting
- Backlog refinement meeting
- Daily Scrum

The sprint planning meeting is a meeting between the product owner and the scrum development team. During this meeting they take features from the product backlog and place them in the sprint backlog. They do this before every sprint begins. Now the development team breaks of the features into tasks, the what to how, and they estimate how long each of these tasks would take from start to finish. Sprints are planned one at a time, one sprint planning meeting before sprint startup, the next will be once the first sprint is completed.

The sprint retrospective meeting is the last meeting during a sprint and the goal is process improvement. This is the time where the development team discusses the development and share with each other what went well and what could be improved.

The sprint review meeting is arranged to give the product owner an update as to the current development. The product is demonstrated and the owner gets a general status of the project. The customer can accept or reject the work presented and might present some input on new ideas or changes for the next sprint.

The backlog refinement meeting is where the product owner and the development team is discussing what the most important parts of the project is. Usually this kind of meeting is held because of time/cost-issues and they have to decide on which feature is to have the highest priority.

Daily Scrum is the most iconic meeting of the Scrum methodology. This meeting is also called a stand up meeting. At the start of every working day the whole development team and the Scrum master meets up to discuss the current progress. Each member will have to answer three questions:

1. What did I do yesterday?
2. What will I do today?
3. Any problems?

The whole meeting should not last more than 15 minutes.

2.2.2 Extreme programming

Pair programming is an important element of Extreme programming where the code is written by two developers working on a single computer [18]. One part writes code and the other reads, discusses and comments on the work. This method is useful as it increases code quality without impacting the development time.

Working this way both developers know the code very well making it easier to discuss and work on.

2.2.3 Lean software development

Lean software development consists of seven principles to make the development process more flexible, effective and with less waste [19]:

1. **Eliminate waste**
   Waste is anything that does not add a value to the product. Do not deliver code that semi-finished.
2. **Amplify learning**
   Learning is a key to great success. This makes the developers write faster and better code.
3. **Decide as late as possible**
   Making big development decisions early in the process could be expensive. Focus on main functionality above add-ons the customer wants.
4. **Deliver as fast as possible**
   It is not the biggest companies that survives, it is the ones that deliver fast. The customers are expecting short delivery time and does not care who delivers.
5. **Empower the team**
   The development team needs motivation and a higher goal to reach for. The leaders should support them and help them in difficult situations.
6. **Build integrity in**
   If a user of a system thinks “Yes! This is exactly what I want”, a system is perceived to have good integrity. It is important that the system fits the user’s needs and that the solution is maintainable, adaptable and extensible.
   User-test version of programs should be the same as the developer-version, because this makes it easier to find errors and bugs.
7. **See the whole**
   Keep tasks at a minimal and make bigger tasks into smaller chunks. The lean software development needs every developer, leader, etc. involved to be ready to use lean.

2.3 Existing solutions

Before we began coding the project, we searched for existing and similar solutions. This to ensure that a solution similar to ours was not already developed by someone. During the search we found a few solutions that were similar in some ways: DVWA, WebGoat, Hera Lab and others. After discovering these somewhat similar solutions, we began looking at the differences from their solution and the planned solution. A common factor for most existing solutions was their usage of PHP/ MySQL, Java and that only a few of them were modular. From this we could conclude that our solution targeting ASP.NET and modularity would indeed be unique.

During the project we did find an existing .NET solution known as WebGoat.NET. This solution is created by two volunteers at OWASP and is somehow similar to our project.

Below is a list of applications targeting information security awareness.

DVWA [20]

Written in: PHP/MySQL

Hosted: Virtual machine

Created by: OWASP

Price: Free

This applications goal is to be an aid for security professionals to test their skills and tools in a legal environment. This is usually run on a virtual machine, which minimizes the risk for damaging the users' own system.

WebGoat [21]

Written in: Java

Hosted: Virtual machine

Created by: OWASP volunteers.

Price: Free

WebGoat is an insecure web application which is maintained by the OWASP team. They made this to have security lessons in information security.

WebGoat.NET[2] [22]

Written in: C#

Hosted: In visual studio

Created by: OWASP volunteers

Price: Free

The WebGoat.NET application is built on Microsoft ASP.NET and is purposefully vulnerable. It is intended for usage in classrooms and contains common vulnerabilities.

Hera Labs [23]

Written in: Unknown

Hosted: Online virtual machines

Created by: eLearn Security

Price: About 5000$ for the complete package

Hera labs is a penetration and learning tool for security professionals. They offer courses with certification afterwards, but it is a paid service. They are also hosting online virtual machines

Hack.me [24]

Written in: Unknown

Hosted: Online virtual machines

Created by: Online community – based on eLearn Security

Price: Free

Hack.me is a free community based project that is hosted by eLearn Security's solutions. They have small challenges that could both be easy or hard to solve and they also have the possibility to ask for help in a comment field.

[2] We actually didn't manage to run this application, why we do not know, but it seems like there some fault in the code in the GitHub repository.

NOWASP (Mutillidae) [25]

Written in: PHP, JS

Hosted: Local webserver

Created by: Jeremy Druin (volunteer at OWASP)

Price: Free

NOWASP (also called mutillidae) is an open source web-application that is hosted by creating a local server with WAMP, LAMP, or similar solutions. It comes with built-in plugins for some specific tasks.

OWASP Webscarab [26]

Written in: Java

Hosted: Local webserver

Created by: Jeremy Druin (volunteer at OWASP)

Price: Free

OWASP Webscarab is more like a tool for analyzing applications that communicate with HTTP and HTTPS. It contains several plugins that could be used for intercepting proxy and listening to communication between the client and the server.

bWAPP (Buggy web application)

Written in: PHP/MySQL

Hosted: Virtual Machine

Created by: ItSecGames

Price: Free

The bWAPP (or buggy web application) is a free and open source vulnerable web application. Its main goal is helping security enthusiasts, developers and students to explore and prevent security issues in web applications.

2.4 Vulnerabilities

The vulnerabilities we have focused on is listed in the book OWASP Top 10 for .NET developers [27]. The book explains the vulnerabilities, how to exploit and how to fix them.

2.4.1 What is OWASP?

OWASP is an acronym for Open Web Application Security Project and is a worldwide non-profit organization focusing on improving software security. The ultimate goal for OWASP is to encourage and help the society to make informed decisions about security risks.

2.4.2 Injection

The SQL injection exploits poorly filtered or badly escaped SQL queries that could enable unauthorized people to extract sensitive data or even make a complete host takeover.
The attacker would construct an injection and insert it into the system via user input, cookies or one of the many other ways to inject code. All data sent by a browser to a web application could be manipulated to insert an injection.

The untrusted data can trick the interpreter into executing unintended commands, making the attack access unauthorized or hidden data [28].

Here is an example from OWASP.org of how a SQL injection query could look like:

```
' UNION SELECT password FROM Users WHERE username='admin'--
```

This query looks in the table for the column “password” where the username is “admin”. If the injection is successful, the attacker would retrieve the password for the admin user of the system.

There are different types of injection attacks, the most common is a SQL injection with a database, but there is also the less common OS command injection, XML, HQL, LDAP, XPath, XQuery and XSLT.

SQL injection could result in breach of confidentiality, integrity and availability.

2.4.3 (XSS)

XSS occurs whenever an application takes untrusted user input and sends it to a web browser without proper validation and escaping. XSS gives attackers the opportunity to execute scripts in the victim`s browser which can result in hijacking the user sessions, defacing web sites, or redirecting the user to malicious sites.
Below is an example of such user input. This may be placed into any field on a website that allows user input and will display an alert message with “this is an XSS example” if the site is vulnerable [29].

```
<script>alert(‘this is an XSS example’);</script>
```

There are mainly three types of XSS, namely stored, reflected and DOM based.

Stored

Stored XSS, which generally occurs when user input is stored on the target server, such as in a database, message forum, visitor log, comment field etc. And then a victim is able to retrieve the stored data from web application without that data being made safe to render in the browser.

Reflected

Reflected XSS occurs user input is immediately returned by a web application, this can be in the form of an error message, search result, or any other response that includes some or all of the input provided by the user as part of the request, without data being mad safe to render in the browser.

DOM Based

DOM Based XSS is as defined by Amit Klein, a form of XSS where the entre tainted data flow from source to sink takes place in the browser, i.e., the data never leaves the browser.

2.4.4 Broken authentication and Session Management

When dealing with login and logout functionality, functions related to authentication and session management are often not implemented correctly. This allows attackers to compromise passwords, session tokens or exploit other implementation flaws [30].

Here is an example from OWASP on Broken authentication and Session Management:

> Application's timeouts aren't set properly. User uses a public computer to access site. Instead of selecting "logout" the user simply closes the browser tab and walks away. Attacker uses the same computer an hour later, and that is still authenticated. [31]

2.4.5 Insecure Direct Object References

A direct object reference occurs when files, directories or other data we have properly secured and is exposed. Without the proper access control check or other protection, attackers can access and manipulate these references to access unauthorized data. The attacker can manipulate the reference to gain access to data that should be inaccessible [32].

Shodan is a search engine for the Internet of Things, https://www.shodan.io/. A poorly configured webcam could be indexed by the search engine and the stream unintentionally made available for the entire world. Also, Shodan indexes industrial control systems, databases and more. Many of the devices indexed by the search engine is not supposed to be accessible by the public and is subject to insecure direct object references.

A specially crafted search term could also be used to access insecure objects. A Google search for *"Top secret" site:gov.uk filetype:docx* | *filetype:doc* | *filetype:pdf* would result in huge amounts of unsecured .docx, .doc and PDF files located at the UK Governments own servers.

2.4.6 Cross-Site Request Forgery

A Cross-site Request Forgery attack (pronounced «cSurf») is an attack where unauthorized commands are sent from a malicious webpage through the browser to a trusted webpage. An example to this could be a user that is logged in to their mail client and at the same time access a malicious website. The malicious website could send commands to the users mail client through the user's privileges.

CSRF differs from XSS in the way that no code is injected into a trusted webpage, instead the attack is performed from a malicious site that the user has to access.

2.4.7 Security Misconfiguration

In a system there can consist of several frameworks, application servers, web servers, database servers, libraries and more. While developing an application, features might be disabled to make the development process easier. If the developer fails to enable these features again, the application might be in danger. [33]

An example from OWASP on how Security Misconfiguration works:

> The app server admin console is automatically installed and not removed. Default accounts aren't changed. Attackers discovers the standard again pages are on your server, logs in with default passwords, and takes over.

2.4.8 Insecure Cryptographic Storage

When dealing with sensitive information like passwords, credit cards, health records and so on, it requires encryption. It is important that only authorized users can access the data, especially the decrypted data. Today there is a lot of different encryption algorithms out there and it is important to use the one that is strong enough for your system and what type of information is stored [34].

An OWASP example for Insecure Cryptographic Storage:

> An application encrypts credit cards in a database to prevent exposure to end users. However, the database is set to automatically decrypt queries against the credit card columns, allowing an SQL injection flaw to retrieve all the credit cards in clear text. The system should have been configured to allow only back end application to decrypt them, not the frontend web application.

2.4.9 Failure to Restrict URL Access

It is normal for a web application to check URL access rights before rendering protected links and buttons. If this isn't done on every page access, the attackers will be able to forge URLs to access these hidden pages [35].

Here's an example from OWASP on for Insufficient Transport Layer Protection:

> The attacker simply forces browsers to target URLs. Consider the following URLs which are both supposed to require authentication. Admin rights are also required for access to the "admin_getappInfo" page.
> http://example.com/app/getappInfo
> http://example.com/app/admin_getappInfo
> If the attacker we have authenticated, and access to either page is granted, when unauthorized access was allowed. If an authenticated, non-admin, user is allowed to access the "admin_getappInfo" page, this is a flaw, and may lead to the attacker to more improperly protected admin pages.

2.4.10 Insufficient Transport Layer Protection

When using a web application, there is always network traffic. Sometimes the application fails to authenticate, encrypt and protect the confidentiality and integrity of this traffic. This is because of weak algorithms, using expired or invalid certificates or it is not used correctly [36].

An example from OWASP on Insufficient Transport Layer Protection:

> A site simply does not use SSL for all pages that require authentication. Attacker simply monitors network traffic (like an open wireless or their neighborhood cable modem network), and observes an authenticated victim’s session cookie. Attacker then replays this cookie and takes over the user’s session.

2.4.11 Unvalidated Redirects and Forwards

Redirecting is a frequently used function while creating a web application. The attacker uses untrusted data to determine the destination page. If the redirect is poorly written and with no validation, attackers can redirect victims to malware sites or other untrusted sites [37].

Here is an example from OWASP how Unvalidated redirects and forwards works:

> http://www.trustedsite.com/redirect.aspx?url=www.evilsite.com
> If a friend or another person sends you this link, you would probably just look at the first part of the URL “www.trustedsite.com”, but when you look closer this link actually redirects you to “www.evilsite.com”.

2.5 Laws and regulations

This chapter encompasses laws and regulations in Norway only and will variate depending on country.

The laws against exploiting vulnerabilities is according to §145b[3] fines or jail up to 1 year. The law is applicable when someone unlawful manufacture, acquire, possesses or make data available for themselves or others, meaning:

- Usernames or passwords
- Software which is intended to exploit vulnerabilities in other software/applications.
- Possession of self-contagious software.

§145c[4] states that if someone transfers, damages, deleted, impair, changes, adds or removes information which is crucial for the operation of a computer system could be punished with fines or jail up to 2 years. Complicity is punished in the same way.

People cannot just try exploiting vulnerabilities, attack web stores and personal web pages because this may be their business and according to the law you could get jailed for trying to exploit them. Peephole makes it possible with hands-on experience with vulnerabilities within a legal environment.

In Norway there are not many convictions for cyber criminality, a few couple of examples:

- One of the most known cases was the politician Tor Johannes Helleland (from the party Høyre) that stole pictures of young naked women and published them online[5].
- Some teenagers who DDOSed a couple of sites, making them unreachable. Because of their age and the law §291 not being specific they might avoid punishment. The law states that something has to be damaged or destroyed in order for that law to take effect[6].

[3] https://lovdata.no/dokument/NLO/lov/1902-05-22-10/KAPITTEL_2-6#§145b

[4] https://lovdata.no/dokument/NLO/lov/1902-05-22-10/KAPITTEL_2-6#§145c

[5] http://www.nrk.no/norge/risikerer-flere-ars-fengsel-1.11156989

[6] http://www.tek.no/artikler/tenaringene-kan-slippe-unna-straff/109165

2.6 Software development design patterns

When making an application modular, that means separating each functionality or program into smaller sections, or modules. Each of these modules are independent and contains everything necessary to run the functionality is design to do [38].

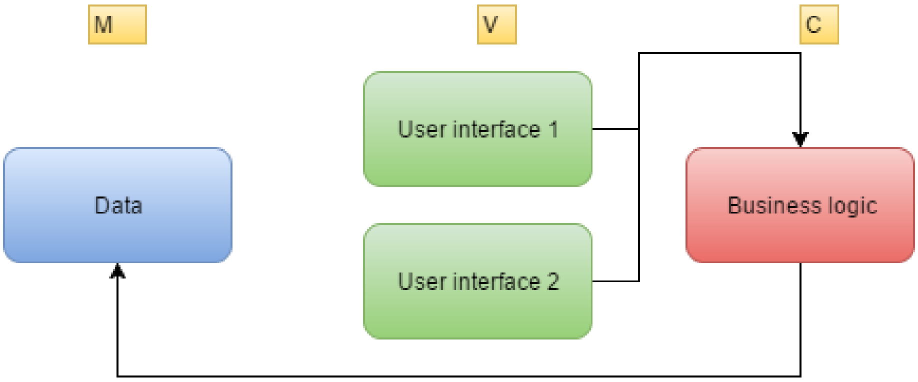


Figure 2.2: This is how the separation of the different layer in MVC works.

MVC is a design pattern used in many frameworks, included the ASP.NET. It separates the user interface from the business logic (See Figure 2.2) [39]. Webforms is using a pattern called MVP, which is much similar to MVC. The differences are that in MVP (See Figure 2.3) the View is more loosely coupled to the model; the Presenter is responsible for connecting the data to the user view. In MVC each controller can have multiple view, but in MVP there can only be one View for each Presenter.

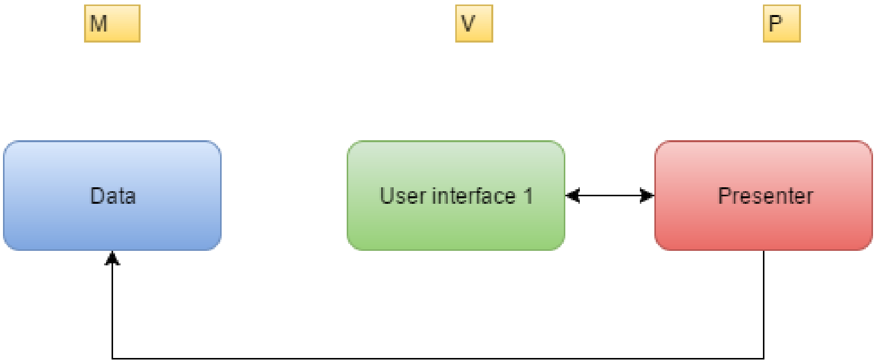


Figure 2.3: How to separation of the different layers in MVP works.

2.7 Design

Designing an interface is not just following a recipe. Peephole takes into account a few UI guidelines from Jeff Johnsons book: *Designing with the mind in mind* [40]. The book covers topics as:

2.7.1 Gestalt principles

Gestalt principles explains about proximity, similarity, continuity, closure, symmetry, figure/ground and common fate.
Humans like it when elements are structured. Items that are located close to each other seems like they are in a group (like buttons, text, etc.).
Closure is used to see objects or elements as a whole and not like a collection of smaller pieces.
Symmetry let us see simple figures rather than complex ones.
Humans also perceive what the Gestalt principles titles common fate, which explains how the human mind perceives items that move together as a group and not as individual items.

2.7.2 Reading is unnatural

Humans are not constructed for reading, instead we find it very easy to understand spoken language. It is important not to use backgrounds colors lower the contrast for the text people are supposed to read.

2.7.3 Color

Managing Image in Different Cultures: A Cross-National Study of Color Meanings and Preferences states that colors influence people in different ways [41]. The colors red usually means danger and the orange color usually means warning. This usually seen in traffic and in web design.

2.7.4 Peripheral vision

Peripheral vision explains about our vision only being able to focus clearly only on the exact spot we are looking at. When displaying a message to a user it is therefore important do display it where the user commonly looks and not some random place.

3 MATERIALS AND METHODS

During this chapter we will look at methods used during the project. We have been through a Scrum development process, different management roles, testing and documented a lot during the project. This chapter also includes which programming languages and software we have used.

3.1 Methods

The methods section will describe the methods used to finalize the product.

3.1.1 Research

As we were new to .NET development we started of this project getting familiar with Visual Studio, C# and VSTS. VSTS were our chosen solution to cooperate on code and the project in general. When getting familiar with .NET development we relied heavily on Microsoft Developer Network [42] for source of information.

Having an overview for .NET development we continued to reviewing current solutions with similar functionality as intended for Peephole (See 2.3 Existing solutions for a summary).

A time-consuming part of the project was to search for, discuss, try, conclude and start all over again when deciding on how to make the Peephole Bank application modular. (See 4.2 Modularity for a list of alternatives).

Once decided on Areas for modularity we could continue the development of Peephole solution. A custom solution for the menu was created, both for making it easier for new Areas/modules to be included in the menu and to have Bootstrap styling for the asp:menu controller (See 0 Table details are attached (See 8.8 Attachment 8: Database tables).

Menu for details).

A study was conducted to map the layout of a typical Norwegian bank web application.

It was decided early in the project to focus on OWASP Top 10 for vulnerability implantations. When decided on the design for the Bank and Areas for modularity we could continue implementing the mentioned vulnerabilities.

The most important sources of information are listed in chapter 7 REFERENCES.

3.1.2 Scrum

As previously mentioned, we use Scrum as our development methodology. We were only three persons and we wanted to use a flat structure to ensure that everyone’s opinion would count equally. We have not had a dedicated Scrum master, this has been a role we took shifts on.

We all think that Scrum is a great methodology for software development as it helps us tackle all the insecurities we have had underway. It does take some time with meetings and planning, but it is well worth it in the end. As a result of the regularly meetings, everyone has been well informed and included in decisions, and planning poker (or Scrum poker) helped us a lot when deciding on the length of each work item in the sprint backlog. This project had a lot of uncertainties with us being unfamiliar with .NET and modularity. With Scrum we could adapt quickly to the changes needed. A more traditional approach, like the Waterfall model, would not be suitable as it is impossible to plan an entire project where technology and content is unfamiliar.

Planning poker

We started using planning poker a bit late in the process, but when we first used it was easy to see the usefulness of it. Planning poker is a gamified technique for estimating how long a task should take. This process can be done with physical cards or with an online application like www.planningpoker.com. For every task in the sprint backlog, each member put's down a number-card (which represents story points) faced down until everyone card is on the table. The one with the lowest and highest number of points will argument for the number given to that particular task. We were surprised by the usefulness of the discussion which gave everyone a much better understanding of the task domain. Maybe the person with the lowest score failed to take an important functionality, the task should have gotten more story points. Maybe the person with the lowest score had a useful code snippet that could be re-used, saving time. Aspects like this were discovered during the planning poker.

Figure 3.1: Scrum Planning Poker

We wanted to try the traditional way of playing it using physical cards. This way it is more social and makes it easier to discuss each other's estimates and opinions.

A study done by Moløkken-Østvold and Haugen shows that:

> The set of control tasks in the same project, estimated by individual experts, achieved similar estimation accuracy as the planning poker tasks. However, for both planning poker and the control group, measures of the median estimation bias indicated that both groups had unbiased estimates, as the typical estimated task was perfectly on target. [43]

By this we can conclude on the usefulness of planning poker that enables us to make far better estimates by eliminating the Dunning-Kruger effect[7].

3.1.3 Extreme programming

We used only parts of this methodology, mainly pair programming. Where the solution was not clear one often wrote the code while the other reviewed the code and contributed with knowledge and discussions.

3.1.4 Time schedule and specification

In the pre-project report (See 8.1 Attachment 1: Pre-project report) we made a Gantt-diagram showing how we thought the development progress would turn out. The estimate was merely a

[7] The Dunning-Kruger effect is a cognitive bias in which relatively unskilled persons is mistakenly assessing their ability to accurate. https://en.wikipedia.org/wiki/Dunning%E2%80%93Kruger_effect

calculated guess made early in the process. As suspected the estimates were far from precise, but it was nice having an indication.

We have been using Scrum development software to keep track of work items. The product backlog contained the specification for the product and the sprint backlog contained the plan for a specific week. We have not created any other specification for the project other than the product backlog. The product backlog we have available is from the day when we started using JIRA.

(See 8.6 Attachment 6: Backlog from JIRA)

3.1.5 Meetings

Upon project startup we agreement with our supervisor to have a meeting every Friday at 11am. At this meeting we would discuss the progress for the week and what we should prioritize for the next week. Each Friday before the meeting we prepared a meeting-note with a fixed structure:

- Date of the meeting
- Participants, who attended the meeting
- Goal for this week
- What we did each day of the week and who did what
- What we wanted to show to our supervisor at the meeting
- What our plan was for next week
- Questions to supervisor

We used this structure from week 6 2016, when we first got access to Confluence. Because of these notes we could always go back and see what we did each week.

During these meetings we also had some questions regards technical aspects and our supervisor had small lessons with us during these meetings. If we wanted to know about something the supervisor could not answer right there, he wrote them down and found us an answer to next meeting or sent us a mail with an answer.

Considering the Lean Software Development principle “Empowering the team”, our supervisor was always good telling us what we did good and what was not. If we had to change anything he also gave us structured feedback about what we should do and how.
(See 8.2 Attachment 2: Friday meetings)

Our assigner at Encripto AS could not ensure high involvement in the thesis because a very tight schedule, but we did have one on-site meeting with him and several email discussions during the project. We discussed the content of the application and he told us he wanted us to focus on the OWASP Top 10. From this point forward this was our main focus regarding vulnerabilities in the application.

(See 8.3 Attachment 3: Meeting with Encripto AS)

3.1.6 Testing and debugging

Debugging is the process of detecting and removing errors from the system. This is very important as errors can cause the system to crash, become unstable or make the system behave erratically.

There are two types of bugs: syntax and logical.

Syntax errors

These are bugs related to spelling errors that prevents the project from compiling.

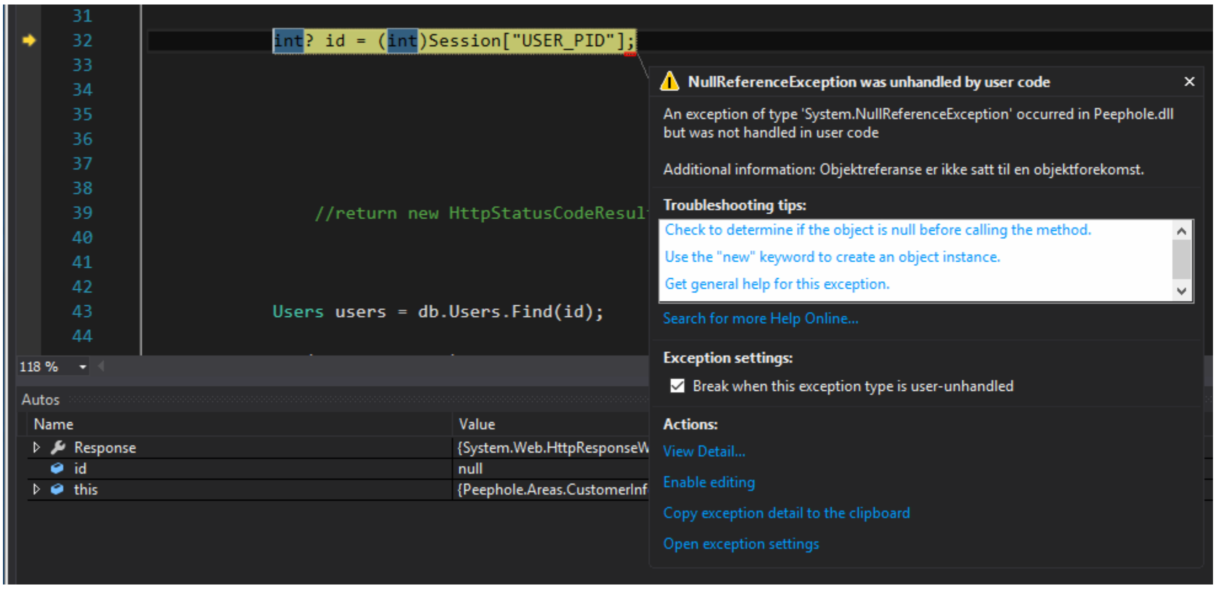
Logical errors

The system compiles and runs but the logical bugs would result in the system not behaving as intended.

The debugging can be done in various ways depending on what is being debugged and which operating system it is running. Some widely used debugging methods are;

- Code and run
- Unit testing
- Breakpoint
- Debugging tools
- Tracing

Testing of Peephole Solution has been done regularly throughout the development, mostly by code and run for syntax errors and debugger / breakpoints for logical errors (See Figure 3.2: Visual Studio debugging utilizing breakpoints).



```
            int? id = (int)Session["USER_PID"];






                //return new HttpStatusCodeResul



            Users users = db.Users.Find(id);

```

NullReferenceException was unhandled by user code

An exception of type 'System.NullReferenceException' occurred in Peephole.dll but was not handled in user code

Additional information: Objektreferanse er ikke satt til en objektforekomst.

Troubleshooting tips:

Check to determine if the object is null before calling the method.

Use the "new" keyword to create an object instance.

Get general help for this exception.

Search for more Help Online...

Exception settings:

☑ Break when this exception type is user-unhandled

Actions:

View Detail...

Enable editing

Copy exception detail to the clipboard

Open exception settings

Autos

Name	Value
Response	{System.Web.HttpResponseW
id	null
this	{Peephole.Areas.CustomerInf

Figure 3.2: Visual Studio debugging utilizing breakpoints

Usability testing

In addition to the testing throughout the development we had a more extensive test towards the end of the development. One usability test was performed by our assigner who has been introduced earlier in the thesis.

His impression and feedback is available in Attachment 9. (See Attachment 9: Feedback).

In addition, our supervisor set up a meeting with three exchange students who specialize in information and computer security. This meeting resulted in a test performed by Huiyun Ge who is a student for Master (M.Sc.) in NFC based mobile payments security at Guangdong University of Technology (GDUT), China.

According to Jakob Nielsen [44], 5 parties is sufficient to test an application. We have had 2 external testers in addition to ourselves and feedback from fellow students. In order to structure the feedback in case of many external testers, we set up a Typeform. This feedback form was distributed as a link to testers, the link being https://brattus.typeform.com/to/h8K2vs.

The feedback from Huiyun Ge was collected using this form.

(See 8.11 Attachment 11: Typeform)

3.1.7 Roles

Using a flat management structure means that everyone has a word in each decision and that they are equally important [45]. The flat structure came natural as a result of Scrum methodology were we have Scrum Master and Scrum development team and us alternating on having the different roles. Every week we alternated on the different roles in order to meet the schools will, to include everyone in every aspect of the development.

As we were a small group we also have a small organisation. We followed Scrum / XP for software development and our organisation was structured like illustrated (See Figure 3.3).

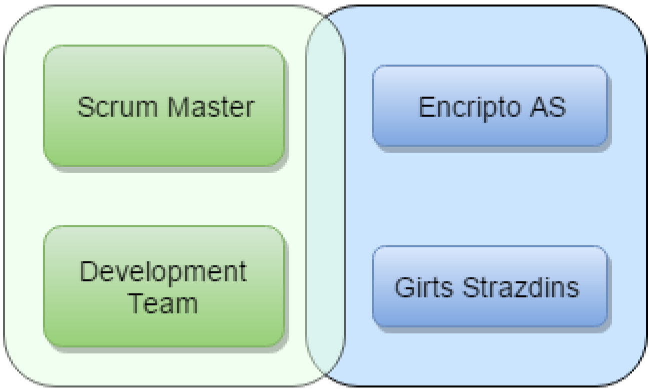


Figure 3.3: Group structure. Green boxes indicate that the function is managed by members of the group, blue boxes are external resources.

Scrum Master Tasks

In addition to the general tasks of a Scrum, for this project the master was also responsible for;

- Prepare for weekly meetings with Girts Strazdins, make an agenda and ensure that all items on the agenda is discussed at this meeting.
- Lead the weekly Friday Scrum meetings, update the backlog and define the next sprint
- Coordinating the continuous workflow
- Gather issues discovered throughout the development and arrange for discussions and solutions.
- Communication with Encripto AS

Development Team Tasks

The members of the group are equally responsible in attending the weekly meetings with our supervisor. Following the weekly meeting is the mandatory sprint meeting where the current and future sprint activities are to be discussed and planned.

During the sprint all members are to pay attention to the tasks in the sprint and work diligently towards the completion of these tasks.

All issues discovered throughout the development are to be shared with the Scrum master appointed for that week.

Cooperation agreement

During the first week of the bachelor project, we created a cooperation agreement. This was for our own safety assuring everyone would meet regularly and at fixed hours.

The agreement is available as Attachment 10. (See 8.10 Attachment 10: Cooperation agreement)

3.1.8 Documentation

As the project is intended to be released to the public as an open-source project we need to have good documentation. This includes documentation of how to use the application, how the application can be extended and known bugs.

International Organization for Standardization defines software like this:

> intellectual creation comprising the programs, procedures, rules and any associated documentation pertaining to the operation of a data processing system. [46]

The software documentation can be presented in both printed and digital versions in which the main task is describe how to use the application [47]. Poor software documentation has been the cause of many project failures and it can reduce the efficiency in the development process drastically [48]. To maintain good software documentation there a list of aspects that helps ensure good quality assurance [49]:

- Completeness
- Understandability
- Traceability
- Clarity

- Consistency
- Evolvability
- Conciseness
- Reusability

The user manual aims to guide the user of the application through installation and usage of the application. The documentation elaborates on different parts of the application, like how the modules work and how to create a new project.

As this is an open source application that we want other people to extend there is also a guide on how further development. The guide explains step by step what an Area is and how it can be used, what information is needed and where to put it.

Considering we are writing a lot of code, we needed to document and organize the code. Means to achieve this was using descriptive names for methods and files, and by giving useful comments throughout the code.

Our project is containing a readme-file which introduces the project. It explains why this project was initiated, what the content is, the possibilities and what programming language is used. Contact information is also provided in case any users should have questions regarding the application.

All of the documentation can be found within the application and as an attachment. (See 8.7 Attachment 7: Documentation)

3.2 Programming languages

During the time as students at NTNU Ålesund we were introduced to Java, HTML, CSS and SQL. The rest of the languages used in this project were new to us. We all spent some time learning these languages, to make sure we were all able to contribute equally in the project.

3.2.1 HTML

HTML is the standard markup language used to create web pages. HTML is used to structure the content and information on a webpage like sections, heading, paragraphs, lists and other types of content. [50]

We have been using HTML for structuring the pages the way we want them to look. The default ASP.NET project gives us a structure, but we have partly changed this into what we thought would be best.

3.2.2 CSS

CSS is usually used to style the HTML in a web page. It can also be used in in XML, SVG and XUL. In the CSS you define sizes, colors and other visual effects to apply to the markup language. The first version of CSS was first proposed by Håkon Wium Lie on October 10, 1994.

Considering ASP.NET by default is using Bootstrap with our own additional code to style the layout.

3.2.3 JavaScript and jQuery

JavaScript is scripting language that is most used to make dynamical elements on web pages, bringing the page to life. This language is used in parallel with other programming languages on the web page and runs within the <script></script> tags or in a separate file.
The syntax is inspired by Java, but other than that they have nothing in common.

jQuery is a JavaScript library designed to simplify JavaScript. It is actually just one large of built-in functions and it is used by 65% of the top 10-million highest trafficked sites [51].

Using JavaScript and jQuery was new to us, but with a little help from Khan academy[8] and Codecademy[9] we found the information needed to create the desired functionality.

3.2.4 C#

C# is a modern object-oriented programming language created by Microsoft. C# was developed to be used in the .NET framework and it is based on Java and C++.

This is probably the programming language we used the most. We discovered it is quite similar to Java, but it has new elements we have never encountered before. When we created the vulnerabilities in the project, this was done by writing C# code in way that is considered bad practice, commonly called "writing bad code".

3.2.5 SQL

Structured Query Language (SQL) is a query language used to interact with databases. In our project we are using MSSQL.

We are using databases a lot and when writing bad code SQL is the key. If we wanted to write secure code, we probably let EF do the database queries for us.

3.2.6 Razor

Razor is not a programming language; it is a template engine using C# syntax for embedding server code in an ASP.NET page.

The files created in Razor syntax are named .cshtml and can contain both HTML and C# code.

[8] https://www.khanacademy.org/computing/computer-programming/html-js-jquery
[9] https://www.codecademy.com/learn/javascript

3.3 Software

3.3.1 Software used

During the bachelor project we have used several different software solutions. We started off using Microsoft Visual Studio Team services. After a few weeks, we switched to Atlassian software. Atlassian is a commercial software, however NTNU Ålesund has installed this software on their local servers, which means we get to use it for free. We thought this was a good opportunity to try the premium software and started using it instead.

Atlassian software

Atlassian software [52] is a complete software pack that contains many tools for the small teams all the way to large enterprise teams. Their most famous software is JIRA, but they also have a lot of other good software like Bitbucket, confluence, HipChat, SourceTree, Bamboo, Clover, Crucible and Fisheye. We have used some of these tools in our project to make the cooperation more seamless.

The possibilities using Atlassian tools represents a big advantage. The tools communicate very well with each other, so that synchronizing information or tasks between them is easy.

Jira

Jira is an issue tracking system that is used both small and major projects [53]. The possibilities with Jira is enormous and highly customizable. Small projects like our bachelor do not need all the tools JIRA can offer. We work metres away from each other allowing us to communicate face to face, discuss and plan during the development. Larger projects were people could be spread all around the world would have even bigger advantages of Jira than we do.
Atlassian consider themselves to be

> the #1 software development tool used by agile teams
> \- *Atlassian* [54]

The product supports agile planning, project estimating, backlog grooming, agile reporting and more.

Confluence

Confluence is a documentation/wiki tool which focuses on teamwork and progress. Everyone who has access can edit and update information in a page. This tool uses an online editor and have a lot of predefined templates which is easy to use.
We have created our own templates which makes it easier to document everything we do during this project. We have our own decision page, which we use to make pros and cons about important decisions. Each Friday, for the meeting with our supervisor, we create the meeting agenda easy and quickly from a template stored at Confluence.

Bitbucket

Bitbucket is a GIT solution much similar to GitHub which handles version control of code stored in the cloud. The service helps us to work on the same code base offering synchronization and conflict handling in case of any conflicts.

Bitbucket also provides a discussion and collaboration section enabling us to discuss changes at BitBucket.org. We have usually done this orally, as we are sitting in a group at almost all times.

SourceTree

SourceTree is a free client for Git and Mercurial that supports almost all platforms. Using Git could require using a “complicated” command line. With SourceTree you do not have to know about Git commands, you are given a nice and simple user interface which makes using Git more user friendly.

> A shout out to developers of SourceTree – a nice GUI for Git and HG. Useful even for a command-line fan like me
> - *Martin fowler* [55]

HipChat

HipChat is a professional communication tool created by Atlassian. This is a free tool that is integrated in the rest of the Atlassian software. It has a secure way of communicating and easy screen and file sharing.

In the beginning of the project, we used Facebook Messenger to communicate with each other. The problem was that we mixed personal and professional stuff. We therefore decided to use this tool to communicate better and more professionally. In reality we pretty much used Messenger because of its simplicity.

Microsoft software

Microsoft is one of the biggest companies in the world and they are not just creating an operating system, but they do make a lot of software to their (and other) operating systems. This is probably one of the reasons why Microsoft is doing as well as they are [56]. Creating these applications which can be installed within the operating system, has probably been crucial for them/their work.

In our project we have used several software solutions from Microsoft besides their operating system, such as Word for writing the thesis, Visual Studio for writing the application code and One Drive for sharing bigger files.

Visual Studio

Visual Studio is the most popular IDE for .NET development and supports all of the most popular programming languages. No matter if you want to create a project using Microsoft languages or you want to use an open source programming language like Python or PHP, Visual Studio is an excellent choice for IDE.

Visual Studio Team Services

VSTS is a cloud collaboration tool made by Microsoft. This tool is compatible with most IDEs and supplies version control, tools for agile collaboration, performance testing and integration with other cloud applications. It supports many programming languages, including C#.

We stopped using VSTS after a few weeks because we were not completely satisfied with how the version control and branch functionality functioned. Since the school offered free use of Atlassian software, we switched Atlassian instead.

Office 365

Office 365 is an online subscription service for those who needs the Office package (Word, Excel, Outlook, etc.).

We use Word to collaborate on the thesis and other reports.

OneDrive

OneDrive is Microsoft’s solution for cloud storage. There are many alternatives for cloud storage, but we chose to use OneDrive as it allows for real-time collaboration for Word documents stored at OneDrive. The real-time collaboration functionality is not available if the Word document is stored at other cloud storages.

We also use this tool to share other project files.

Facebook Messenger

Facebook Messenger is an instant communication tool for Facebook users. It allows anyone who is a part of a spesific group to share text, links, photos, videos and more with the other group members.

Messenger was easily available wherever and whenever as it is installed on both phones and computers. The great availability resultet in a lot of information sharing using this channel.For instance, late at night when one of us discovered a relevant article in the news, it was very quick to share it in messenger for later discussion and inclusion in the project. Messenger was also of great aid when scheduling meetings, notifying of development issues and general sharing of laughter and grief.

3.3.2 Software needed to run the application

We recommend running Peephole in a virtual machine only, this due to security issues.

The virtual machine can be created on the host machine using one of these virtualization solutions;

- VirtualBox (Windows, Linux, Macintosh, Solaris)
  https://www.virtualbox.org/

- VMware (Windows, Linux)
  https://www.vmware.com/

- VPC (Windows)
  https://www.microsoft.com/en-us/download/details.aspx?id=3702

- HyperV (Windows Server)
  https://technet.microsoft.com/en-us/library/mt169373.aspx

- Vagrant (Windows, Linux, Macintosh)
  https://www.vagrantup.com/

A Windows Operating System running on the virtual machine, guest

A Windows ISO valid for 90 days can be downloaded free of charge from
https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/windows/

Visual Studio installed on the Virtual Machine

Visual Studio Express can be downloaded for free from https://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx

Installing the Peephole solution on the Virtual Machine

Open Visual Studio and clone from Git, the repository address is
https://github.com/Urdar/Peephole.git

More detailed instructions can be found in Attachment 7. (Attachment 7: Documentation)

4 RESULTS

The application is modular with the possibility to extend both with new Areas within the Bank project and by adding new projects in the solution, side by side with Peephole Bank. We have facilitated the application for both Webforms and MVC in order to avoid excluding any developers extending the solution. The vulnerabilities we have implemented is inspired from OWASP Top 10.

In this chapter we will elaborate on how we decided to build the application, the vulnerabilities implemented, how Peephole stands out towards other similar applications and security expert feedback on the solution.

4.1 Architecture

In the planning phase of the thesis we used a reasonable amount of time trying to figure out what kind of architectural structure was most commonly used and what would be used in the future.

4.1.1 Webforms

This has been the traditional way of making a web application in ASP.NET. It has drag and drop server controls, server events and state management techniques. There are a lot of tutorials on this online, and when searching on forums we could find a lot of good information (like how to fix error messages). The documentation from Microsoft's official pages are good, but some of them may be old.

4.1.2 MVC

To our knowledge this is the current way of doing it. Microsoft has been putting a lot of effort into MVC in their new ASP.NET Core 1.0 which will release later in 2016 [57]. The MVC pattern is used in several other software development frameworks intended for PHP, Android, Java and others. MVC is lightweight and provides full control of the mark-up.

4.1.3 Web Pages

Web pages is also using Razor syntax. It is lightweight and easy to start developing, but has a few limitations which makes it difficult to use when scaling the application. Peephole could potentially evolve into a big project, so web pages is not a relevant architectural design for the project.

4.1.4 Conclusion

Since we wanted to build a scalable and modular web application, we could not decide on whether Webforms or MVC was the most suitable option. We concluded that Webforms was considered the traditional way of developing and that MVC was the current and increasingly popular alternative. MVC uses the Razor syntax while Webforms uses ASP.NET delimiters, <%= %>. Both can embed C# code into the view page. Webforms had the most tutorials and information about information security, but MVC is the chosen structure for most developers. We started looking into using both concurrently. After some time, we were assured it was indeed possible, even with shared Site.master layout file.

With the solution found we increased flexibility for the developers and in addition enabled us to find a lot more information security content online than if we had support for MVC only.

Originally we had three options, where two of them are explained earlier (See 2.6 Software development design patterns).

The shared layout in Site.master is very easy for developers to utilize, they only have to change the return statements in the controller from return View() to return this.RazorView(), as illustrated below.

Default return statement

```
public ActionResult Index()
  {
    ViewBag._Title = "MVC Template";
    return View();
  }
```

Custom return statement

```
    public ActionResult Index()
  {
    ViewBag._Title = "MVC Template";
    return this.RazorView();
  }
```

A side effect by sharing the Site.Master is that Visual Studio erroneously changes the namespace automatically whenever the file is being edited. To resolve this the developer has to change it back to the correct namespace. Why this happens we do not know, we guess it is a combination of Visual Studio automatic process in combination with the custom solution in order to share the layout between Webforms and MVC areas.

Below you can see what needs to be changed every time there is a change to Site.Master.

Correct namespace

```
namespace Peephole {
```

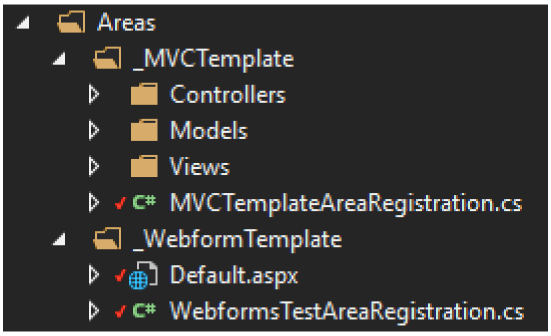
Incorrect namespace

```
namespace System.Web.Mvc {
```

When combining these two, you also have a mix-up in the file structure. As mentioned in the modular section (See 4.2 Modularity) we are using Areas as modules, which has a MVC structure by default.

To use Webforms in an Area, simply delete all files but the AreaRegistration file, this file us used in context with the menu (See 0 Table details are attached (See 8.8 Attachment 8: Database tables).

Menu).



```
◢ Areas
   ◢ _MVCTemplate
      ▷ Controllers
      ▷ Models
      ▷ Views
      ▷ ✓ C# MVCTemplateAreaRegistration.cs
   ◢ _WebformTemplate
      ▷ ✓ Default.aspx
      ▷ ✓ C# WebformsTestAreaRegistration.cs
```

Figure 4.1: How the MVC template and the Webforms template looks like in Areas

No need for internet connection

When creating this application, we also had in mind that it should be possible to run it on a virtual machine without internet. By this the system should not load any JavaScript or jQuery from content discovery networks (CDNs) through the internet, which is a common thing to do. Because of this we stored them locally in folders.

For an overview of the Peephole structure, see Figure 4.2 below.

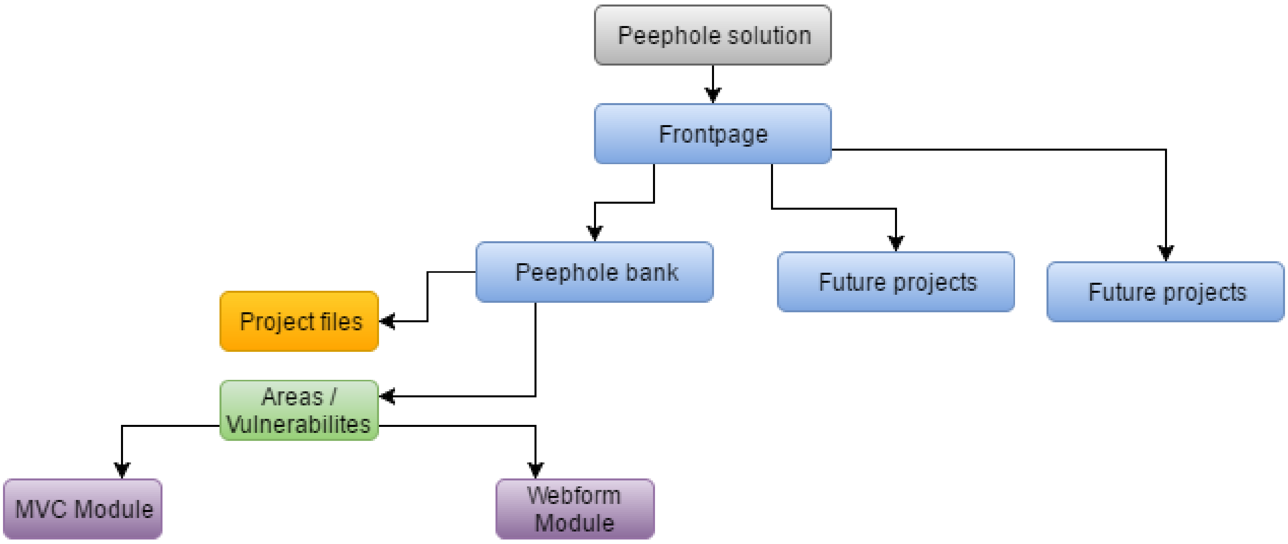


Figure 4.2: Project structure. The boxes are either folder or a group of files.

The complete file structure for Peephole Bank is available as Attachment 4. (See 8.4 Attachment 4: File structure).

4.2 Modularity

The first thing we thought about when making this application was how to make it easy to extend, and the reasons for making it so are many:

- Make it open source and free to let other people continue our work after we are done. Making it modular would make it easier for others to build additional modules.
- Make it easier for ourselves to develop more functionality faster.
- More rapid deployment of new projects.
- Have a platform to gather all these projects.
- Easier extension and change each time something new is created, you could reuse some parts of code.
- Easier maintenance and debugging of the code

For learning purposes, users should have access to the code as it is easier to understand how the vulnerabilities and exploitations are functioning. To achieve this, we decided it was better to distribute the entire project in a Git repository contrary to distribute the modules as plug-in .dll files.

With access to the code, for instance if used in a course, the teacher could make tasks in context to specific the code. If the teacher for some reason do not want the students to have access to the code, the project could be exported as a .dll and distributed to the students.

We did use a lot of time figuring out how to best make the modularity. There were several options further described in the following subsections.

4.2.1 Multiple projects within solution

In a ASP.NET solution you will have the opportunity to create as many project as you want. Each of these project could have their own .NET version, you could create them as a MVC project, a traditional Webforms project, a Web Pages project or just a simple html site. Each of these projects can be a complex web application, they could contain just classes of C# code or they could be just databases; the opportunities are endless.

One idea of ours was having each module as a separate project in the solution. Each project gets its own port at localhost. Linking between projects that should look like one project would be very difficult due to the port issue. This would also complicate sharing of layout. Importing or inherit the Site.master file from another project was problematic. We concluded that this would not be a good solution for making the Bank project modular.

4.2.2 Multiple projects within project

Multiple project within projects means creating a project for each module and then importing that modular-project to a main project. This would make each module have their own port when running the application locally. We made 10 modules and that would have been 10 projects, 10 different ports to redirect to and 10 different project to load each time we want to run the project. The management would be very good, at least if there was not going to be too many modules, but the amount of time it would take to load each project makes this option not suited for Peephole.

Microsoft has a guide for this type of implementation on their site[10].

4.2.3 Shared data as NuGet-packages

Shared data as NuGet-Packages seemed like a more difficult approach to modularity. By doing it this way, we would have to make a .dll of the Areas. Proposed ways of distributing the DLLs was by a downloading from a website or through NuGet. We found this approach overly complicated and not suited for Peephole.

4.2.4 Areas

Areas was a structural architecture that we found early during our process and found out that it could be used to create each module. Areas is a way to separate larger MVC projects into folders that contains "Models, view and controllers".

An Area usually have this structure:

- ProjectName
  - Areas
    - Products
      - Controller
        - NameController.cs
      - Model
      - View
        - Home
          - Index.cshtml
    - Services
      - Controller
        - NameController.cs
      - Model
      - View
        - Home
          - Index.cshtml
  - Other project files

To access the product index, you would go to www.domain.com/areas/products/ in the browser.

We liked this structure since it would make creating modules easy, but still with a good structure. The Area comes with MVC structure by default. We made some changes to the project (See 4.1 Architecture) so that the MVC Area also could utilize the Site.master. This was a great option for making the Bank application extendible.

4.2.5 Portable areas

Portable areas are a set of reusable multi page functionality, and can be dropped into an application without having to custom build, which would be the same in every application. This sounded like a good idea when we first heard of it, but we later found out that accessing the source code could be a bit harder. This is because the portable areas are saved as a .dll and contain items. These items are

[10] https://support.microsoft.com/en-us/kb/307467

the Area section as we mentioned in the previous section; Views, Controllers, Models and other files like JS, CSS and images.

The more we read about this solution, we found out that this was not the way we wanted it to be.

Portable areas could probably work better if you are at a late stage of the process and not in the beginning of the development phase. When we completed the project, we did only have one project. If portable areas should come in handy the best way would be several project containing the same portable areas, then you could save a lot of space and time.

4.2.6 DLL

It was considered to distribute the system as a DLL ready to deploy on a IIS webserver and the modules as DLLs plugin. We would create a website where developers could upload the modules and the end user could download the modules of interest and import into Peephole through a web interface.

This solution would prevent the users of the system from accessing the source code and complicate the meaning of this teaching tool; to learn information security. An important aspect of teaching is to investigate and learn from the actual source code.

4.2.7 Conclusion

As the Peephole project is meant to be used as a teaching tool, users need to have access to the actual code. It was decided to distribute the entire source code through Bitbucket. Since we wanted to make the source code easily accessible, we could not use methods that distributed DLLs. The chosen solution with Areas makes the application extendible in two directions; using Areas for modules within a project and through multiple project to make a whole new project within the Peephole solution. For instance, a developer could create projects like “Peephole Social Media”, “Peephole Accommodation” or any other ASP.NET application.

Extending “Peephole Bank” with Areas

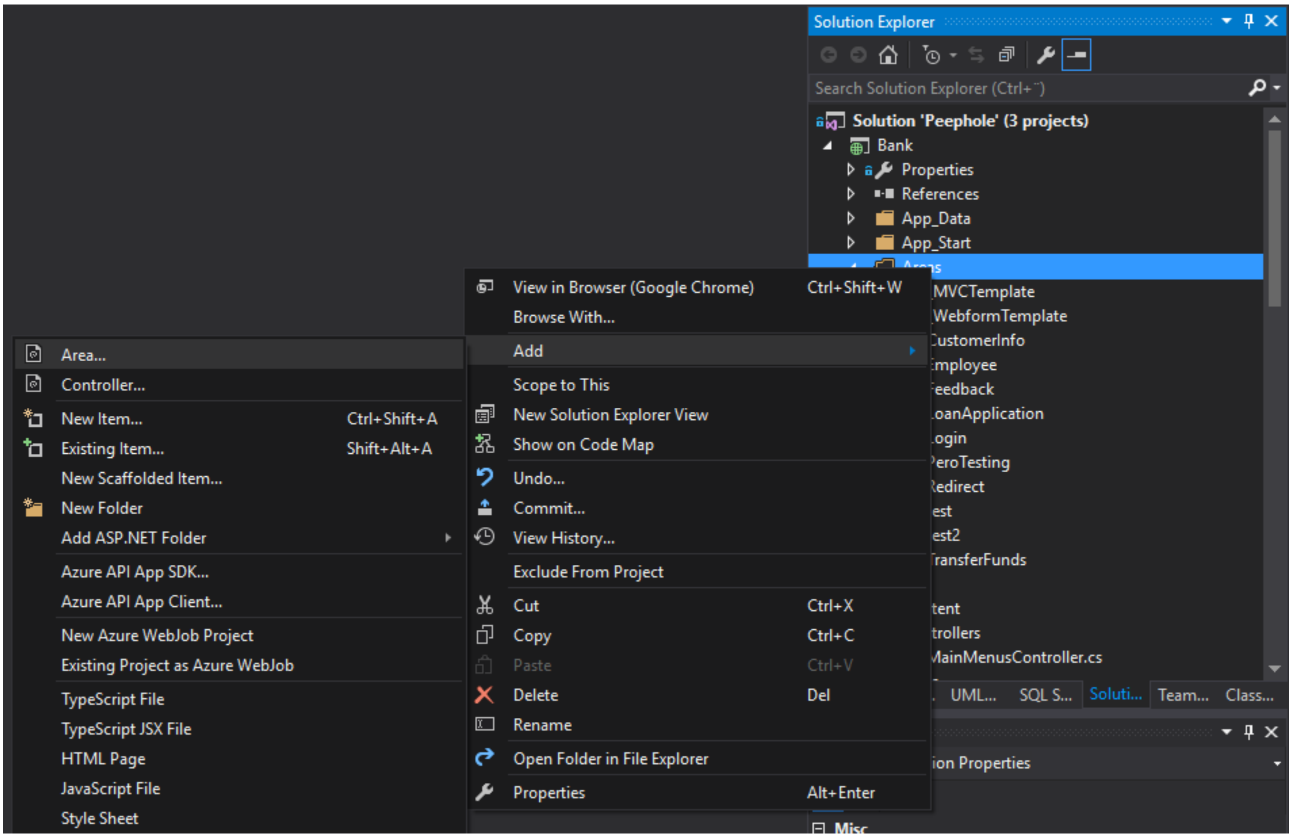


Figure 4.3: Adding a new Area

When adding a new Area this could be done using the built in functionality in Visual Studio which is simply right clicking the *Areas* folder, *Add* and then *Area* (See Figure 4.3: Adding a new Area). Since the Area could contain either Webforms or MVC/Razor, you may have to do some changes to the structure of the code, more about that in the documentation.

The documentation is available as Attachment 7. (See 8.7 Attachment 7: Documentation)

Extending "Peephole" with additional projects

To add a new project, simply right click the solution, go to sub menu "Add" and select "New Project", see Figure 4.4: Adding a project.

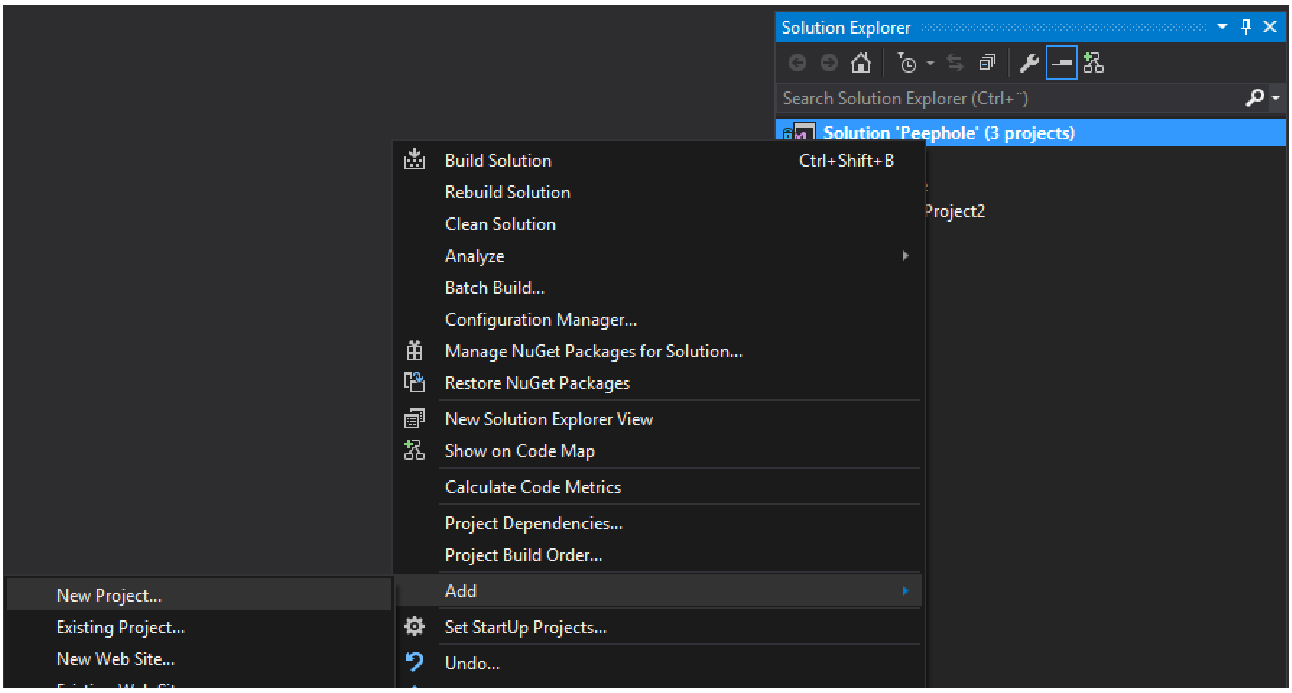
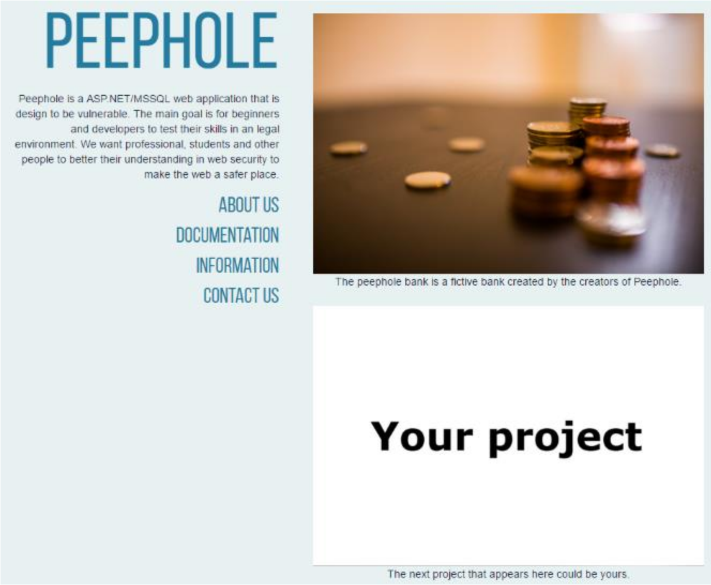


Figure 4.4: Adding a project

When adding these additional projects, you also need to make a link to that project in the Frontpage project. The Frontpage (see Figure 4.5) is the start-up page for the whole solution and that is where you will choose what application you want to access.



Peephole is a ASP.NET/MSSQL web application that is design to be vulnerable. The main goal is for beginners and developers to test their skills in an legal environment. We want professional, students and other people to better their understanding in web security to make the web a safer place.

The peephole bank is a fictive bank created by the creators of Peephole.

The next project that appears here could be yours.

Figure 4.5: Peephole Frontpage GUI

Each of the application have a picture or a frame each that shows what kind of application it is. When hovering over the picture, you will see the application name and a short description of the project. When adding a new project, you will have to either edit or add a few links of HTML code to make a new picture for your project. In this code you also have to edit the header, project name and a description of the of the project (See Figure 4.7).

In the Bank application (See Figure 4.6)
Header: Peephole
Project: Bank
Description: The Peephole Bank is a fictive bank created by the creators of Peephole.

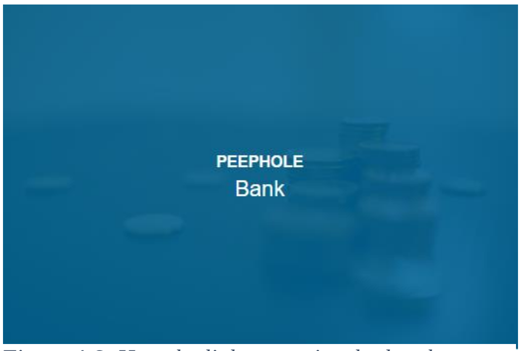


Figure 4.6: How the links to project looks when hovering

```
<div class="content col-md-9 col-sm-9 col-lg-9">
   <a href="http://localhost:3988/" class="portfolio-box">
      <!-- Add image-->
      <img src="img/bank.jpg" class="img-responsive" alt="">
      <div class="portfolio-box-caption">
         <div class="portfolio-box-caption-content">
            <div class="project-category text-faded">
               <!-- Header-->
               Peephole
            </div>
            <!-- Project name-->
            <div class="project-name">
               Bank
            </div>
         </div>
      </div>
   </a>
   <span class="description col-md-12 col-sm-12 col-lg-12">
      The peephole bank is a fictive bank created by the creators of Peephole.
   </span>
</div>
```

Figure 4.7: Code For "New Project" Image At Peephole Frontpage

In Figure 4.7 you see the code that makes a new project picture with the text needed. This uses some of the bootstrap framework to style the pictures and the structure.
If you want to use another image, this is also where this is edited. It is possible to add an image from the web, but it is preferable to save the image locally.

4.3 Vulnerabilities

When we first started this project we wondered if the different .NET version had any vulnerabilities. We did a lot of research on this topic and we wanted to combine the multiple project with different .NET versions, because you can only have one .NET version for each project. While searching for this, we did not find out anything specific other than a site that list all .NET vulnerabilities[11].

After discussing and brainstorming with our supervisor and our assigner we found out that it does not matter. As long as the code is written in a not-suggested way and handles data in a bad way, it would not matter if there is a vulnerable part in .NET 2.0 or 4.5. .NET does not tell the developers if the code is bad or could be vulnerable to injection or other attacks.

We then decided to use the OWASP Top 10[12] and tried making modules and vulnerabilities from that list. This worked very good, and it is highly relevant since it is the most common types of attacking web applications and web sites.
Below we will describe what we did in each of the vulnerabilities, how it could be exploited and possible fix the vulnerability.

When making an application focused on teaching information security and how to handle user data in the .NET framework, it is crucial to stay current for todays and future vulnerabilities. [58]. We did this by looking through different security reports such as: OWASP Top 10 (See 2.4 Vulnerabilities) Application Security Risks and WhiteHat Security – 2014 Website Security Statistics Report[13]. By this research we got an overview of which vulnerabilities to were the most common.

For an overview of implemented vulnerabilities and where to find them in the application, see Table 1: Peephole .

4.3.1 Overview

A short overview of the vulnerabilities implemented in Peephole Bank

Vulnerability	**Area Name**	**Link Name in Menu**
Injection	Employees	Employees
Cross-Site Scripting (XSS)	Feedback	Feedback
Broken authentication and Session Management	Whole application	Not in menu
Insecure Direct Object References	TransferFunds	Make Payment
Cross-Site Request Forgery	Feedback	Feedback

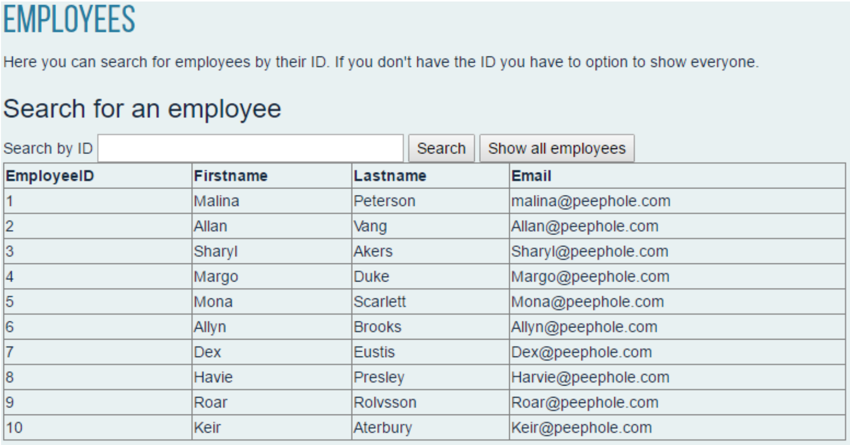
[11] https://www.cvedetails.com/vulnerability-list/vendor_id-26/product_id-2002/Microsoft-.net-Framework.html
[12] https://www.troyhunt.com/free-ebook-owasp-top-10-for-net/
[13] http://info.whitehatsec.com/rs/whitehatsecurity/images/statsreport2014-20140410.pdf

Security Misconfiguration	Employees	Employees
Insecure Cryptographic Storage	CustomerInfo	Profile
Failure to Restrict URL Access	LoanApplication	Loan Application
Insufficient Transport Layer Protection	Whole application	Not in menu
Unvalidated Redirects and Forwards	Redirect	Not in menu

Table 1: Peephole bank vulnerabilities overview

4.3.2 Injection

The injection in the Bank application is an SQL-injection done when you search for employees, but you insert a malicious query which the database interprets like a query to do something else than searching (See Figure 4.8).



EMPLOYEES

Here you can search for employees by their ID. If you don't have the ID you have to option to show everyone.

Search for an employee

Search by ID

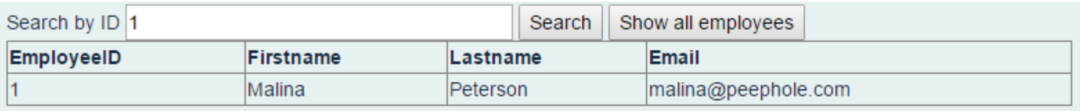
EmployeeID	Firstname	Lastname	Email
1	Malina	Peterson	malina@peephole.com
2	Allan	Vang	Allan@peephole.com
3	Sharyl	Akers	Sharyl@peephole.com
4	Margo	Duke	Margo@peephole.com
5	Mona	Scarlett	Mona@peephole.com
6	Allyn	Brooks	Allyn@peephole.com
7	Dex	Eustis	Dex@peephole.com
8	Havie	Presley	Harvie@peephole.com
9	Roar	Rolvsson	Roar@peephole.com
10	Keir	Aterbury	Keir@peephole.com

Figure 4.8: Show all employees

When using the “Employee”-site normally, you will get a list of all the employees with their ID, firstname, lastname and email. This is the result you would expect from this kind of page. Since this website is vulnerable, the search-field could be inserted with malicious queries. These queries could make the system display information not intended, delete databases or any other SQL command.

```
var sqlString = "SELECT * FROM Users WHERE ID = " + ID;
```

This is how the query on the server-side looks like. This means that if you write the number “1” in the search-field, you would return only “Malina Peterson” (See Figure 4.9):



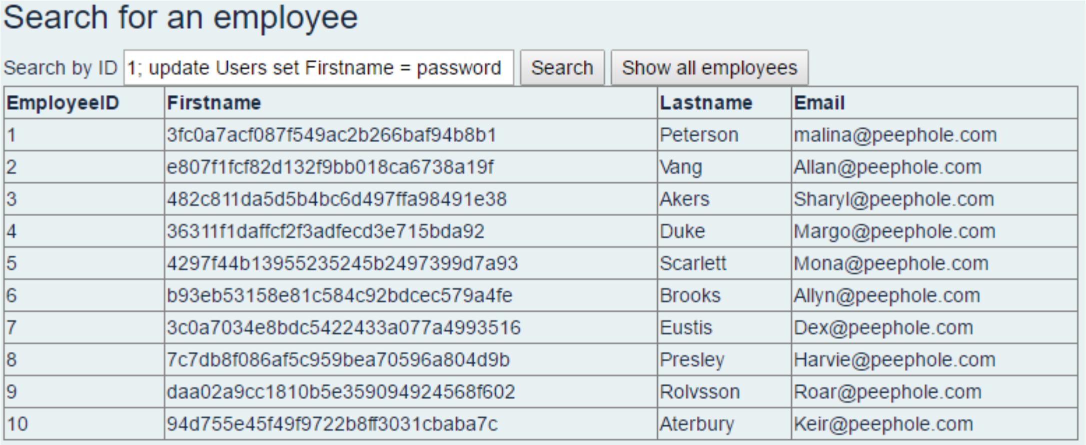
Search by ID 1

EmployeeID	Firstname	Lastname	Email
1	Malina	Peterson	malina@peephole.com

Figure 4.9: Search for specific employee

The database would read the following as SELECT * FROM Users WHERE ID = 1

If you find out names of the other columns in the table, the attacker can retrieve anything from that database (See Figure 4.10).



Search for an employee

Search by ID: 1; update Users set Firstname = password

EmployeeID	Firstname	Lastname	Email
1	3fc0a7acf087f549ac2b266baf94b8b1	Peterson	malina@peephole.com
2	e807f1fcf82d132f9bb018ca6738a19f	Vang	Allan@peephole.com
3	482c811da5d5b4bc6d497ffa98491e38	Akers	Sharyl@peephole.com
4	36311f1daffcf2f3adfecd3e715bda92	Duke	Margo@peephole.com
5	4297f44b13955235245b2497399d7a93	Scarlett	Mona@peephole.com
6	b93eb53158e81c584c92bdcec579a4fe	Brooks	Allyn@peephole.com
7	3c0a7034e8bdc5422433a077a4993516	Eustis	Dex@peephole.com
8	7c7db8f086af5c959bea70596a804d9b	Presley	Harvie@peephole.com
9	daa02a9cc1810b5e359094924568f602	Rolvsson	Roar@peephole.com
10	94d755e45f49f9722b8ff3031cbaba7c	Aterbury	Keir@peephole.com

Figure 4.10: SQL-injection. Firstname is swapped with password

The query executed would be:

```
SELECT * FROM Users WHERE ID = 1; UPDATE Users SET Firstname = Password
```

Now the attacker has the password hashes. Since this is hashed with MD5, cracking it would take seconds by using a rainbow table[14].

An improvement would be to sanitize the input from user. This could be fixed by checking if the input is other than numbers in this example (if the search is done by text, filter the text by symbols like ;'@-.

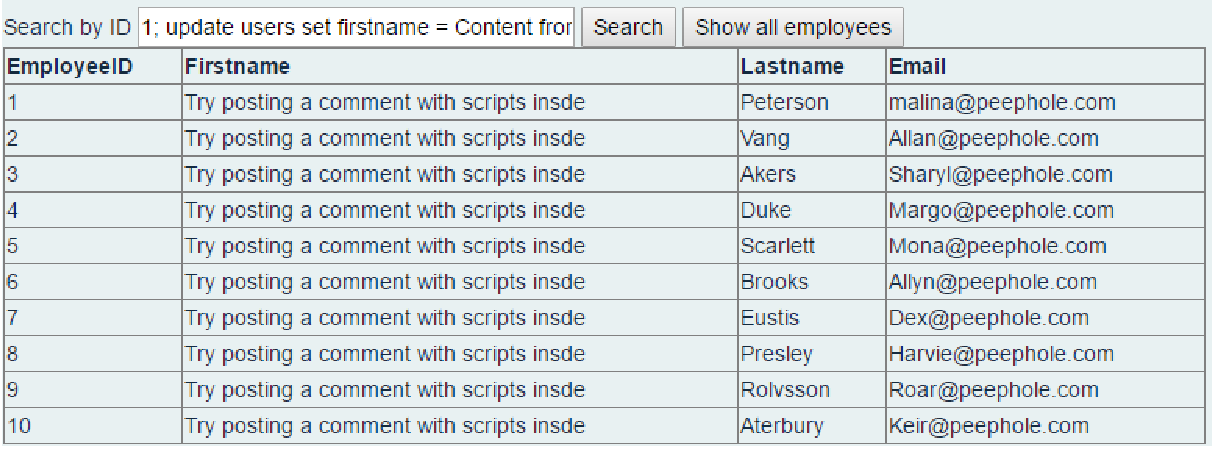
```
var positiveIntRegex = new Regex(@"^0*[1-9][0-9]*$");
        if (!positiveIntRegex.IsMatch(ID))
        {
           //Code
        }

        else {
           //Code
        }
```

This code above allows only number to be put into the search-field, making an SQL-injection harder.

Since all the tables are stored on the same database, it is possible to retrieve information from other tables as well.

[14] There is a lot of MD5 crackers on the web. This is the one we like the best http://md5cracker.org/



Search by ID 1; update users set firstname = Content fror

EmployeeID	Firstname	Lastname	Email
1	Try posting a comment with scripts insde	Peterson	malina@peephole.com
2	Try posting a comment with scripts insde	Vang	Allan@peephole.com
3	Try posting a comment with scripts insde	Akers	Sharyl@peephole.com
4	Try posting a comment with scripts insde	Duke	Margo@peephole.com
5	Try posting a comment with scripts insde	Scarlett	Mona@peephole.com
6	Try posting a comment with scripts insde	Brooks	Allyn@peephole.com
7	Try posting a comment with scripts insde	Eustis	Dex@peephole.com
8	Try posting a comment with scripts insde	Presley	Harvie@peephole.com
9	Try posting a comment with scripts insde	Rolvsson	Roar@peephole.com
10	Try posting a comment with scripts insde	Aterbury	Keir@peephole.com

Figure 4.11: Content from another table

Microsofts Entity Framework[15] is commonly used in ASP.NET. This framework eliminates the need for most of the data-access code for the developers and that means it is less vulnerable because of the lack of human mistakes.

EF is one of the most secure ways of displaying and using database information, but there are other ways as well:

- LINQ[16]
- Parametrized queries[17]
- Regular expression[18] (which is used in the example code above)

Cross-Site Scripting (XSS)

In the OWASP Top 10 report XSS made it to a third place and a second place in WhiteHats report. Based on these reports we chose to implement the XSS vulnerability in the web application.
As mentioned in section 2.4.3, there are three types of XSS. In the web application we chose to implement two of them; Reflected and Stored XSS.

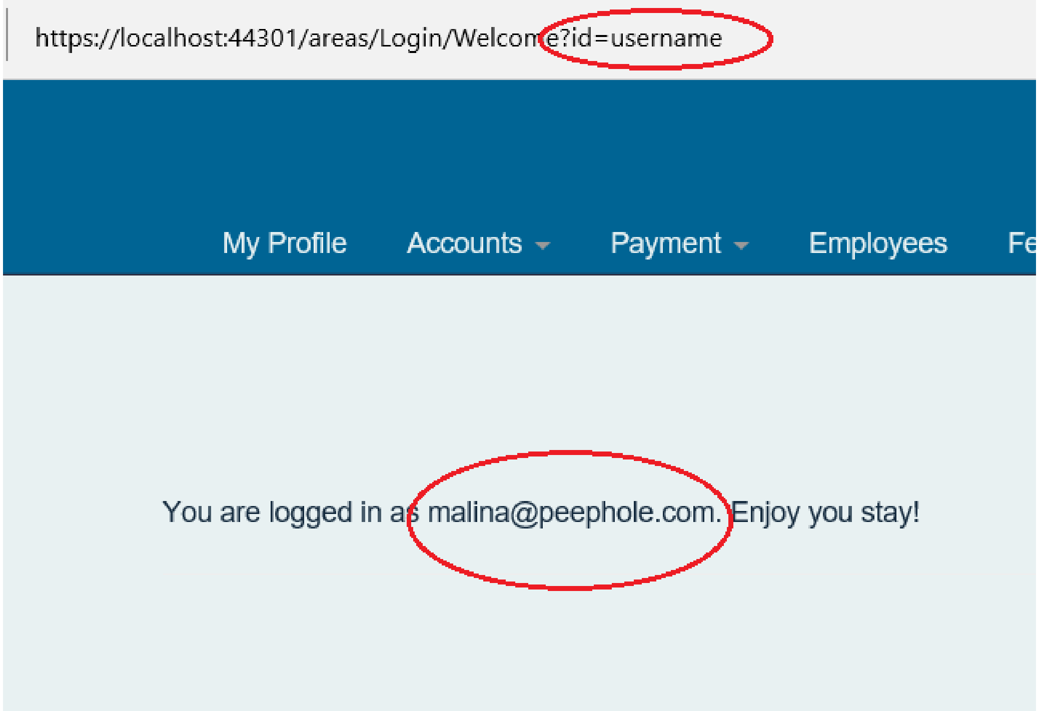
We made the application look as realistic as possible, and less like a “go-to-vulnerability” kind of application. The user is presented with links and functionality usually found in a bank, so we had to find natural points to implement the vulnerabilities. Reflected XSS can be found and only accessed after a user has logged in. After the user has been authenticated he is redirected to a welcome page, this is where we show a static welcome message along with the username in the form of a variable seen in the URL (See Figure 4.12).

[15] http://www.asp.net/entity-framework

[16] https://msdn.microsoft.com/en-us/library/bb907622%28v=vs.100%29.aspx?f=255&MSPPError=-2147217396

[17] http://www.asp.net/web-forms/overview/data-access/accessing-the-database-directly-from-an-aspnet-page/using-parameterized-queries-with-the-sqldatasource-vb

[18] https://msdn.microsoft.com/en-us/library/ms972966.aspx

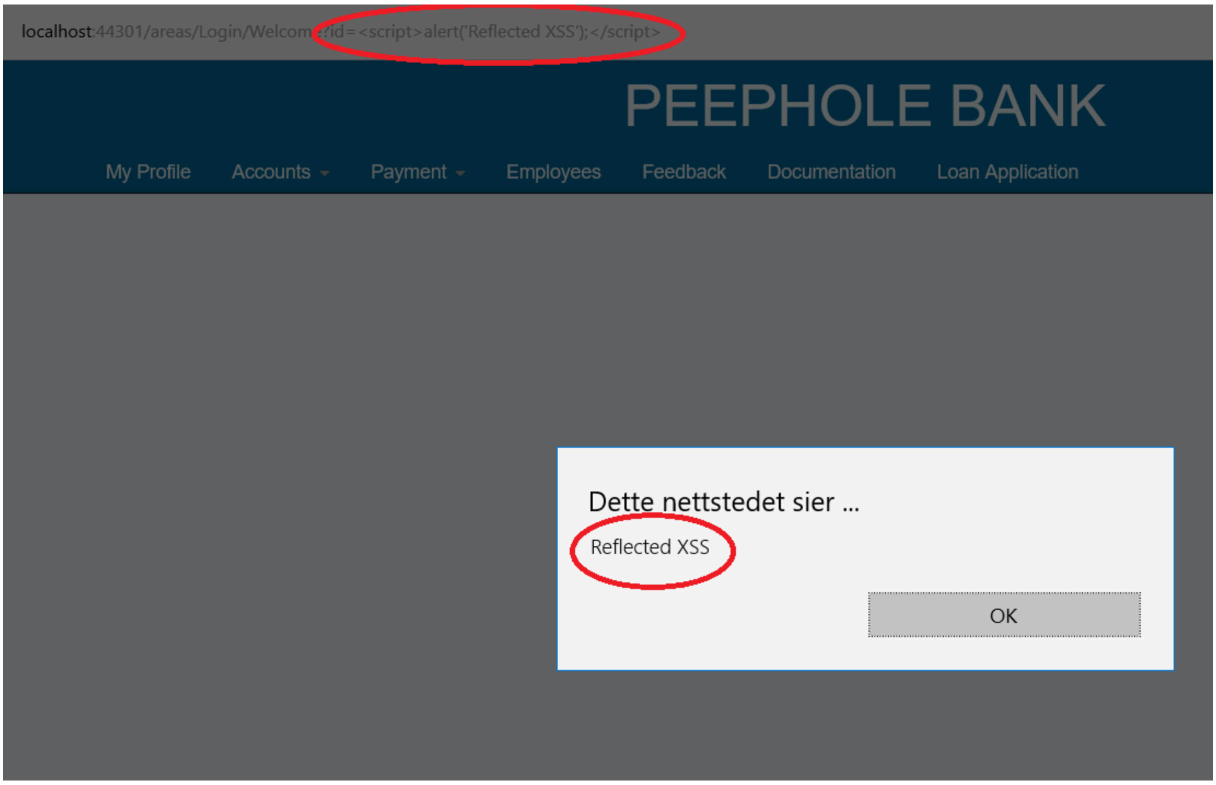


https://localhost:44301/areas/Login/Welcome?id=username

You are logged in as malina@peephole.com. Enjoy you stay!

Figure 4.12: Screenshot showing that the username is reflected back on the site through a variable in the URL

In order to exploit this vulnerability, you can put malicious code directly into the URL field. Most of the big browsers have their own filtering of such vulnerabilities by checking the URL input and filtering out specific input. Some browser however, does not filter such input. Among those are Internet Explorer and Microsoft Edge. This is why the examples are shown in Microsoft Edge (See Figure 4.13).



localhost:44301/areas/Login/Welcome?id=<script>alert('Reflected XSS');</script>

Dette nettstedet sier ...

Reflected XSS

Figure 4.13: Screenshot showing what happens when we change the variable in the URL to a script

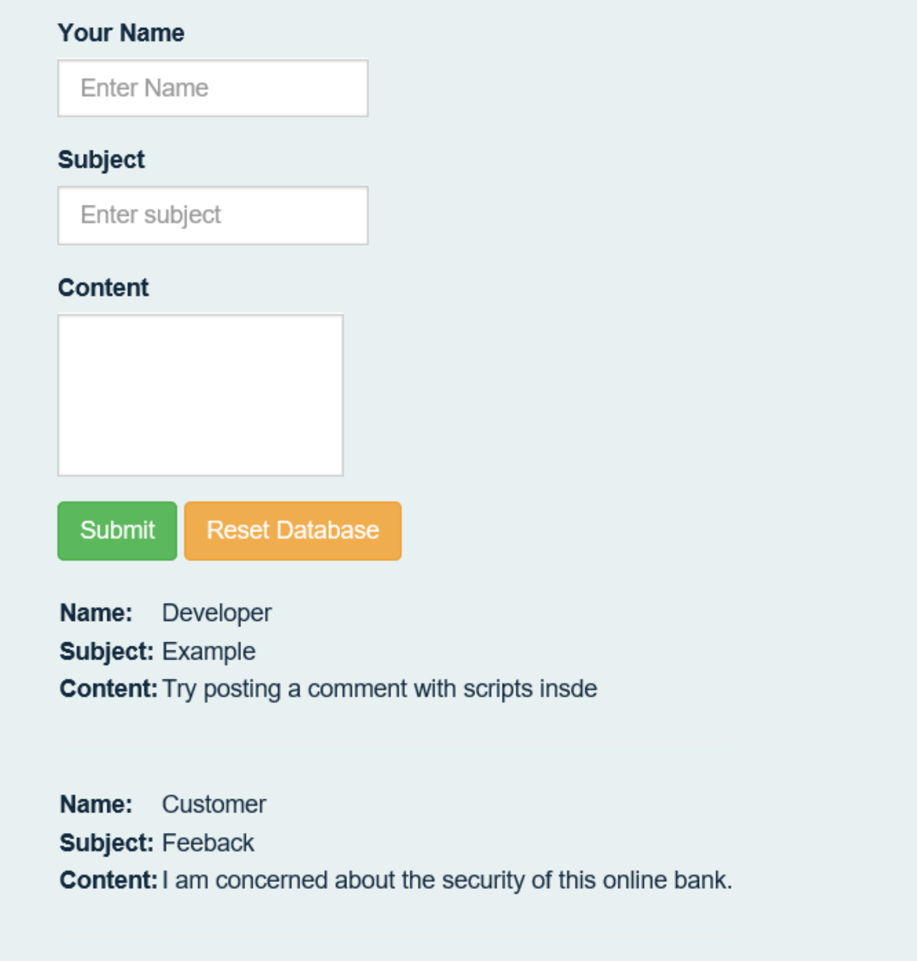
The script used above and that can be seen some of in the URL is

<script>alert('Reflected XSS');</script>

This is just a simple alert box, but with such a vulnerability you can do an enormous amount of damage. You can for example steal credentials in non-HTTPOnly cookies, send requests to a server with the user's credentials, make the user download content, display a password input, log keystrokes, and send the result to a site of your choosing. These are just a few of the things you can do with reflected XSS.

Performing the same example in for example Google Chrome would require to either disable the filtering itself. This however, does not mean it is completely secure. The input can be encoded in a way not yet known to the filter implemented in the browser for example, and the script will run. Protecting against XSS is mainly the developers' responsibility, but most of the big browsers have acknowledged the vulnerability and taken steps to help reduce the risk.

The stored XSS can be found under customer feedback in the Bank application, this is where customers and visitors can leave a comment about the bank or their experience with the bank service. The user input is handled and displayed back to the web application without any kind of validation. Here is an example of a normal comment:



Name: Developer
Subject: Example
Content: Try posting a comment with scripts insde

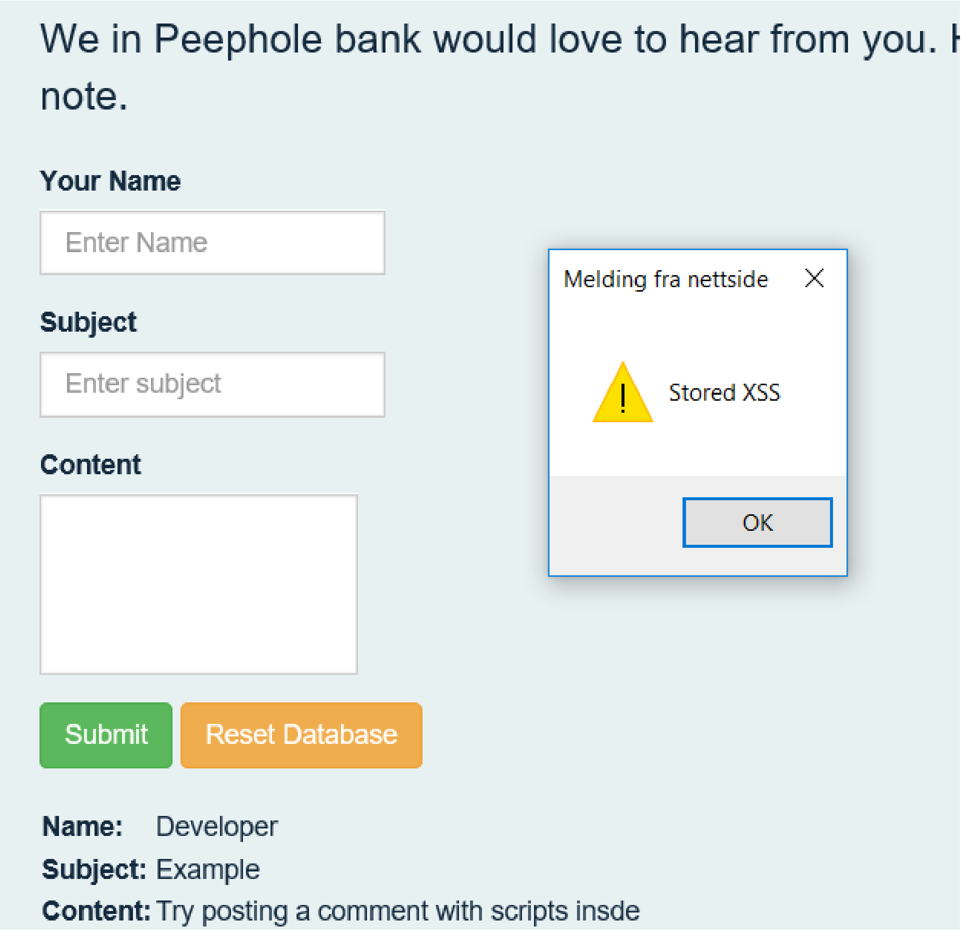
Name: Customer
Subject: Feeback
Content: I am concerned about the security of this online bank.

Figure 4.14: A normal comment form displaying comments back on the site.

Two normal comments with no harmful code (See Figure 4.14). We also see a button called "Reset Database". This is implemented so that users can try out the vulnerability without concern for breaking the application, as it will be restored to a default state once you push the reset database button.

In the next picture we will see what happens when we use the same script as we did in the reflected XSS but with different text inside, see Figure 4.15:

```
<script>alert('Stored XSS');</script>
```



We in Peephole bank would love to hear from you.
note.

Stored XSS

Name: Developer
Subject: Example
Content: Try posting a comment with scripts insde

Figure 4.15: Here we make a comment with a script. The result is that the script is run on the site

Here we can see that we get the same result as in the reflected XSS. This time the script is stored into a database and then displayed back onto the site. This means that everyone visiting this specific page after the exploit has been done successfully will run the script.

To prevent XSS in .NET, there are two main countermeasures:

- Constrain input
- Encode output

The .NET framework provides some methods to help prevent XSS, both for constraining input and for encoding output. Using the provided methods where they are needed should help secure but not guarantee protection against XSS. In ASP.NET version 4.5, Microsoft included AntiXSS based methods to help developer protect against XSS, whereas earlier versions of .NET have an AntiXSS library. Fully preventing cross site scripting is harder than it may seem. OWASP has a list of over 80 vectors that can be targeted using XSS[19].

4.3.3 Broken authentication and Session Management

Broken authentication and session management encompasses security flaws to all aspects of user authentication and session management, including but not limited to: Password strength, password storage, weak session IDs, accessible account lists on the website and browser caching.

This vulnerability is represented on several points in the application. When creating an ASP.NET application, Microsoft has a default login with strong password requirements. These requirements included special characters, letters, password length of 8+ characters and upper and lower case characters. For the sake of the application, we chose to make our own custom register and login to better show the security threat with weak passwords, insufficient hashing and storing of passwords along with weak protection. With the weak password requirements, the user can make passwords that are easily guessed as seen in this list of most common passwords of 2015 [59]. This list also indicates that there are still many sites out there with a weak password policy.

In the application the session is not deleted when closing the browser making it easy for an attacker to take over that account. The attacker simply has to open the browser to access the already logged in account. To log out the user has to use the logout button.

4.3.4 Insecure Direct Object References

Insecure Direct Object Reference is implemented, as a part of the user being able to edit their profile from “My Profile” page accessible from the main menu.

An attacker could create an account, access the edit profile page and inspect the packet sent when submitting the form. The attacker will easily spot the user ID being submitted and suspect the possibility for editing other users accounts simply by changing the ID. The packet can be replayed for ID 1, 2, 3, N, setting the password to whatever he prefers. Once the password has been changed the attacker can simply log in to the victims account and transfer the available money to a bank account of choice.

We have made available a video on YouTube that demonstrates how this could be done.[20]

4.3.5 Cross-Site Request Forgery

Cross-Site Request Forgery (CSRF) is a highly relevant vulnerability that we have implemented or rather, not added protection against, on the account money transfer form. A defence against this in .NET is a helper called "AntiForgeryToken". This helper generates a unique ID for each form, which is impossible for an attacker site to guess. This means that when an attacker site tries to submit a form with the victim’s permissions, the targeted site will see that the form comes from an untrusted source since it lacks the unique ID.

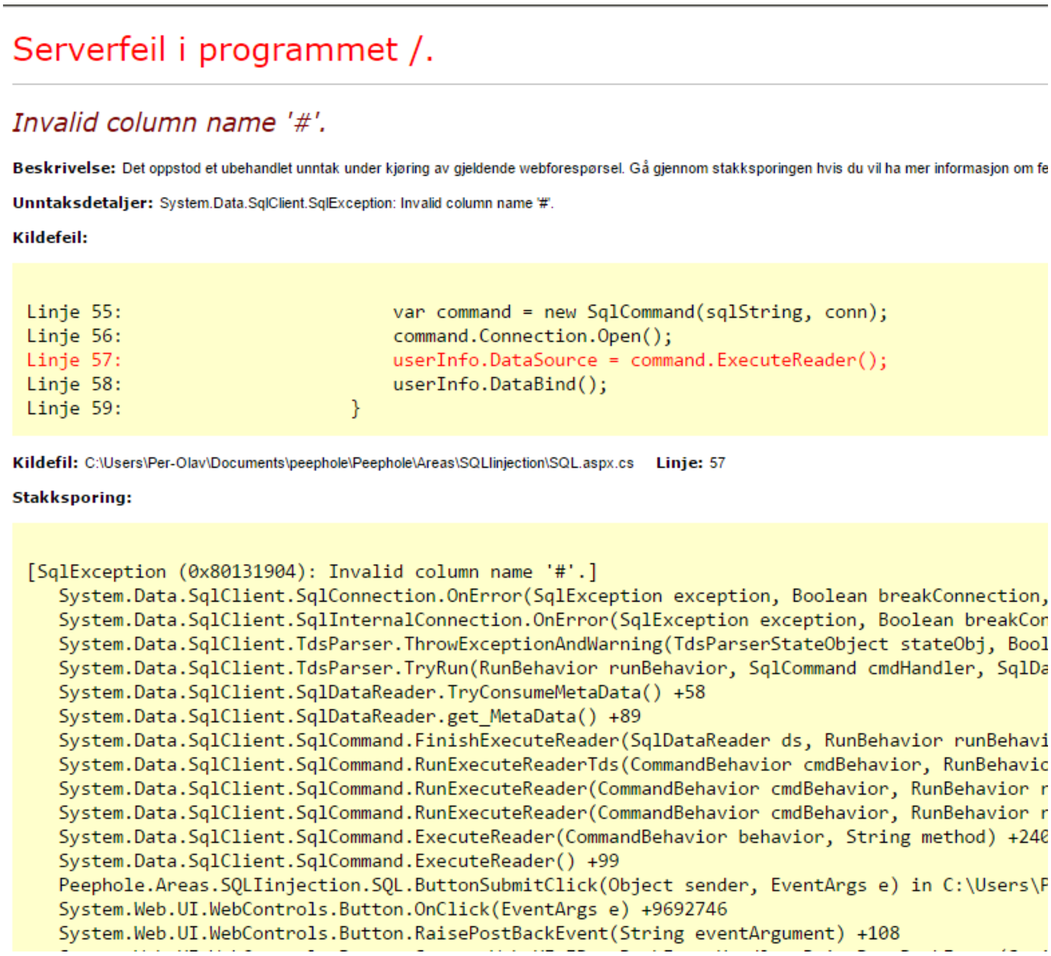
[19] https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

[20] https://youtu.be/jWGMPw2JizI?t=3m22s

Our application is vulnerable against CSRF in the Area Feedback. To simulate CSRF you need to have an attacker site, which we have not prioritized due to the restricted development time we have spent on the application.

4.3.6 Security Misconfiguration

Security misconfiguration is a term that describes when any one part of the application stack has not been hardened against possible security vulnerabilities. In OWASP Security misconfiguration is listed at number 5 of their top 10 most critical web application security flaws. When developing an application in .NET as we did, ASP.NET applications can be configured to produce debug binaries which may be extremely helpful when developing. When releasing an application, such a misconfiguration can give extremely helpful information to attackers since it displays direct information about the backend of the system. An example to this is improper error handling. When an application crashes without handling the exception, it will display a stack trace and information not meant for users. This information may prove very useful for an attacker wanting to learn more about the system. The Bank project produces an error when you search for an user and at the same time include special characters in the search term (See Figure 4.16).



Serverfeil i programmet /.

Invalid column name '#'.

Beskrivelse: Det oppstod et ubehandlet unntak under kjøring av gjeldende webforespørsel. Gå gjennom stakksporingen hvis du vil ha mer informasjon om fe

Unntaksdetaljer: System.Data.SqlClient.SqlException: Invalid column name '#'.

Kildefeil:

```
Linje 55:                          var command = new SqlCommand(sqlString, conn);
Linje 56:                          command.Connection.Open();
Linje 57:                          userInfo.DataSource = command.ExecuteReader();
Linje 58:                          userInfo.DataBind();
Linje 59:                      }
```

Kildefil: C:\Users\Per-Olav\Documents\peephole\Peephole\Areas\SQLinjection\SQL.aspx.cs **Linje:** 57

Stakksporing:

```
[SqlException (0x80131904): Invalid column name '#'.]
   System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection,
   System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakCor
   System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Bool
   System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDa
   System.Data.SqlClient.SqlDataReader.TryConsumeMetaData() +58
   System.Data.SqlClient.SqlDataReader.get_MetaData() +89
   System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavi
   System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavic
   System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior r
   System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior r
   System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior, String method) +24€
   System.Data.SqlClient.SqlCommand.ExecuteReader() +99
   Peephole.Areas.SQLInjection.SQL.ButtonSubmitClick(Object sender, EventArgs e) in C:\Users\F
   System.Web.UI.WebControls.Button.OnClick(EventArgs e) +9692746
   System.Web.UI.WebControls.Button.RaisePostBackEvent(String eventArgument) +108
```

Figure 4.16: Showing the error produced by incorrect input when searching for users

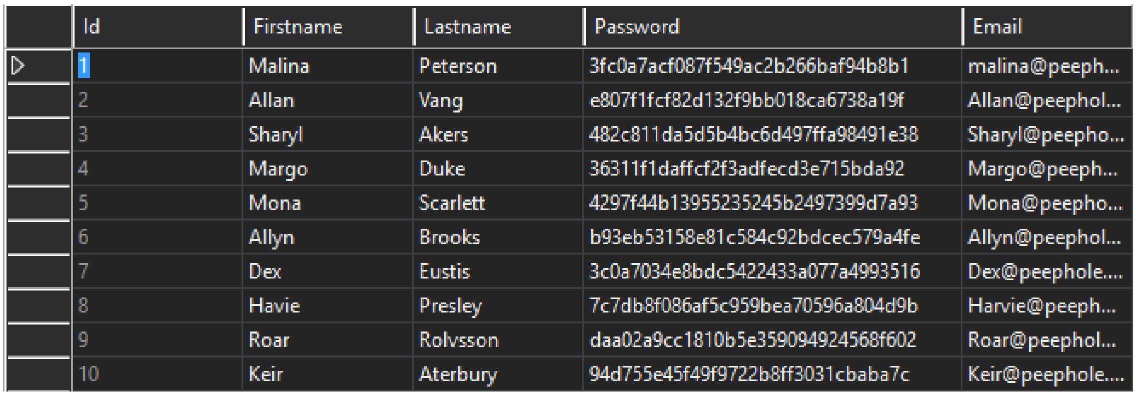
From the error produced above we can see that we got direct contact with the database in the system, which gives us a good indication that the system is vulnerable to SQL-injection.

4.3.7 Insecure Cryptographic Storage

In the project we have used MD5 as the hashing algorithm. This means that that we are using a hashing algorithm that should not be used to protect confidential information such as passwords. It should not be used because it is unsalted and it is fast to decrypt. MD5 is also vulnerable to collision, which means two inputs producing the same hash. In theory it means that if the password is “123” that gives a specific hash output, but another string, let’s say “Asjdaskdbv879” could give the same hash output and the attacker login to account that has the password “123” with “Asjdaskdbv879”.

There are several MD5 rainbow tables available online, which makes cracking MD5 hashes incredible easy. One if these are www.md5cracker.org.

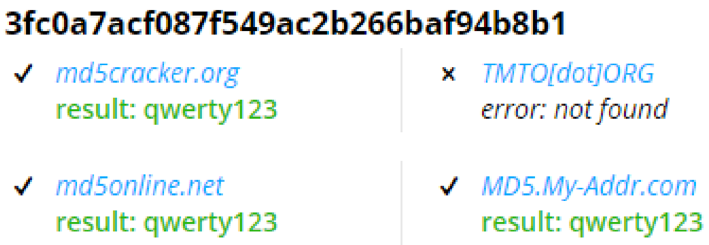
If you do an SQL injection like we have done in 4.3.2 Injection, you will get the hashed password as an output (See Figure 4.17). Insert one of those hashes into MD5cracker.org and you will have the password in clear text. After you have the password, you can proceed to logging in with the username and password.



	Id	Firstname	Lastname	Password	Email
▷	1	Malina	Peterson	3fc0a7acf087f549ac2b266baf94b8b1	malina@peeph...
	2	Allan	Vang	e807f1fcf82d132f9bb018ca6738a19f	Allan@peephol...
	3	Sharyl	Akers	482c811da5d5b4bc6d497ffa98491e38	Sharyl@peepho...
	4	Margo	Duke	36311f1daffcf2f3adfecd3e715bda92	Margo@peeph...
	5	Mona	Scarlett	4297f44b13955235245b2497399d7a93	Mona@peepho...
	6	Allyn	Brooks	b93eb53158e81c584c92bdcec579a4fe	Allyn@peephol...
	7	Dex	Eustis	3c0a7034e8bdc5422433a077a4993516	Dex@peephole....
	8	Havie	Presley	7c7db8f086af5c959bea70596a804d9b	Harvie@peeph...
	9	Roar	Rolvsson	daa02a9cc1810b5e359094924568f602	Roar@peephol...
	10	Keir	Aterbury	94d755e45f49f9722b8ff3031cbaba7c	Keir@peephole....

Figure 4.17: The passwords is stored in the database as hashed passwords. This got to be secure, right?

If we try taking the first hashed password into MD5cracker.org we will see the what the password actually is.



3fc0a7acf087f549ac2b266baf94b8b1

✓ md5cracker.org
result: qwerty123

× TMTO[dot]ORG
error: not found

✓ md5online.net
result: qwerty123

✓ MD5.My-Addr.com
result: qwerty123

Screenshot 1: The result of the MD5 crack. This took less than a second.

There are also rainbow tables for many other hashing algorithms such as SHA1 and LM hashes.

A good solution for hashing now is the SHA3, which was released in August 2015 by NIST and FIPS [60]. This is more secure because cracking it takes considerably more time and there is no collision.

Microsoft has not yet support for a SHA3 hashing functionality, but they have for SHA2-256[21].

[21] https://msdn.microsoft.com/en-us/library/system.security.cryptography.sha256%28v=vs.110%29.aspx?f=255&MSPPError=-2147217396

4.3.8 Failure to Restrict URL Access

Failure to restrict URL access is implemented several places in Peephole Bank.

My Profile

When accessing "My Profile" the user is directed to URL https://localhost:44301/CustomerInfo/Default/Edit/1

An attacker can simply change the last number in the URL, a number representing user ID, to edit some other person's account

Edit bank account

The URL presented to the user when editing a bank account is https://localhost:44301/TransferFunds/BankAccounts/Edit/3994045019

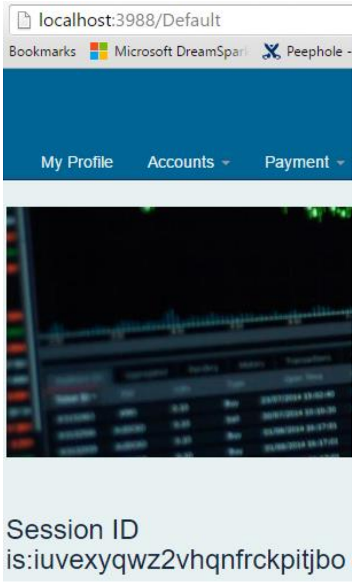
While not being the most serious vulnerability, an attacker could change the bank account name and type by guessing another person's bank account number in the URL. As all account increment by 1 from 3994.04.5000 other people's account numbers are easily discoverable and the information could be used for instance to increase the success rate of a spear phishing attack.

4.3.9 Insufficient Transport Layer Protection

SSL is only partly implemented in Peephole Bank. When the user first enters the bank the session ID is created and transmitted in cleartext over unprotected HTTP (See Figure 4.19). The transmitted information does not get encryption from HTTPS (See Figure 4.18) until the user logs in. The session ID remains the same leaving a severe risk of session hijacking.

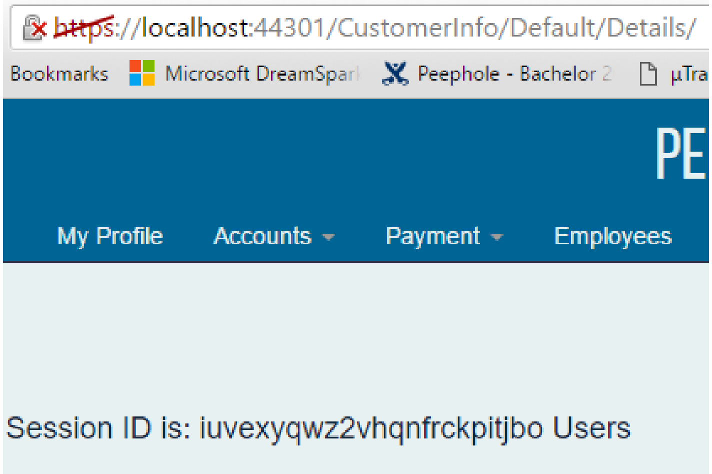
The attacker could simply capture the transmitted packets when the victim is connected to an unencrypted public Wi-Fi and get the hold of the session ID. Once the victim has authenticated against the bank the attacker holds a valid, authenticated session and can perform whatever actions the victim is authenticated to do.

Two-factor authentication or any other level of secure user authentication would not help as long as the authenticated session ID is up for a grab.



Session ID is:iuvexyqwz2vhqnfrckpitjbo

Figure 4.19: Session ID when user we have logged in, HTTP



Session ID is: iuvexyqwz2vhqnfrckpitjbo Users

Figure 4.18: Session ID when is logged in, HTTPS

4.3.10 Unvalidated Redirects and Forwards

Once the site is loaded it redirects to another site which happens almost instantly - meaning that the user most likely will not notice.

What this site does is getting a single parameter from the URL which is redirects to.
An example is:
http://localhost:3988/Redirect/Redirect/Redirect?url=http://www.ntnu.no

What this does is first accessing the Redirect-module, checking the URL for the parameter called “url” (…?url=http://www.ntnu.no) and redirect to that page. The code behind that does the redirection is:

```
string url = Request.QueryString["url"];
   Response.Redirect(url);
```

But how can this be used by an attacker’s points of view? Because the parameter is in the URL, the attacker can change this before sending it to the victim. If the victim trusts the site that redirects which in this case it would be localhost:3988, he/she probably would not check the whole URL. If the attacker changes the parameter to “http://www.malware.com” the victim most probably would not notice.

http://localhost:3988/Redirect/Redirect/Redirect?url=http://www.malware.com
From the victim’s perspective, this link is pointing towards localhost:3988 and not a malicious site.

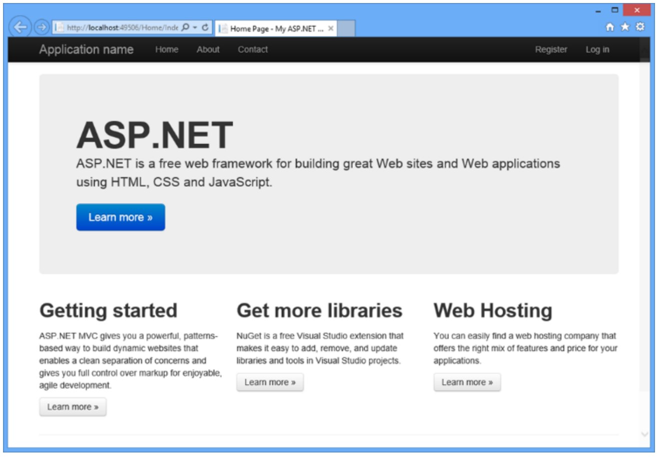
The safest way to make this non-existent is using making a redirect page for each redirect you want to do. Instead of requesting the redirecting-URL as a parameter from the URL, you will put the redirecting URL directly into the Redirect function like this:

```
Response.Redirect(“http://www.safeURL.com”);
```

By doing it this way, the attacker cannot change the URL to his advantage.

4.4 Design

During the first months of the bachelor-development-process we had the standard ASP.NET template for our application, but later we decided to implement another design.



ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

Learn more »

Getting started

ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.

Learn more »

Get more libraries

NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.

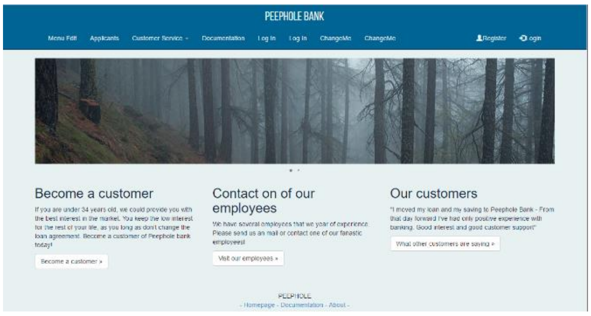
Learn more »

Web Hosting

You can easily find a web hosting company that offers the right mix of features and price for your applications.

Learn more »

Figure 4.20: How the default project looks like



Become a customer

If you are under 34 years old, we could provide you with the best interest in the market. You keep the low interest for the rest of your life, as you long as don't change the loan agreement. Become a customer of Peephole bank today!

Become a customer »

Contact on of our employees

We have several employees that we year of experience. Please send us an mail or contact one of our fanastic employees!

Visit our employees »

Our customers

"I moved my loan and my saving to Peephole Bank - From that day forward I've had only positive experience with banking. Good interest and good customer support"

What other customers are saying »

PEEPHOLE
- Homepage - Documentation - About -

Figure 4.21: How the project looks now

We did follow the topics from Jeff Johnsons book (See 2.7 Design).

The colour schemes used are retrieved from Coolors[22]. We tried a lot of different colors before we were satisfied. We tried looking at the colors our self, but we also did a small presentation to our supervisor and to other people outside of the group, in this case people in our class and people with other higher education.

When using different colors it is important to know what they mean. We had an example of this talking to our supervisor with the first color scheme as seen in Figure 4.22: Here's two different color schemes that we wanted to use, but ended up with the one the right.. The color red usually means danger [61] and our supervisor made us aware of this.

This is the reason we tried with several other colour schemes and ended up with a light-blue colour.

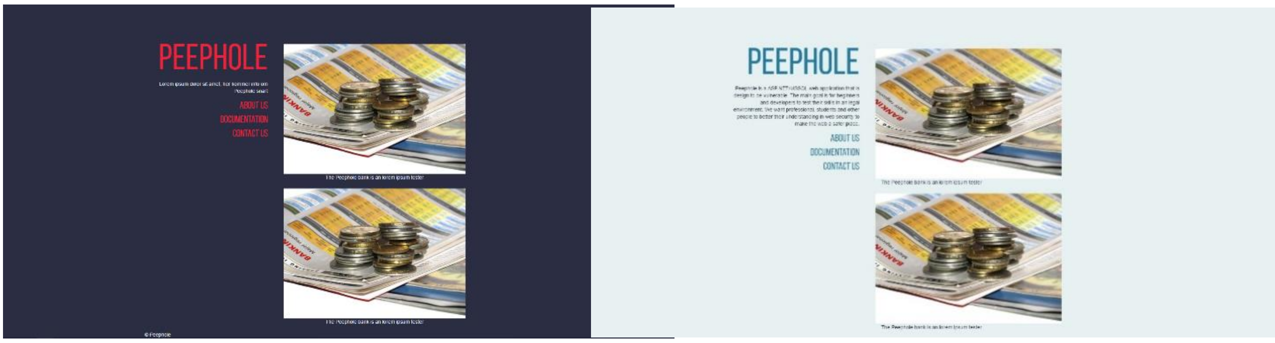
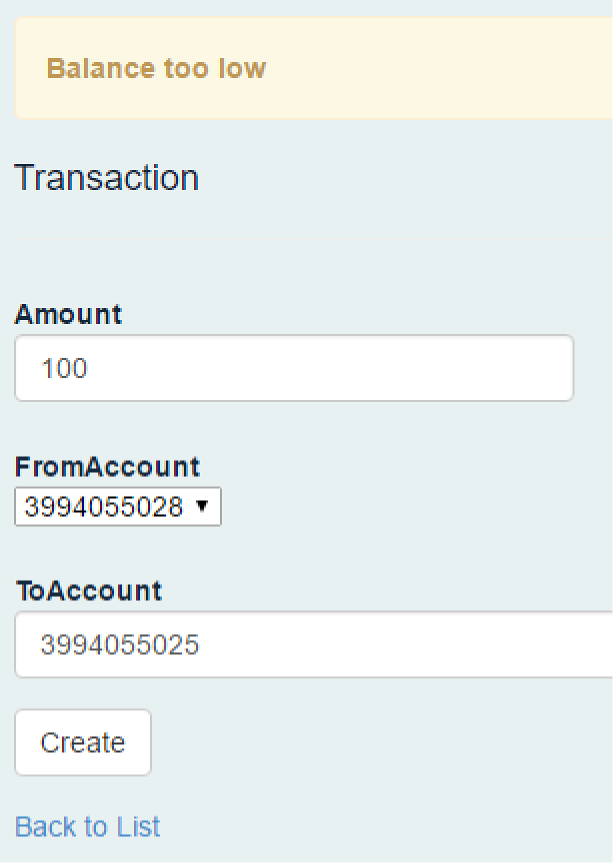

PEEPHOLE
ABOUT US
DOCUMENTATION
CONTACT US
The Peephole bank is an lorem ipsum tester
The Peephole bank is an lorem ipsum tester
© Peephole
PEEPHOLE
ABOUT US
DOCUMENTATION
CONTACT US
The Peephole bank is an lorem ipsum tester
The Peephole bank is an lorem ipsum tester

Figure 4.22: Here's two different color schemes that we wanted to use, but ended up with the one the right.

Blue is often associated with trust, loyalty, confidence, etc. [62].

[22] https://www.coolors.co



Balance too low

Transaction

Amount

100

FromAccount

3994055028 ▾

ToAccount

3994055025

Figure 4.23: Error message that appears when trying to transfer money with insufficient funds. Peephole Bank

Another example of this when transferring money from one account to another. If the balance is too low a warning occurs (see Figure 4.23). The color for this is orange. This color is usually used for warnings and less important errors. The error message shown here is the standard error-message in the bootstrap-framework.

The buttons that redirects to the application is just a big picture. This is because of the parts of Jeff Johnsons book where he mentions that humans are not made for reading. A picture is much more describing and it is easier to make a good structure with pictures instead of text-links.

If we did not have symmetry, web applications tend to be very unbalanced with all the information/text/pictures on one side. We wanted the information that is most useful such as the header and the information, to be at the top left. This is where the user looks first, so this has to be the primary focal point [63]. The content is in the secondary focal points which is below and to the right. At the pictures below, you can see the Frontpage project in our solution, which we have experimented with and changed the colors for several times.

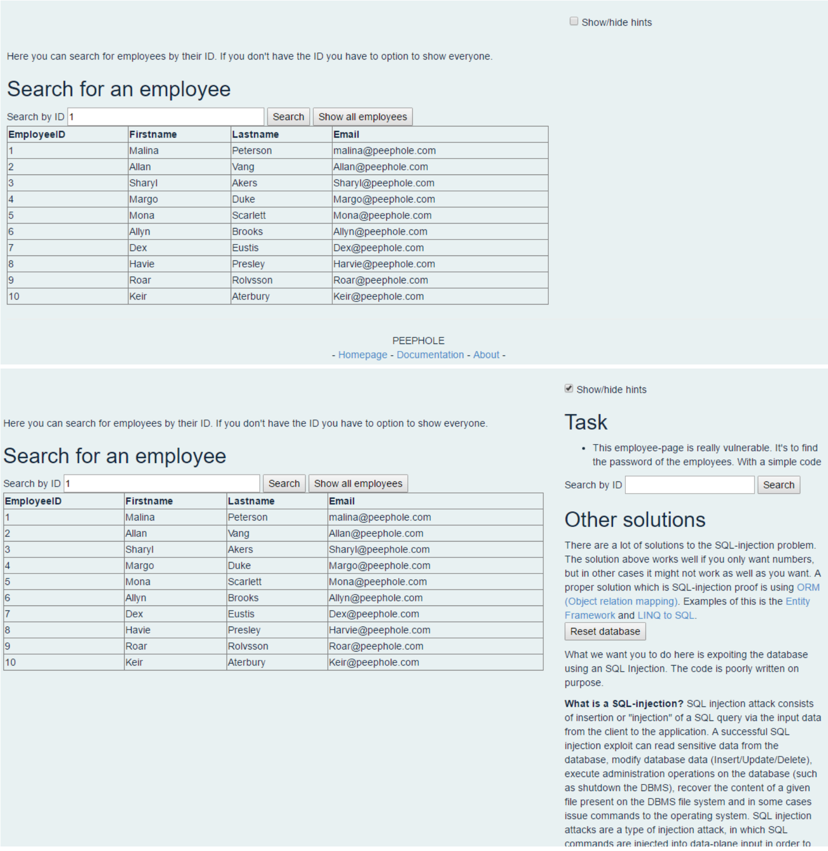
We did not use a lot of text in our Frontpage, we wanted the pictures of each project to speak for itself. We only used <h1> and <p> with some styling for the text.

The peripheral vision would first of all look at the primary focal point, which would be at the “Peephole”. The closure of the other text means that it is connected to the heading and that it is here the user naturally will look next.
Using big images on the projects with a hovering description is the biggest part of the site. These pictures are big buttons which redirects to the different projects which is in our solution.

The design is responsive, but we have not been focusing on making it responsive towards phones. This application is not supposed to be used by phones, but everything else than the menu should work fine on a mobile device.

We have tested the design against the most common resolutions [64]; 1920x1080 and 1366x768. On these resolutions the web application works without any problems. The applications work without any problems at 800x600, which is common at old projectors.
We believe our users will use the application on a laptop or desktop PC, and therefore think that the resolution will not be a problem, because most have the resolution between 1920x1080 and 1366x768.

The vulnerable sites structure is much alike the main page. The heading is in the primary focal point and the content is below and to the right. In these sites we wanted the user to be able to play around without any distraction. Because of this we made a checkbox which uses cookies to determine if you want this to be shown or if you want it to be hidden, see Figure 4.24. Inside of this you could include links to OWASP or hints for the users of the application to perform an attack in a legal environment.



☐ Show/hide hints

Here you can search for employees by their ID. If you don't have the ID you have to option to show everyone.

Search for an employee

Search by ID 1 [Search] [Show all employees]

EmployeeID	Firstname	Lastname	Email
1	Malina	Peterson	malina@peephole.com
2	Allan	Vang	Allan@peephole.com
3	Sharyl	Akers	Sharyl@peephole.com
4	Margo	Duke	Margo@peephole.com
5	Mona	Scarlett	Mona@peephole.com
6	Allyn	Brooks	Allyn@peephole.com
7	Dex	Eustis	Dex@peephole.com
8	Havie	Presley	Harvie@peephole.com
9	Roar	Rolvsson	Roar@peephole.com
10	Keir	Aterbury	Keir@peephole.com

PEEPHOLE
- Homepage - Documentation - About -

☑ Show/hide hints

Here you can search for employees by their ID. If you don't have the ID you have to option to show everyone.

Search for an employee

Search by ID 1 [Search] [Show all employees]

EmployeeID	Firstname	Lastname	Email
1	Malina	Peterson	malina@peephole.com
2	Allan	Vang	Allan@peephole.com
3	Sharyl	Akers	Sharyl@peephole.com
4	Margo	Duke	Margo@peephole.com
5	Mona	Scarlett	Mona@peephole.com
6	Allyn	Brooks	Allyn@peephole.com
7	Dex	Eustis	Dex@peephole.com
8	Havie	Presley	Harvie@peephole.com
9	Roar	Rolvsson	Roar@peephole.com
10	Keir	Aterbury	Keir@peephole.com

Task

- This employee-page is really vulnerable. It's to find the password of the employees. With a simple code

Search by ID [Search]

Other solutions

There are a lot of solutions to the SQL-injection problem. The solution above works well if you only want numbers, but in other cases it might not work as well as you want. A proper solution which is SQL-injection proof is using ORM (Object relation mapping). Examples of this is the Entity Framework and LINQ to SQL.

[Reset database]

What we want you to do here is expoiting the database using an SQL Injection. The code is poorly written on purpose.

What is a SQL-injection? SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to

Figure 4.24: Hidden content. How a module looks with and without the hidden-content menu

4.5 Database design

The Peephole Bank project has a simplistic database design, only the most necessary in order to make the system run and demonstrate the vulnerabilities.

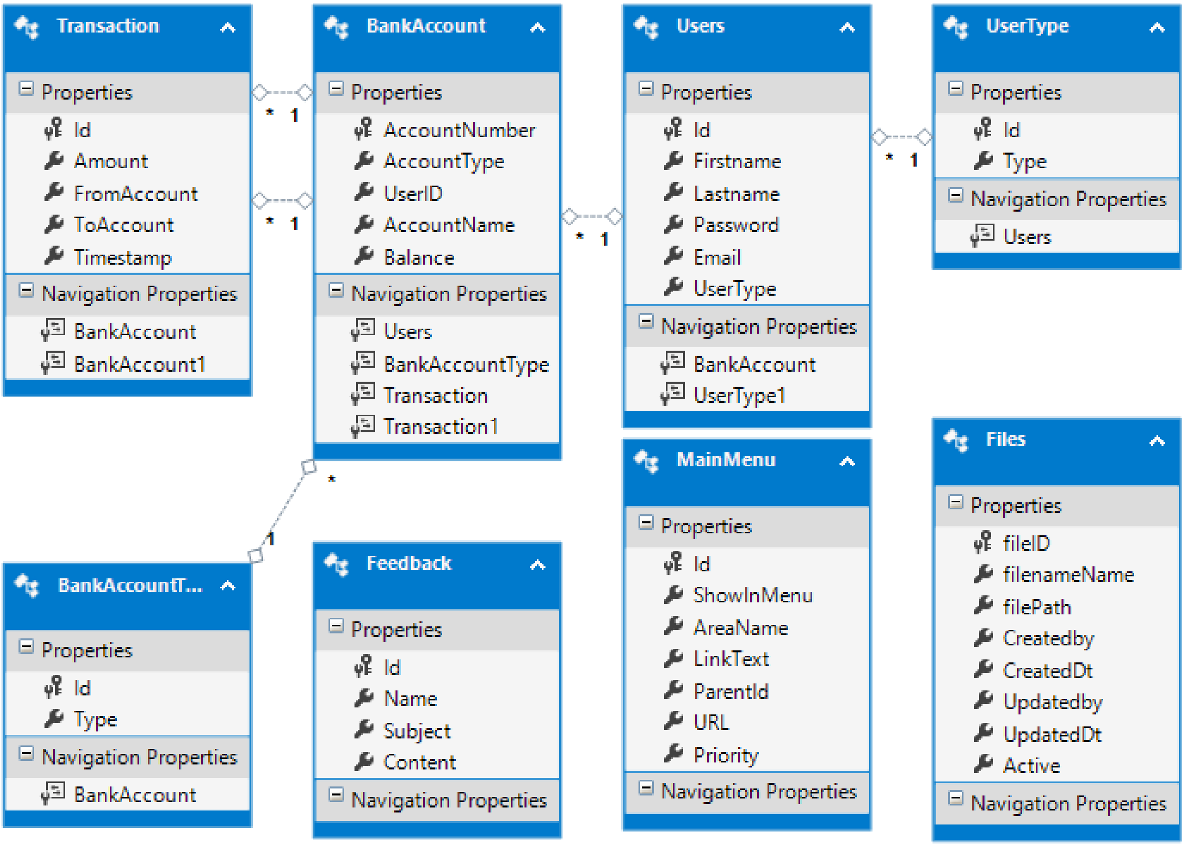


Figure 4.25: Database Table Overview

The main tables are Users, BankAccount, Transaction, MainMenu, Feedback and Files (See Figure 4.25).

Users

This table stores users of all types, types as defined in table “UserType”. Password is stored as a MD5 hash.

BankAccount

Bankaccounts are stored as an auto-incrementing account number from 3994.04.5000 with an optional account name.

Transaction

Stores all the transactions generated when a user makes a payment.

MainMenu

Stores all the menu items for Peephole Bank. Each area that wants to be listed in the menu simply has to add

```
AreasHandling.insertInMenu(context.AreaName);
```

into its AreaRegistration file.

Feedback
Comments from users is stored in this table.

Files
When a customer searches for a loan, attachment information is stored in this table.

Table details are attached (See 8.8 Attachment 8: Database tables).

4.6 Menu

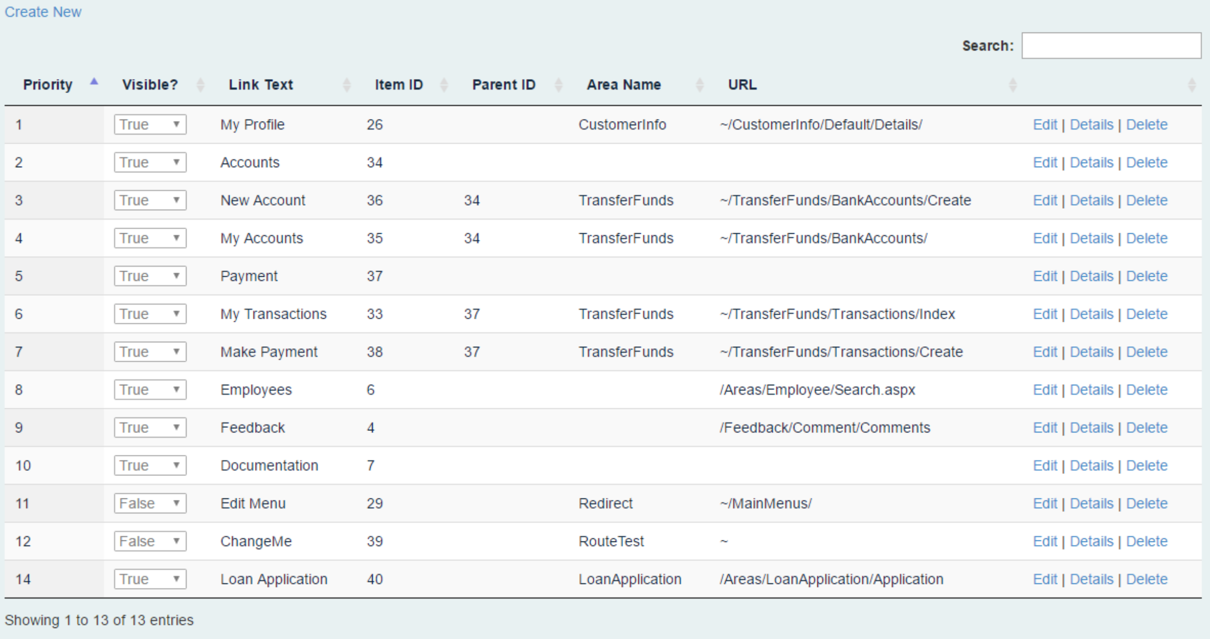
The menu is generated from table MainMenu in the Peephole database. Any new module/Area that is to be included in the menu simply has to add one line into its AreaRegistration file;

```
AreasHandling.insertInMenu(context.AreaName);
```

When the application is executed each Area is scanned and the AreaRegistration file within the Area is executed, ensuring that the Area is indeed included in the menu.

The method *insertInMenu* checks if the Area name is already present in database and adds it if not.

Once added to the database table, the item can be further edited from the *menu configuration page* [23]. For each menu item you can change its visibility, display name, display order, parent/child relationship, link reference and more.



Create New

Search:

Priority	Visible?	Link Text	Item ID	Parent ID	Area Name	URL	
1	True	My Profile	26		CustomerInfo	~/CustomerInfo/Default/Details/	Edit | Details | Delete
2	True	Accounts	34				Edit | Details | Delete
3	True	New Account	36	34	TransferFunds	~/TransferFunds/BankAccounts/Create	Edit | Details | Delete
4	True	My Accounts	35	34	TransferFunds	~/TransferFunds/BankAccounts/	Edit | Details | Delete
5	True	Payment	37				Edit | Details | Delete
6	True	My Transactions	33	37	TransferFunds	~/TransferFunds/Transactions/Index	Edit | Details | Delete
7	True	Make Payment	38	37	TransferFunds	~/TransferFunds/Transactions/Create	Edit | Details | Delete
8	True	Employees	6			/Areas/Employee/Search.aspx	Edit | Details | Delete
9	True	Feedback	4			/Feedback/Comment/Comments	Edit | Details | Delete
10	True	Documentation	7				Edit | Details | Delete
11	False	Edit Menu	29		Redirect	~/MainMenus/	Edit | Details | Delete
12	False	ChangeMe	39		RouteTest	~	Edit | Details | Delete
14	True	Loan Application	40		LoanApplication	/Areas/LoanApplication/Application	Edit | Details | Delete

Showing 1 to 13 of 13 entries

Screenshot 3: Main Menu Configuration Page

An asp:menu controller generates the actual menu. This controller is natively incompatible with Bootstrap UI framework which is used for the rest of the project. As a mean to overcome this we have implemented a workaround developed by Jeremy Knight. [65]

[23] http://localhost:3988/MainMenus/Index

4.7 User stories

The user stories are fictive stories created a short story on how he thought a user or a developer would advance the Peephole application.

4.7.1 The student

The student discovers there is a new tool for learning information security, Peephole. The student clones the GitHub repository and install the project as described in the documentation. When the application is ready, the student can start exploring the Peephole Bank and try finding vulnerabilities. The student may find some, depending on how good he/she is. The student wants to find them all and look in the documentation on how to exploit the rest of them. He/she look at the Source code to see why it is exploitable. Since this student is going to create a new ASP.NET application in the future, he/she knowns what to not do when creating a search-field, a feedback section or any other functionality.

4.7.2 The developer

The experienced developer finds out there is a new tool for learning information security. The developer clones the GitHub repository and install the project as described in the documentation or just simply open it in Visual Studio. He tries finding all the vulnerabilities, but he feels like there is missing important vulnerabilities. He remembers seeing how to create a new project in the documentation and starts developing his own project and modules which he calls Peephole Hospital, which focuses on securing confidential personal information. He thinks this is so good we want it to be part of the official Peephole. He contacts the creators of Peephole and let review the project.

4.8 Expert feedback

As mentioned in 3.1.6 Testing and debugging, we requested some experts to test our application in order to get feedback on what is good and bad, and also to have other people look at the application with a fresh pair of eyes.

The first one to test the application was our assigner and the other one is a master student which studies information and computer security in China.

The overall feedback was good with a few minor bugs which was not intentionally bugs, and other issues we needed to fix. The version they got is not the product we delivered, but a similar unfinished product. We know that at least one of the test subjects had a lot of experience with other tools like the already existing solutions out there. We did not know what experience the other tester had when we sent him the application. He answered he had some experience with WebGoat, but not any of the other applications.

Both of the testers said the application was good and they both found several vulnerabilities.

Vulnerabilities found by the Master student:

- Reflected XSS
- Stored XSS
- SQL Injection
- Unrestricted file
- Modify any user password
- View any bank account details
- Modify any bank account – Accounttype and Accountname
- Cross Site Request Forgery

We presented the application to our assigner and he gave us some feedback of what his impression was. Since he is a security expert and have tested most of similar application and used them in classroom situations his opinion was crucial. This is the feedback he gave us:

> Teaching and learning web security is a difficult matter, since applications deployed in the real world cannot be used as training platforms.
>
> Vulnerable web applications designed for web security training are great resources that help closing this gap. However, these kinds of applications tend to be predictable, very unrealistic, and they have important limitations.
>
> For example, vulnerable web applications available these days are very focused on open source technologies, such as PHP and MySQL.
> This limitation gives teachers and students a very narrow set of possibilities, with a short-sighted view of web security and platform-specific vulnerabilities.
>
> Peephole is an excellent project that takes web security training to a next level, since it tackles many of the unsolved challenges seen until now.
> This application has been built with current .NET technology, which offers an environment not common for web security training, but which has a very important market share in web application development.
>
> Peephole also provides an attractive banking environment, which allows students to explore and test their web security skills, without expecting predictable results.
>
> Finally, it is important to mention that Peephole offers a very exciting development platform, which may allow other students to go deeper in web security.
> This could be shaped in the development of new vulnerable modules, extending existent functionality, or even learning how to implement recommended solutions for fixing vulnerabilities from a programming perspective.
>
> CEO & Lead IT Security Consultant
> Encripto AS - Information Security.

We have included all the feedback as an attachment, Attachment 9: Feedback.

4.9 Comparison to existing solutions

Table comparing similar applications with Peephole.

🟩 Yes 🟥 No ▨ N/A

Application name Feature	Peephole	DVWA	WebGoat	WebGoat.N ET	Hera	Hack.ME	NOWASP	Webscarab	bWAPP
Open source	🟩	🟩	🟩	🟩	🟥	🟥	🟩	🟥	🟩
Free	🟩	🟩	🟩	🟩	🟥	🟩	🟩	🟩	🟩
Updated within last year	🟩	🟩	🟩	🟥	▨	▨	🟥	🟥	🟥
English language	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Does not require internet connection	🟩	🟩	🟩	🟩	🟥	🟥	🟩	▨	🟩
Real-life application	🟩	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥
Includes OWASP Top 10	🟩	▨	🟩	🟩	▨	▨	▨	🟥	🟩
Over 50 vulnerab.	🟥	🟥	🟩	🟥	🟩	🟩	🟥	▨	🟩
Modular	🟩	🟥	🟩	🟥	🟥	🟥	🟥	🟥	▨
Guidance	🟩	🟩	🟩	🟩	🟩	🟥	🟥	🟩	🟩
Written in	ASP.NET C# MSSQL	PHP MySQL	Java	ASP.NET C# MSSQL	▨	▨	PHP JS	▨	PHP MySQL

Table 2: Differences between Peephole and other existing solutions.

As we can see in Table 2, the biggest differences are that Peephole is a Real-Life application and that it is using the ASP.NET framework. It does not have over 50 vulnerabilities for now, but in the future it could be one of the biggest. Since Peephole support multiple projects the possibilities are unlimited. The application that has the most in common with Peephole is WebGoat, but in reality there are big differences between them. It is not a real-life application and it is written in Java, which is a much smaller platform than ASP.NET [14].

4.10 Results published in an academic conference

We had a meeting with our supervisor at the end of the project where we discussed and came to the conclusion that it would be good to publish a scientific article at NIKT2016.

NIKT2016 is a Norwegian IKT-conference in Bergen[24]. This includes NISK (Norwegian Information Security Conference) which we have submitted an article too.

The content of this article is the most important information that you will find in this thesis. We focus on what we have made and that we want to share Peephole with the security community.

Accepted articles are to be presented at the Norwegian Information Security Conference held in Bergen this year from November 28th to November 30th. The deadline for delivery is August 25th, and the feedback whether that article gets accepted or not is given on October 1st. This means that we do not know the outcome of the article before the bachelor thesis has been delivered and presented.

(See 8.5 Attachment 5: NISK article)

[24] http://nikt2016.hib.no/nb/

5 DISCUSSION

In the discussion we attempt to review the achieved results, implementation of management techniques, the use of tools and challenges we have met. The discussion is based on the statements made earlier together with our own impressions and thoughts.

During the discussion we sometimes compare our application, Peephole, with the most common existing application, DVWA and WebGoat.

5.1 Planning and project management

When writing the pre-project report in the spring 2016 we had already started the early stages of planning and development of our solution. At that time, we had a few meetings with our supervisor and set a product backlog for the features we should implement. Also, we defined the level of functionality that should be implemented in order to consider the application as complete. When comparing the backlog (Attachment 6: Backlog from JIRA) and the state of our project, we see that the requirements set have been met at a satisfactory level.

At the point of writing the pre-project report we were using Visual Studio Team Services for project planning and management as well as using Scrum methodology. Scrum was a natural because of the high level of uncertainties in the project, we needed a methodology that could make the development process as flexible as possible. Using Visual Studio Team Services was also a natural decision because of its close relation with Visual Studio, which was the only real alternative for the solution. As the documentation part of the project became more complex and we experienced problem with VSTS Git solution, we decided to try out Confluence and JIRA after recommendation from NTNU.

5.1.1 Methodology

Initially we intended to use Scrum only as our development methodology. After some time, we analysed how we actually managed our work, discovering that we were utilizing parts of other methodologies as well.

From the extreme programming methodology (See 0

Extreme programming) we used pair programming. This is the method used when two people is using one computer to write code together. We thought this was a great way of doing things, because we could learn more about coding than if we just wrote it our self. The best way of coding could be the way someone else does it. Using this method makes it easier to discover errors and mistakes.

From Lean Software Development (See 2.2.3 Lean software development) we have used several of the principles listed. The first one is “eliminate waste”, which is an important principle when developing software. We reviewed the code several times trying to make the code as short as possible. As we were building Peephole on an ASP.NET template there was a lot of code and functionality we did not utilize. An example of this was the default login functionality from Microsoft. We made our own vulnerable version leaving the default login as excess data.

“Amplify Learning” is another principle. We had to learn a lot of new technologies and software at the beginning and throughout the project. By sharing relevant and structured feedback with each other we managed to increase the learning level. During the first couple of weeks we also watched YouTube videos and tutorials that explained subjects we knew little about at that time, like ASP.NET, MVC, Webforms and more.

The principle “decide as late as possible” were truly valuable to us. By focusing on early prototypes to decide on the fundamental structure and modular solution saved us quite some time compared to developing bigger parts of the project before running into problems forcing us to start all over again.

The fourth principle “deliver as fast as possible” came natural to us as a part of using Scrum. We usually had sprints for 1-2 weeks and at the end of each sprint we had a new deliverable.

The communication within the group has been used to empower the team, the fifth principle. Whenever someone made good progress, fixed a bug or some other positive action, it was noticed and applauded by the other members of the group. Both our supervisor and assigner were also open-handed with structured and positive feedback. Support and help in difficult situations is indeed paying off, it gave us motivation boost and made us work even harder.

The sixth principle is “build integrity in”. Our assigner did have a tight time schedule, but still managed to make it clear what elements should be included in the final product. Combining these required elements with regular meetings and feedback from our supervisor made it clear along the way that we were on the right track.

The seventh and last principle is “see the whole” also came natural as a result of Scrum. When deciding on tasks to put in the sprint backlog we discussed each backlog item, breaking them down to smaller parts.

We have had both good and bad experiences with Scrum (See 2.2.1 Scrum). One of the big surprises was as we mentioned planning poker, this was a good way to estimate how long time a backlog item would take to finish. We did not use it as often as we wanted, because at first we thought it was time consuming. As should also have been stricter with the meetings we should have had. We probably had daily meeting about half the time, sometimes we arrived at different times during the morning and because of that we did not have a proper meeting at the beginning of the day.

The sprint retrospective and review meeting we kind of did with our supervisor each Friday. During these meeting we would tell how things went and what we did good/bad. We somehow also did the backlog refinement during this meeting as well. This worked well, but we could have had separated these meeting for a better understanding and completion of Scrum.

The sprint planning meetings we did properly. Every Friday after the meeting with Girts, we planned the next sprint; which backlog items, who was doing what and estimated a time for that task. Because of this, we were already planned and ready for the next week.

5.1.2 Communication

During the project communicating has been of great importance, both within the group and to our supervisor and assigner. The group communication has been very good as we have been working in closely at the same computer lab at school each day between 08:00 and 16:00. During the time when we were all present at school, communication was usually orally. When we were at home communication was usually performed with Facebook Messenger. Communication with our supervisor was mostly performed by mail and the regular Friday meetings. Mail and meetings was also how we communicated with our assigner.

There have been situations where we should have contacted teachers for help, instead of using an excessive amount of time trying to figure it out by ourselves. Besides this, getting the necessary feedback was easy and quickly once requested.

5.2 Modularity

From the very beginning, having the solution modular and easy to extend was one of the primary goals. We are satisfied with the modularity as it turned out, as the application extendible in two ways; both as a new project in solution and as Areas/modules within a project. With a new project the developers could decide even on which .NET version to use and core references to include, they are totally free to create whatever project they would like.

We believe our solution is a great foundation for developers to extent the application, where they are free to choose between Webforms and MVC and still use the default layout template. A downside with the shared layout feature is a bug where the namespace in the Site.master designer file keeps getting autocorrected each time Site.master is edited. To minimize the impact of this, the Bank project is designed in such a way that editing the Site.master file should be unnecessary, for instance the developer do not have to edit a menu within Site.master, the menu is generated from database instead.

DVWA and WebGoat does not, as far as we have experienced, have the same level of modularity as Peephole. It is easier to start extending Peephole compared to similar applications and the Peephole solution enables us to even take one of the other existing applications and simply include it.

We have released the project on a git-repository where people can clone it in order to run or extend it.

5.3 Software used

As mentioned in materials (See 3.3.1 Software used) we started using VSTS, but it did not function as good as expected. After recommendation from NTNU Aalesund, we started using Atlassian software.

We have been using Confluence pretty much every day. We used it for deciding on big decisions, creating meeting notes, saving notes, saving links and adding other content we need to include in the thesis. This has been an extremely useful tool for us. An alternative application for this could have been Microsoft Word or a note sharing software like Keep, but then we would not have the great structure that Confluence is providing.

We used JIRA to manage our backlog, specifications can be found in attachment 6 (Attachment 6: Backlog from JIRA). We added backlog features to a sprint and assigned each work item to a specific person. From this the work item changed status from “To do” to “In Progress” and then to “Done” once finished. JIRA supplied a great overview of what each of us was doing. Periodically we were not to active updating the system and we found out we were instantly less productive. Using JIRA as it is intended is a big benefit when using Scrum.

The Backlog from JIRA is available as Attachment 6. (See 8.6 Attachment 6: Backlog from JIRA)

For version control and backup of the source code we used Bitbucket and SourceTree. Both of these application are very powerful tools for managing code. At first, we had to do a lot of testing to get the pushing and pulling right and to get to know the new software. We had to spend some time learning how they both worked and to avoid conflicts when merging, pulling and pushing code. We had a few incidents when pulling changes to one of the computers, but we always managed to recover from it. JIRA had the possibility to make a graph (See Figure 5.1) to show all activity from commits we had.

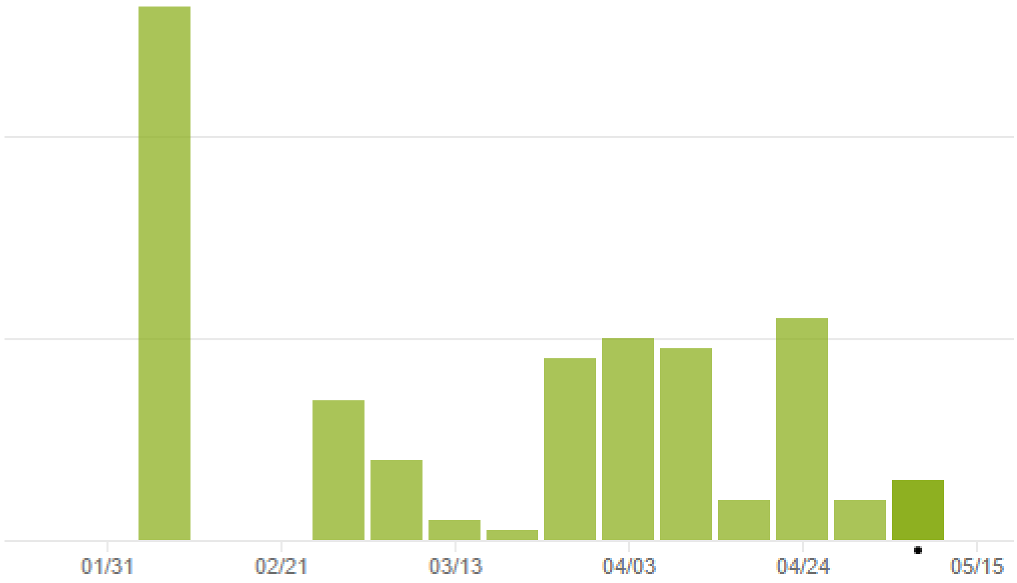


Figure 5.1: All the commits we had during the project. The lowest horizontal line is 20 commits.

Using a version control system is crucial when developing software in our opinion!

5.4 Demand

During the course "Information security", one of us asked Juan J. Güelfo at Encripto AS if he knew about a potential bachelor project. He expressed the need of a teaching application for web application security written in .NET. He also wanted an application that was more real-life-like than the existing comparable applications.

> I'm missing vulnerable application based learning, which is actually a «real application". By this I mean something that goes a step further than "click Module 1 – Injection" Thinking a bit more like the "Hackme Casino" or "Hackme Bank" application. These are two very old applications that follow a more real scenario principle[25].
>
> *- Email from Juan J. Güelfo*

This requirement has made influenced the structure and design of Peephole Bank. Area names could have been named by their vulnerability like "injection", instead we name them by function like "Employee". The naming convention should also be effectuated for the menu link labels.

Encripto AS has written an article "3 Reasons to Invest in IT security during Recession" [26]. The article describes why a company should invest in security also during a recession and is highly applicable for the situation in Norway right now. Investment cutbacks makes the security weaker which increases the success rate for cyber criminals to have a successful attack on a company. Our application could potentially help companies, employees and students get a better understanding of securing web applications, free of charge.

5.5 Programming languages

Developing the application in ASP.NET was one of the requirements. The technology posed some issues for us as we did not have any prior experience with it.

We found some books and tutorials to learn C#, but quickly discovered that a more suitable approach was learning-by-doing. For instance, we needed to have some cookie-functionality. The answer does reside in books as well, but it was much quicker to find and use tutorials from Codecademy[27] and KhanAcademy[28]. Both of these sites have good tutorials which could help a beginner or an intermediate to grow their knowledge.

We expected it to be hard work learning the technology, and it was even harder than expected. We encountered problems that were very difficult to fix by reading documentation and looking at online guides.

5.6 Testing

Code and run was heavily used as testing method during the development. By this the developer had the best prerequisite to discover and fix bugs as he was familiar with the most recent code

[25] http://www.mcafee.com/us/downloads/free-tools/hacme-casino.aspx
http://www.mcafee.com/us/downloads/free-tools/hacme-bank.aspx

[26] https://www.encripto.no/nb/2016/03/reasons-to-invest-in-it-security/

[27] https://www.codecademy.com/learn

[28] https://www.khanacademy.org/

changes to the code. In addition to this we worked on separate branches for each function. When a function was finalized and tested by one of us, the branch was merged and pushed to Bitbucket. From Bitbucket the merge was pulled and tested / reviewed once again by another member of the group. The combination of code-and-run with breakpoints, Git and branching has been a great way for us to continually test the code and resolve bugs.

From the user testing (4.8 Expert feedback) we received some valuable feedback. The testing was done by experts who has more experience with information security than anyone else we know. With this testing we got the first real test of the system as a whole. They had some constructive feedback for items we could change for the better. Some of the suggested changes has been implemented, others are listed as suggested improvements to be. Having the external review of the system has been of great value as having a fresh set of eyes often discovers issues blind to us working on the project daily.

We also had meeting with our supervisor every week. Each Friday we showed him what had been done and how it was done. The design of code has been influenced by these meetings.

5.7 Design

When making the design, we wanted it to have a modern look. We got several opinions on the colors, we tried a few, and ended up with a light blue scheme. We also wanted the Bank to look similar to a real bank webapplication. For inspiration, we visited several Norwegian bank sites and a few international banks. We discovered what the banks had in common, what their content was and got an idea of how the project should be designed. When we had finished coding the design it was already late in the semester. Together with our supervisor we decided that the design was sufficient and that we should focus more on the thesis itself and preparing the application for distribution.

Even though we have used limited resources on the design, we are pretty happy with it. Especially considering the amount of time into making it, and even made it possible to share between MVC and Webforms. We used what we had learned during an earlier subject with HMI and how people react to different object and colors.

> When I first see the Peephole I feel the UI is very comfortable and many modules, 8 out of 10.
> - *Huiyun Ge*

Compared to the other comparable applications available, we believe the design feels more modern. For instance, DVWA and WebGoat has an older and more traditional design.

As a consequence of the Bank being real-life-like, it could be hard for a user to understand what to do at certain pages as the vulnerabilities are “hidden”. We believe the hidden content section at selected pages could be a great way to inform the users of the application and convey vulnerability information for a specific page. With these hidden content sections, the creator of the module can explain the context for the page and convey tips for exploitation. These sections are not limited and can display HTML/JavaScript/jQuery and any other typical web technology.

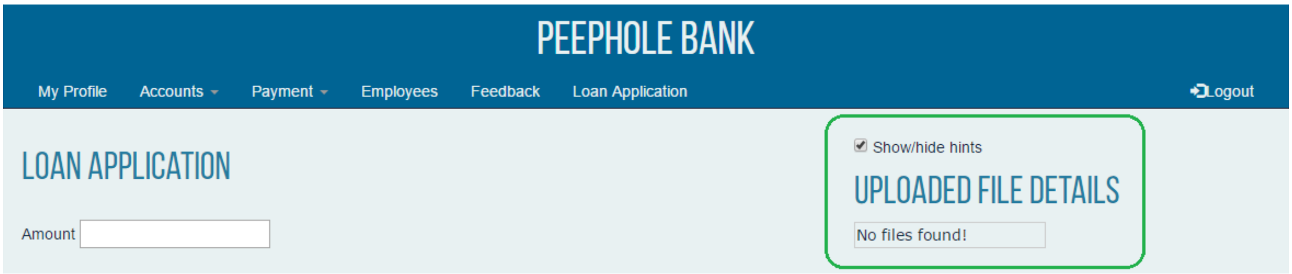


Figure 5.2: Hidden content section

All images used in the Peephole solution is licensed as CC0[29] and is retrieved from http://negativespace.co.

[29] https://creativecommons.org/about/cc0/

6 CONCLUSION

This bachelor project was all about discovering how to create the most comprehensive, useful and over time current information security teaching tool for ASP.NET. To achieve this, we had to consider distribution, architecture, extensibility and the implemented vulnerabilities.

To distribute the application, we are using GitHub. GitHub makes it easy for developers, security testers and other people like teachers and student to clone the repository and start using the application.

We ended up with a structure that supported both Webforms and MVC. This combination makes the application targets a broad selection of developers no matter which programming model they favor. The application is extendible with additional projects as well as the vulnerable Bank application is easily extendible with Areas, making a great foundation for the most comprehensive information security teaching tool available. Vulnerabilities implemented in Peephole Bank includes the entire OWASP Top 10, as the assigner suggested. Other vulnerabilities might be best demonstrated under totally different circumstances, with the support for multiple projects this can be done easily.

From the finalized Peephole solution, we can conclude that the application meets the specifications given in the product backlog (See Attachment 6: Backlog from JIRA), made by us in cooperation with our supervisor and our assigner.

The final version of Peephole has been tested externally by our assigner and a Master Student in information and computer security and received great feedback from them both.

> Peephole is an excellent project that takes web security training to a next level, since it tackles many of the unsolved challenges seen until now.
> This application has been built with current .NET technology, which offers an environment not common for web security training, but which has a very important market share in web application development.
> *- Juan J. Güelfo*

During the project we have acquired a better understanding of how to succeed with development in teams and knowledge about the .NET framework. We have also achieved a better understanding for system development in general considering meetings, analyses, designing, implementation, testing and documenting.

The core experiences we have gained regarding project management is:

- Give high priority for the first couple of meetings to get a good overview of how the application should work and what the customer (assigner) has in mind.
- Make extensive research on the fundamentals before starting the actual coding. For this project this was mainly; How could we best make the application modular and which view engine(s) should we support?
- Divide big tasks into smaller tasks to simplify and make the task more manageable.
- Communication is essential. Use communication well to plan and empower the team.

7 REFERENCES

[1] Microsoft. [Online]. Available: https://msdn.microsoft.com/en-us/library/kx37x362.aspx.

[2] Microsoft. [Online]. Available: https://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.110).aspx.

[3] Wikipedia, "Agile software development," [Online]. Available: https://en.wikipedia.org/wiki/Agile_software_development.

[4] M. Piattini and F. J. Pino, "IEEE Explore," 2011. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6063151.

[5] Unknown. [Online]. Available: https://en.wikipedia.org/wiki/Integrated_development_environment.

[6] Wikipedia, "Microsoft," [Online]. Available: http://docs.asp.net/en/latest/client-side/bootstrap.html.

[7] Microsoft, "ASP.NET MVH Overview," [Online]. Available: https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx. [Accessed 22 01 2016].

[8] F. A. University, "Modular programming," FAMU-FSU College of engineering, [Online]. Available: https://www.eng.fsu.edu/~dommelen/courses/cpm/notes/progreq/node2.html. [Accessed 01 2016].

[9] Atlassian, "Jira," [Online]. Available: https://www.atlassian.com/software/jira.

[10] Atlassian, "Confluence," [Online]. Available: https://www.atlassian.com/software/confluence.

[11] Atlassian, "Bitbucket," [Online]. Available: https://www.atlassian.com/software/bitbucket.

[12] Microsoft, "ASP.NET Master Pages," [Online]. Available: https://msdn.microsoft.com/en-us/library/wtxbf3hh.aspx.

[13] RandomStorm, "DVWA," [Online]. Available: http://www.dvwa.co.uk/.

[14] W3Techs, "All server side programming languages," [Online]. Available: http://w3techs.com/technologies/overview/programming_language/all.

[15] APM, "What is project management," [Online]. Available: https://www.apm.org.uk/WhatIsPM. [Accessed 14 05 2016].

[16] A. Styve, "Projektledelse i IT prosjekter," 2016.

[17] Agile-Tools, "Scrum," [Online]. Available: http://www.agile-tools.net/scrum.asp.

[18] Wikipedia, "Extreme programming," [Online]. Available: https://en.wikipedia.org/wiki/Extreme_programming.

[19] A. Esley. [Online]. Available: http://200.17.137.109:8081/novobsi/Members/teresa/optativa-fabrica-de-sw-organizacoes-ageis/artigos/Addison%20Wesley%20-%20Lean%20Software%20Development%20-%20An%20Agile%20Toolkit.pdf. [Accessed 14 05 2016].

[20] RandomStorm, "DVWA github," 18 02 2016. [Online]. Available: https://github.com/RandomStorm/DVWA.

[21] WebGoat, “GitHub,” WebGoat, [Online]. Available: https://github.com/WebGoat/WebGoat. [Accessed 18 02 2016].

[22] OWASP, “OWASP WebGoat.NET,” [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_WebGoat.NET#tab=License. [Accessed 23 02 2016].

[23] E. l. security, “Hera Labs,” E learn security, [Online]. Available: https://www.elearnsecurity.com/virtual-labs/hera/. [Accessed 16 02 22].

[24] eLearnSecurity, “Hack.me,” [Online]. Available: https://hack.me/. [Accessed 2016 02 22].

[25] OWASP, “OWASP Mutillidae,” [Online]. Available: https://www.owasp.org/index.php/OWASP_Mutillidae_2_Project. [Accessed 22 02 2016].

[26] OWASP, “OWASP webscarab,” [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project#tab=Project_About. [Accessed 23 02 16].

[27] T. Hunt, OWASP Top 10 for .NET developers, Microsoft, 2011.

[28] OWASP. [Online]. Available: https://www.owasp.org/index.php/SQL_Injection.

[29] OWASP. [Online]. Available: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

[30] OWASP. [Online]. Available: https://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management.

[31] OWASP, “Broken authentication and session management,” [Online]. Available: https://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management.

[32] OWASP. [Online]. Available: https://www.owasp.org/index.php/Top_10_2010-A4-Insecure_Direct_Object_References.

[33] OWASP. [Online]. Available: https://www.owasp.org/index.php/Top_10_2013-A5-Security_Misconfiguration.

[34] OWASP. [Online]. Available: https://www.owasp.org/index.php/Top_10_2010-A7-Insecure_Cryptographic_Storage.

[35] OWASP. [Online]. Available: https://www.owasp.org/index.php/Top_10_2010-A8-Failure_to_Restrict_URL_Access.

[36] OWASP. [Online]. Available: https://www.owasp.org/index.php/Top_10_2010-A8-Failure_to_Restrict_URL_Access.

[37] OWASP. [Online]. Available: https://www.owasp.org/index.php/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.

[38] Cornell University - Department of Computer Science, [Online]. Available: http://www.cs.cornell.edu/courses/cs3110/2011sp/lectures/lec07-modules/modules.htm. [Accessed 23 05 2016].

[39] Microsoft, “Model-View-Controller,” [Online]. Available: https://msdn.microsoft.com/en-us/library/ff649643.aspx?f=255&MSPPError=-2147217396. [Accessed 23 05 2016].

[40] J. Johnson, Designing with the mind in mind.

[41] K. H. a. M. S. R. Thomas J. Madden, Managing Images in Different Cultures: A Cross-National Study of Color Meanings and Preferences, American Marketing Assocation, 2000.

[42] Microsoft, “Developer Network,” [Online]. Available: https://msdn.microsoft.com/nb-no/default.aspx.

[43] K. M.-Ø. a. N. C. Haugen, “Combining Estimates with Planning Poker - An Empirical Study,” IEEE.

[44] J. Nielsen, “Why You Only Need to Test with 5 Users,” NNgroup, 2000.

[45] M. Mujic, “Ledere og ansattes oppfattelse av flat struktur og lederrollen,” Universitet i Bergen, Bergen, 2013.

[46] I. O. f. Standardization, “ISO 9000-3:1991, Quality Management and Quality Assurace Standards - Part 3: Guidelines for the application of ISO 9001 to the Development, Supply and Maintenance of Software,” ISO, Geneva, Switzerland, 1991.

[47] T. Barker, “Writing Software Documentation: A Task-Oriented Approach,” 2003.

[48] D. Parnas, “Precise documentation: the key to better software,” in *The Future of Software engineering*, Zurich, 2011, pp. 125-148.

[49] P. L. A. T. H. v. V. Wei Ding, “Knowledge-based approaches in software documentation,” *Information and software technology*, pp. 4-5, 2014.

[50] Wikipedia, “HTML,” [Online]. Available: https://en.wikipedia.org/wiki/HTML.

[51] W3Techs. [Online]. Available: http://w3techs.com/technologies/overview/javascript_library/all. [Accessed 21 05 2016].

[52] Atlassian, “Atlassian Software,” [Online]. Available: https://www.atlassian.com/software. [Accessed 24 02 2016].

[53] J. Fisher, “Utilizing Atlassian Jira for large-scale software development management,” Lawrence Livermore National Laboratory, San Fransico, 2013.

[54] Atlassian, “Agile tools for software teams,” [Online]. Available: https://www.atlassian.com/software/jira/agile. [Accessed 24 02 2016].

[55] A. SourceTree, “SourceTree features,” [Online]. Available: https://www.atlassian.com/software/sourcetree/overview/. [Accessed 25 02 2016].

[56] A. G. a. M. A. Cusumano, “Platform leadership - How Intel, Microsoft, and Cisco Drive Industry Innovation,” in *Platform leadership - How Intel, Microsoft, and Cisco Drive Industry Innovation*, HBSP.

[57] Microsoft, “Introduction to ASP.NET Core 1,” [Online]. Available: http://docs.asp.net/en/latest/conceptual-overview/aspnet.html.

[58] O. Organization, “OWASP,” [Online]. Available: https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting. [Accessed 01 04 2016].

[59] Computer world, [Online]. Available: http://www.computerworld.com/article/3024404/security/worst-most-common-passwords-for-the-last-5-years.html. [Accessed 2016].

[60] NIST. [Online]. Available: http://www.nist.gov/itl/csd/sha-100212.cfm. [Accessed 22 05 2016].

[61] C. teacher, “Enotes,” [Online]. Available: http://www.enotes.com/homework-help/why-red-colour-used-danger-signals-there-any-191865. [Accessed 19 05 2016].

[62] Color Wheel Pro, “Color wheel,” [Online]. Available: http://www.color-wheel-pro.com/pics/gsinfographic1(color)large.jpg.

[63] P. Laja, “ConverionXML.com,” [Online]. Available: http://blog.eyequant.com/blog/2014/01/15/the-3-most-surprising-insights-from-a-200-website-eye-tracking-study. [Accessed 21 04 2016].

[64] W3Schools, “Screen Resolution Statistics,” 2016. [Online]. Available: http://www.w3schools.com/browsers/browsers_display.asp.

[65] J. Knight. [Online]. Available: https://jeremyknight.wordpress.com/2014/02/25/asp-net-forms-bootstrap-menu-control/.

[66] G. Rodriguez, “IEEE Explore,” 05 01 2016. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7372484.

[67] Microsoft. [Online]. Available: https://www.visualstudio.com/products/visual-studio-enterprise-vs.

[68] OWASP, “OWASP Top 10 Project,” 2016. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

[69] OWASP, “The MySpace Worm,” [Online]. Available: http://lists.owasp.org/pipermail/owasp-losangeles/2008-December/000037.html. [Accessed 02 05 2016].

8 ATTACHMENTS

The format on the attachment may be slight off.

8.1 Attachment 1: Pre-project report

TITLE:
Pre-project Report

CANDIDATE NUMBER(S):

020161, 130204 and 130201

DATE:	SUBJECT ID:	SUBJECT:	DOCUMENT ACCESS:
11.01.16	IE303612	Bachelor Thesis	

STUDY:	PAGES/ATTACHMENT:	BIBL. NR:
Computer engineer, bachelor	19/1	18

ASSIGNER(S)/SUPERVISOR(S):
Encripto AS
Girts Strazdins
Kjell Inge Tomren

TASK/ABSTRACT:

This paper gives an introduction to our bachelor thesis with word definitions used throughout the paper. It shows how we structure the project development as well as the agreements we have made with the assigner and supervisors. The start of this paper gives a description of what the project is all about with a summary of issues, objectives and purposes of the project. We give a small risk assessment on what might not go as planned and how likely this might be. The equipment required to complete the bachelor thesis is also listed. At the end of the document you will find a list of resources used when writing this document and links to the origin of the information.

Denne oppgaven er en eksamensbesvarelse utført av student(er) ved NTNU i Ålesund.

INTRODUCTION

In the spring semester 2015 we had the subject ID302809 Information Security and all of the members of the group enjoyed this subject. The teacher mentioned missing an equivalent to the information security teaching tool DVWA, but written in ASP.NET/C# for the windows platform. We thought it would be a good bachelor thesis to develop this teaching tool as we were all interested in the subject and that it would be a unique and useful product. In addition, we will learn ASP.NET/C# which has been left out of the 3-year bachelor study program we have followed.

The application we are to develop is a vulnerable ASP.NET/C# web application. Its main purpose is to provide an environment for individuals, developers and security professionals to test and hone their information security skills. This could help web developers to better understand the process of securing a web application and aid teachers/students to teach/learn web application security in a class room environment.

The problem that needs to be addressed is how do we create an e-learning tool for information security that remains relevant to a threat picture that is rapidly changing?

1 DEFINITIONS

C# [1] is a modern programming language created by Microsoft. C# was developed to be used in the .NET framework and it is based on Java and C++.

> C# is designed to be simple, powerful, type-safe, and object-oriented. – Microsoft

.NET [2] is a cross-platform software framework developed by Microsoft which is mainly used on the Windows platform. The advantage of .NET is the possibility to use several programming languages with the CLR (Common language Runtime) and that it is using software virtualization.

Scrum [4] is an agile software development methodology for managing product development. Scrums goal is having a development team work as a unit to reach a common goal. This includes physical meetings at a physical or close online collaboration of all team members, as well as daily face-to-face communication across the project.
An important part of the Scrum methodology is that the customers can change their minds about what they want and what they need, though this could make it more unpredictable for the developers.

Visual studio team services (or VSTS) is a cloud collaboration tools made by Microsoft. This tool works with most IDEs and includes version control, tools for agile collaboration, performance testing and integration with other big cloud applications. This tool supports many programming languages, including C# which is the main programming language we are using.

An **integrated development environment** (**IDE**) [5] is a software application that provides a development facility to programmers developing applications and other software. The IDE have several components like the source code editor, debugging tools and build automation tools. Most of the IDEs also have something called intelligent code completion, which helps the programmer completing code snippets.

Bootstrap [6] is a collection of tools for creating websites and web applications that is free and open source. It includes HTML and CSS design templates for most interface components, as well as optional JavaScript extensions. It is a tool meant to make development of dynamic websites and/or applications a lot easier. Bootstrap is a front end framework.

The **MVC (Model view controller)** [7] is a popular structural file architecture framework/pattern. This framework is used for building web applications containing:

- The model: the application core (usually contains some kind of database)

The view: responsible of displaying the data

- The controller: responsible for the inputs, which means it is reads the data from view, controls the user input and then send the data to the model and then is arranging all that data to the user of the application.

Modular programming [8] is dividing the application into separate sections where these sections each have their own functions and routines. These sections have minimal to none interaction between them making it easier to read, understand and extend the application.

2 PROJECT ORGANISATION

Project group

The project group working on this bachelor thesis consists of three students as listed below.

Student numbers
020161
130204
130201

Table 3: Student numbers for all group members

Project group tasks – organization

As we are a small group we also have a small organization. We follow Scrum / XP for software development and our organization will be structured like illustrated below.

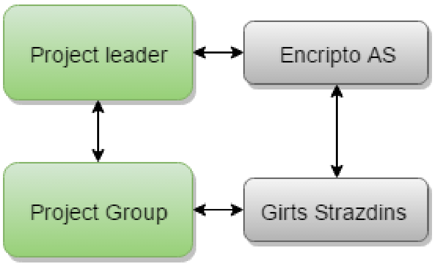


Figure 1: Group structure. Green boxes indicate that the function is managed by members of the group, grey boxes are external resources.

Every third week we are alternating on having the role of the project leader. The project leader is also a working member of the project group.

Project Leader Tasks

In addition to the tasks given for the project group, the leader is responsible for;

- Prepare for weekly meetings with Girts Strazdins, make an agenda and ensure that all items on the agenda is discussed at this meeting.
- Lead the weekly Friday Scrum meetings, update the backlog and define the next sprint
- Coordinating the continuous workflow
- Gather issues discovered throughout the development and arrange for discussions and solutions.
- Communication with Encripto AS

Project Group Tasks

Members of the project group are responsible of attending in weekly meetings with Girts Strazdins. Following this weekly meeting is the mandatory Sprint meeting where the current and future sprint activities are to be discussed and planned.

During the sprint all members are to pay attention to the sprint tasks listed at the task board and work diligently towards the completion of these tasks.

All issues discovered throughout the development are to be shared with the project leader.

Management group

Our supervisor is Girts Strazdins and Kjell Inge Tomren.
The contact person for the assigner company Encripto AS is Juan J. Güelfo.

3 AGREEMENTS

Agreements with assigner

The contact person at Encripto AS, Juan J. Güelfo, has very limited time to spend attending in meetings and discussions. As a result, we have agreed to try to minimize his involvement during the development. His involvement might be limited to have a couple of meetings throughout the semester. For the "everyday" discussions we are to utilize resources available at the University.

Workplace and resources

Formally we have no regular facility to work during this semester, but we have an understanding that the lab "Lovelace" is pretty much available to us during this semester. Regarding resources, we don't need anything other than ourselves, computers and an Internet connection to do the development. We have spoken with several teachers before and during the thesis and they are available for assistance.

"Lovelace" might not always be available and we have agreed that the library or a room at "Fagskolen" might be an OK alternative. The latter has projector available great for watching online tutorials and informative videos.

Project rules – cooperation rules – attitudes

Early in the process we made an agreement of cooperation, which is a set of rules each member of the group is to follow. The agreement also specifies when we should arrive at school, how long we should be "at work" and other formalities.

This document is found under the attachments section.

4 PROJECT DESCRIPTION

Issues – objective - purpose

The main issues with this bachelor thesis will be how to structure the modular system in such a way that it is easy for external developers to make extensions to the application. By making the application in ASP.NET/C#, compared to DVWA and other PHP/MySQL applications, we include a whole new group of developers that could contribute and keep the application up to date and relevant.

The main goal of the application is to make a unique education tool that can be used not only by teachers teaching students, but also companies coursing employees and even individuals wanting to learn more about how vulnerabilities work and how to fix them. The application aims to show developers and programmers current vulnerabilities, information about these vulnerabilities and teach how to avoid making their own work vulnerable.

None of us have any previous experience with C#. A major task in the beginning of the project is therefore to learn about and understand C# and MVC. On the positive side Microsoft has good documentation on their site about the syntax and general structure of C#. We have also found physical books, digital books and online tutorials which will help us getting a better understanding of C#. Our supervisor knows C# and he told us to mail or ask him if we needed any help.

Requirements for solution or project results - specification

Our main requirement for a completed project is to have a complete, modular main application and some modules implemented and working. This means that there will be a working application at the end of the bachelor thesis that can be further extended by modules developed by community contributors at later times. One of the requirements for a complete project is having a responsive design that will adjust to the different screen sizes out there. This will be done utilizing Bootstrap, which is automatically included in the project when creating an ASP.NET web application within Visual Studio. Bootstrap helps managing content in such a way that even on different screen sizes, the content on the page will be displayed correctly.

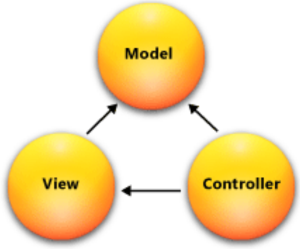


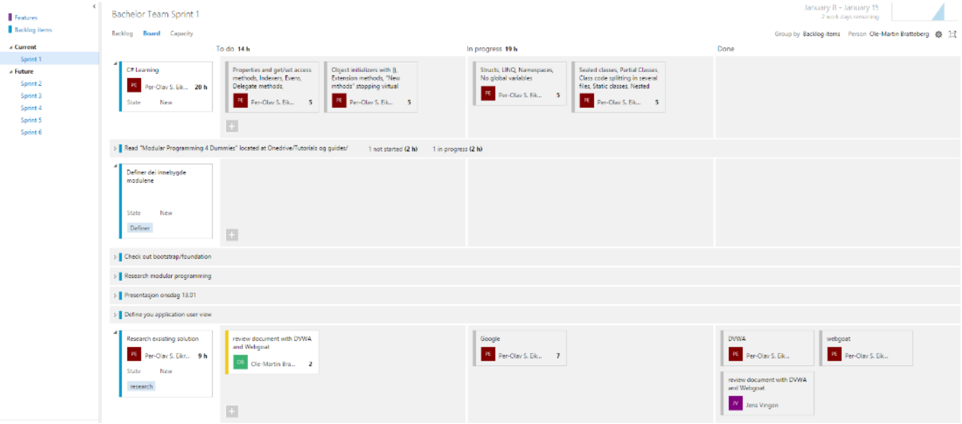
Figure 2: How the MVC structural patterns looks like

In ASP.NET there is also a structural file architecture named MVC. This is the most common architecture when building web applications. We were not familiar with this kind of file architecture from previous projects. After some research we found that it suits our needs for structuring our modular based application and it also works well with ASP.NET / Visual Studio.

Planned methods for development

During this project we will be using Microsoft Visual Studio Team Services, which is a cloud-powered collaboration tool that works great with Visual Studio (which is the IDE we are using). Team Services includes Version Control, tools for agile development methods and much more.

We are using the Scrum methodology within Team Services and this makes planning collaborative work very easy since we have a visual board where every task is viewed.



Features
Backlog items
Current
Sprint 1
Future
Sprint 2
Sprint 3
Sprint 4
Sprint 5
Sprint 6

Bachelor Team Sprint 1
Backlog Board Capacity
January 8 - January 15
2 work days remaining
Group by Backlog items Person Ole-Martin Bratteberg

To do 14 h | In progress 19 h | Done

C# Learning
PE Per-Olav S. Eik... 20 h
State New

Properties and get/set access methods, Indexers, Evens, Delegate methods,
PE Per-Olav S. Eik... 5

Object initializers with {}, Extension methods, "New mthods" stopping virtual
PE Per-Olav S. Eik... 5

Structs, LINQ, Namespaces, No global variables
PE Per-Olav S. Eik... 5

Sealed classes, Partial Classes, Class code splitting in several files, Static classes, Nested
PE Per-Olav S. Eik... 5

Read "Modular Programming 4 Dummies" located at Onedrive/Tutorials og guides/ 1 not started (2 h) 1 in progress (2 h)

Definer dei innebygde modulene
State New
Definer

Check out bootstrap/foundation

Research modular programming

Presentasjon onsdag 13.01

Define you application user view

Research existing solution
PE Per-Olav S. Eikr... 9 h
State New
research

review document with DVWA and Webgoat
OB Ole-Martin Bra... 2

Google
PE Per-Olav S. Eik... 7

DVWA
PE Per-Olav S. Eik...

webgoat
PE Per-Olav S. Eik...

review document with DVWA and Webgoat
JV Jens Vingen

Screenshot 4: Overview of the first sprint we had in out project. You can see how we use SCUM in Visual Studio Team Services.

A study published by IEEE (Institute of Electrical and Electronics Engineers) shows that students maximize their performance when using an agile methodology like Scrum [66]. We have used Scrum earlier in our Bachelor program. Our own experiences together with the IEEE study made it clear for us to use this methodology.

Collecting information – performed and planned

Before we began development of the project, we searched for existing and similar solutions. This was done to see if our proposed solution was already developed by someone else, and if so, how we would make our solution better. During this search we found a few solutions that were similar in some ways: DVWA (Damn Vulnerable Web App), WebGoat, Hera Lab and some more. After finding these somewhat similar solutions we began looking at the differences from their solution and our planned solution. A common factor for most existing solutions was their usage of MySQL and PHP and only a few of them were modular. This concludes that we target a different set of possible users, namely the developers using .NET and C#.

APPLICATION	SIMILARITY	DIFFERENCE TO OURS
DVWA	• Used for teaching information security • Web application • Runs on virtual machine • Free	• Written in PHP/MySQL • Has little to no guidance • Lacking user feedback • Not modular • Targets different audience.

WEBGOAT	• Vulnerable application • Teaching web application security • Runs on VM • Free	• Is platform independent. • Not written in C# / .NET • Not modular • Targets different audience.
HERA LAB	• Vulnerable application • Tool for learning web application security • Runs on VM	• Expensive • Runs on eLearnSecuritys own VM
HACK.ME	• Vulnerable application • Community can build extensions to the program • For education purposes • Free	• Only in MySQL / PHP • Targets different audience.
NOWASP	• Vulnerable application • Free	1. Written in PHP / MySQL 2. Not modular 3. Targets different audience
BWAPP	• Vulnerable web application • Free • For educational purposes	• Written in PHP/ MySQL • Not modular • Targeting different audience

Risk assessment

Since the group has limited resources and experience, there is a moderate risk of not getting the whole project completed in time. This may vary on what difficulties we meet on the way and how we are able to solve them. If we should come to the point where we see no way of completing the project, we will in cooperation with our supervisor limit the functionalities of the application. In the worst case scenario where we are unable to develop a working main application, we are to write a detailed report of why we got in such a situation.

One of the persons in the group could get sick resulting in the workload becoming unbalanced. Since we are using the Scrum methodology, we have a strict *Available Capacity/Remaining work balance for each sprint*. In case of sickness we would probably have to move tasks from one sprint to the next one.

We are highly dependent of our laptops as this is where we do all the work. The Visual studio team service and Office 365 is doing the backup, so we do not have to worry about losing any code due to a hard drive crashing or similar. If a computer goes down or missing, we would have to find a replacement and reconfigure the development environment.

If a computer gets damaged or has to be repaired for some reason the warranty makes the retailers responsible of providing a backup.

Main activities

NUMBER	MAIN ACTIVITY	RESPONSIBILITY	RESOURCES/TIME
A1	Research	All	2 weeks
A2	Learn C#	All	Learn while we go
A3	Learning ASP.NET	All	4 weeks
A3.1	Learning the structure	All	1 week
A3.2	Learning how MVC works	All	1 week
A4	Designing the website	Unknown (1 person)	3 weeks
A4.1	Learning how bootstrap is working and how we could benefit from this solution	Unknown (1 person)	1 week
A5	Implementing webpage on the OS	Unknown (1 person)	2 weeks
A6	Creating the modules Each module has a different difficulty level and therefore it is hard to estimate the time	Unknown (2 persons)	6 weeks
A7	Testing and fixing bugs	Unknown (2 persons)	3 weeks
A8	Planning and meetings	All	4 hours a week

Table: List of main activities in the project

Work schedule

Overall plan

NUMBER

A1	This activity Girts came up with. He told us to start researching other solutions that's already out there. This is because we were not going to make a project that already out there. The results can be found under 5.4.
A2	We have been working this Java in this study, but none of us have any experience with C#. Since .NET and ASP.NET have to be written in C#, this is something all of us needs and want to learn.
A3	Learning ASP.NET is important because this is what we would use to make the website application engine.
A4	The website needs to have a design. One of us (with help from the other on the group) needs to make a web design for our application.
A5	When we are all done we need to implement the web application on an external OS.
A6	The modules are an important phase of the project, since this is something that would make this project unique.
A7	All software development productions have a testing and bug fixing phase. This is important so that the users of the application wont experience any errors.
A8	Planning is an important phase in which takes time, but is critical to have a good workflow when working in a team and as an individual.

Table: More detailed information about our main activities

Management tools

We are using Microsoft Visual Studio Team Services as both a version control system, sprint planning program and IDE for this project. Files such as reports and other documents are written in Microsoft Word and synchronized through OneDrive.

Development tools

We are using Microsoft Visual Studio 2015 Enterprise edition for development of the project. This is an IDE we chose to use due to its flexibility and the ability to function as both management tool and development tool.

Internal evaluation

We are using Scrum agile development method which has periods where we assign workloads in periods called "Sprints". In these sprints we can assign work to everyone on this project and through Microsoft Team Services see if they have started a task, completed it or how much work may be left. This gives us a good overview of who does what and how much work is assigned to everyone. Each week we have a meeting with our supervisor where we go through what has been done and what should be done next.

Decision process

The biggest decisions about what to implement and the size of the project will be taken as a group with our supervisor at our meeting once a week. If we encounter lesser problems during development, this must be discussed within the group and decisions for how to counter it should be made. As mentioned earlier, we are using Scrum for the development process. This means we have a small meeting at the end of each day where we can discuss any difficulties we may have encountered and from there try for find a solution or work around for it.

5 DOCUMENTATION

Reports and technical reports

Since we are writing code that we want others to extend when we are done, we need to document the code well so that beginners, intermediates and experts can understand the code we have written. This involves describing how exactly code and methods are to function. This does not only help people who hasn't participated in this project, but it will also be helpful to us when writing the code.

Writing good documentation is a routine we need to have, so we will use the last 30 minutes of each day to check out our own code and write good documentation and in line comments describing the function and functionality. On top of this we read each other's sections to review the quality of the documentation and provide feedback to each other.

Everything we create will be synchronized with VSTS, so this is where it would be stored within the timeline of the project.

6 PLANNED MEETINGS AND REPORTS

Meetings

Supervisor meetings

Girts, our supervisor, proposed having a meeting every Friday at 11.00. On these meetings we will discuss what we have done so far, ask questions about issues/problems we have encountered, planning the next sprint in our Scrum methodology, etc. We also have his email-address, which makes it effortless to send him an email if there is something we need to know before the Friday meeting.

Project meetings

Since we are using Scrum we are having daily meetings each morning. This is where we plan what we should do for the rest of the day and how we are doing according to the plan.
Each Friday, after we are done with the supervisor meeting, we are planning the next week. This means putting all the new tasks into VSTS. This is also a critical state, because we cannot put everything we want into this planning phase. Since we only have 25-30 hours for each person in the group, we need to keep the workload at approx. 75-90 hours each week.

Periodic reports

As we have weekly meetings with our supervisor, periodic reports are not required.

Progress reports (incl. Milestones)

Having progress reports is usually an important aspect of the project, to keep all involved partners up to date for the different aspects of the project.

The supervisor will be kept in the loop through our weekly Friday meetings. Our supervisor also has access to the VSTS. From this point of view, he can see what we have been doing at any given time and if we are ahead or behind schedule.

Our assigner, Encripto AS, will be kept in the loop through a few meetings during the semester.

Milestones are used in project management to mark a specific point in a project timeline. These milestones are small goals within the project that leads to the finished product.

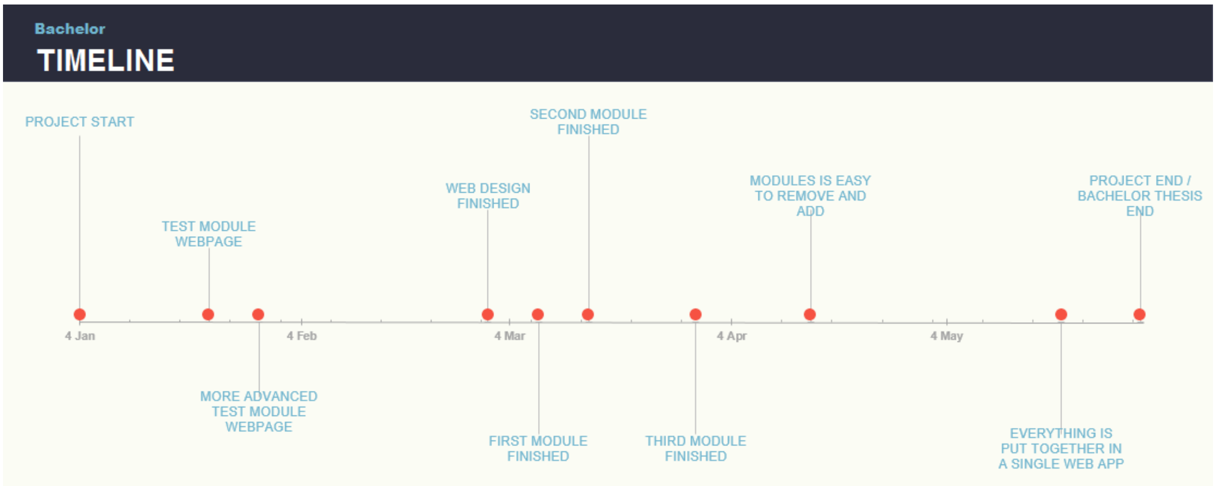


Figure 3: Milestones in our project. Note this might be inaccurate according to how it actually would be.

7 PLANNED DEVIATION MANAGEMENT

Since we have had weekly meetings with our supervisor, we have had someone besides ourselves that could step in and tell us if the project isn't going as planned.
If it does get out of hand, we would have to narrow down the scope of the project in order to have a deliverable.

Everyone involved in the project has a responsibility to ask themselves if what we have planned is achievable. We have to pay attention to our development plan, planned milestones, the weekly sprints and instructions from our supervisor to ensure that our development is on time.

8 EQUIPMENT REQUIREMENTS/PREREQUISITES FOR EXECUTION

Visual studio Enterprise [67] is usually a costly software, but as we are students we get the enterprise edition for free through DreamSpark. Visual studio team services are a subscription-based service. VSTS is free for up to five contributors. As we are only three students and one supervisor, we get it for free. Since we are mostly writing code and doing research, we don't need any other physical equipment other than our computers to do the development.

Other required software like the Microsoft Office and cloud storage is provided by each group member.

SAMARBEIDSAVTALE

Denne avtalen gjelder samarbeid mellom følgende organisasjoner/parter:

1. Jens Vingen
2. Per-Olav Eikrem
3. Ole-Martin Bratteberg

Avtalen gjelder samarbeid om planlegging, gjennomføring og rapportering for følgende tiltak:

Slik som vi har forstått skal vi rullere på «rollene» som vi har i løpet av prosjektet, slik at alle sitter igjen med samme kunnskap.
Alle på gruppen har et ansvar for å ta kontakt med kontaktlærer omtrent hver 14 dag.

Arbeidstider vil være fra 08.00 til 16.00 hver dag (vi har og noen timer på tirsdag og onsdag som alle på gruppen deltar i). Helger vil bli brukt ved behov, men hovedsakelig vil vi bruke ca. 8 timer hver hverdag til jobbing med bachelor.

Mål for samarbeidet:

Målet for samarbeidet er å utarbeide en bacheloroppgave på best og mest mulig effektiv måte for alle involverte.
Vi skal kunne vise til samarbeidsavtale ved uenigheter, for i dette prosjektet skal alle jobbe tilsvarende lik arbeidsmengde/timer.

Ansvarsfordeling – planlegging og gjennomføring av tiltaket:

Vi kommer til å ha flere roller i prosjektet, blant annet prosjektleder, Scrum-master, Git-master, rapporterings-sjef, rapport-sjef, m.m. Her kommer vi nok til å rullere litt hvem som har ansvar, slik at alle har mulighet for å lære det samme.
Vi må være i kontakt med kontaktlærer minst hver 14 dag, og dette er også noe som en eller flere av oss tar ansvar for å sette opp et avtalt møte.

Alle skal møte hver dag til avtalt tid og være på skolen avsluttende «skoledag».
Ved sykdom eller andre akutte saker skal vedkommende si ifra så snart som mulig.
Ved reise/planlagt bortetid skal vedkommende si ifra tidligst mulig.

Samarbeidsavtalens varighet:

Fra januar 2016 til slutt av bachelor prosjekt mai/juni 2016

Informasjonsplikt
Alle parter forplikter seg til å informere de andre om alle forhold som kan ha innvirkning på gjennomføringen av tiltaket.

8.2 Attachment 2: Friday meetings

Uke 6

Skip to end of metadata

- Created by Ole-Martin Bratteberg, last modified on Feb 12, 2016

Go to start of metadata

Date

12 Feb 2016

Attendees

- Ole-Martin Bratteberg
- Jens Vingen
- Per-Olav S. Eikrem
- Girts Strazdins

Goals

- Gjennomgang av uken som gikk og mål for neste

Discussion items

	Hva har blitt gjort		• Mandag/tirsdag kræsj. Bitbucket mot slutten av dagen tirsdag • Onsdag Bitbucket/Git branching. Meny/XSS/SQL etc kveldstid • Torsdag Webdagen. Features//XSS/SQL kveldstid • Fredag: XSS/SQL
	Vise		• Meny / Settings • XSS • SQL (kan legge inn data i database fra form)
	Neste uke		• Gjøre seg kjent med Confluence og JIRA • Overføre fra Team Services til Jira. Sette opp sprint • Areas arve layout • Slette route / disable area når "showInMenu" blir endret • Fullføre XSS og SQL modul • Undersøke hva/hvordan vi kan interagere med registrerte routes • Spørsmål som skal besvares i rapporten. ◦ Sårbarheter i .net ◦ Kilder ◦ Andre spørsmål som vi kan svare på med tekniske svar • Common module interface

			• Implement set-up method
	Spørsmål		• Demo av hvordan vi jobber på Git, OK?
Time	Item	Who	Notes

Action items

During meeting at 11

- Hva er konseptene som trengs i dette prosjektet?
  - Hva er spørsmålene vi ønkser å finne ut?
  - Har vi funnet ut nok om temaet før vi begynder?
- Start alltid med å søke opp
  - 2 kilder (kanskje 3?)
  - Skriv ned alt, inkl kilder

Spørsmål:

- Er .NET sårbart?
- Gå ut fra Top 10
  - Se om noen fortsatt ikke er fikset (siden pdf er fra 2011)
  - Lag en modul ut fra dette, siden det er det som kan vere er sårbarhet i nyeste .NET (/ASP.NET)

Entity framework Orm(?), guide something?

Framewrok should update db

Uke 7

Skip to end of metadata

- Created by Ole-Martin Bratteberg, last modified by Jens Vingen on Feb 22, 2016

Go to start of metadata

Date

19 Feb 2016

Participants

Ole-Martin Bratteberg

Per-Olav S. Eikrem

Jens Vingen

Girts Strazdins

Goal

-

What have we done, and what should we do?

What	Notes
Write with a few lines what we have done each day of the week	• **Monday** Started with developing the modules, but found out that we needed to answer some basic questions before continuing. Questions like: What is the most used version of .NET, what vulnerabilities exist for said version and if we can support different versions of .NET on our application. • **Tuesday** Lecture • **Wednesday** Researched vulnerabilities in the different versions of .NET. Finding usefull links with statistics which can be found under the Links section of Confluence. • **Thursday** Startet writing on the thesis and found and documented information about the most used versions on .NET as well as some vulnerabilities for that version. At the end of the the day, we also found out that there is a WebGoat version for .NET4 • **Friday**
What do we want to show at friday?	Documentation Ny meny (concept) "Golden" link m/sårbarheter
What are we going to do next week?	Write on the thesis, Explore solution to implement other projects (to better support other versions of the framework)

What	Notes
	Send this guy a mail (https://www.owasp.org/index.php/User:Jeff_Knutson) and ask him if there are any XSS vulnerabilities in ASP.NET 4.5.x++
Questions to Girts?	Mulitproject solutions? What is best practices? Vise coremvcapplication / dll / hvordan kan hovedapp "åpne cs klasse i dll"

Uke 8

Skip to end of metadata

- Created by Ole-Martin Bratteberg, last modified by Jens Vingen on Feb 26, 2016

Go to start of metadata

Date

23 Feb 2016

Participants

Ole-Martin Bratteberg

-
- Per-Olav S. Eikrem

Goal

-

What have we done, and what should we do?

What	Notes
Write with a few lines what we have done each day of the week	**Monday** Started writing on the other solutions that's out there Found information on possible new module (Open Redirect Attack), How the "main app" could find and communicate with multiple projects within solution

What	Notes
	▪ **Tuesday** Finished writing on the other solutions. We also discussed the content we though was important for our upcoming exam. How the "main app" could find and communicate with multiple projects within solution More on how to best do the module based system; portable areas, project in project, projects in solution, main app scans IIS for active URLS, Shared data as nuget packages.. • **Wednesday** Started writing on software we have been using during the project. Began looking into the theory behnd Open Redirect Attack and what might be the best implementation. More on how to best do the module based system; portable areas, project in project, projects in solution, main app scans IIS for active URLS, Shared data as nuget packages.. • **Thursday** Finished the writing on the software • **Friday** Discussed the work done througout the week and organised for meetings for the following week. Communication with Juan and planned meeting Monday 29. February at 15:00 - 16:00 B434. Tried to run WebGoat.NET without success.
What do we want to show at Friday?	Monday / Tuesday?
What are we going to do next week?	▪ Meeting with Juan Monday ▪ Final decision on how to do the module structure ▪ Find out how to make a proper module template ▪ Progress / Complete Open Redirect attack ▪ Continue writing on Bacelor Thesis ▪ Preparing for upcoming exam ▪ Further test multiproject solutions and implementations.
Questions to Girts?	Since the WebGoat.NET is using Webforms, should we use the MVC stucture (or atleast encourage developers to use it) in our project? This would make them more different since the structure and the code in webform vs MVC very different from each other.

What	Notes
	Or should be restrict the whole project to MVC, only allowing developers to use this architectural structure? What is the best way to use multiple project within one solution? Any teachers that has created a project like this before? Is there anybody at school we could ask questions that has experience with Visual Studio and project structure?

Uke 9

Skip to end of metadata

- Created by Jens Vingen, last modified on Mar 04, 2016

Go to start of metadata

Date

04 Mar 2016

Participants

Jens Vingen

Ole-Martin Bratteberg

Per-Olav S. Eikrem

Goal

Weekly meeting to show progression and general discussions about the project

What has been done?

Monday

- Meeting with Juan. Discussed problems and asked a few questions
- How make Webforms and MVC share the same layout

Tuesday

- How make Webforms and MVC share the same layout
- Started working with the SQL module (MVC structure)

Wednesday

- How make Webforms and MVC share the same layout
- Created first part of zip bomb
- Exam preparations for IF300114 Ingeniørfaglig systemteknikk og systemutvikling

Thursday

- Exam preparations for IF300114 Ingeniørfaglig systemteknikk og systemutvikling

Friday

- Discussions related to the weeks work
- Preparations for meeting
- Worked more on the SQL module, going to look into this this weekend.
    - Have some problems doing this without using the entity framework (this is SQL injection proof).

What to show?

Project where Webforms and MVC Areas share the same Master page (Jens)

Per-Olav S. Eikrem Du he modul å vise? - Ja, sann ca.

What to do next week

- Merge project with "shared layout" with existing module code
- Create additional modules
- Plan which of us is going to make which module. Juan suggested taking the OWASP top 10 and then divide them between us.
- Since we have an exam at Tuesday 15th, we will spend some time this week to prepare for this.

What to discuss?

- Have you made any research on "Visual Studio modules" and have any comment to Areas which we have decided to use?
- Meeting with Juan at Monday, he said it didn't matter whether we used webforms or MVC, any comments to this?
- Juan said the applications UI preferably would be like a "real life application". Our thought is to focus on the modules now and the UI later on
- When is the thesis supposed to be finished? Who could we ask?

Uke 11

Skip to end of metadata

- Created by Jens Vingen, last modified by Ole-Martin Bratteberg on Mar 21, 2016

Go to start of metadata

Date

18 Mar 2016

Participants

Jens Vingen

Ole-Martin Bratteberg

Goal

Weekly meeting to show progression and general discussions about the project

What has been done?
Week 10

Exam preparations

Monday

Exam preparations

Tuesday

Exam

Wednesday

Broken authentication module research

Started with making some design changes

Continued with the SQL module, but things don't work as they should.

Database usage research

Thursday

Broken authentication module research

Site.Master design bug, which makes working effective not possible.

Friday

Site.Master design "bug", still haven't found a solution on this problem.

What to show?

What to do next week

Monday

Discuss and deside on

- UI, content and effects
- Template content

Tuesday

Broken authentication module

SQL injection module

Week 13

Scrum-4-real

Finalize Broken authentication module

Finalize SQL injection module

Make a plan for and write on bachelor thesis

One additonal module

Questions to Girts?

Meeting

Affecting grade:

- Starting point: C
  - No big bugs and everything is OK
- Good argument in report
  - Clear

    - Discussion
- What is our WOW-factor? Needed to get a A
- Scientific paper
- Proving this is vvery good for company or business or school.
- Something outside of the box (multiple views?)
- We are using something outside the program (.NET, Visual Studio, Microsoft) Argument for this!!
- Leave material for next students who wants to learn .NET
- Good selling points for .NET for future NTNU

Try making a small user study

- Let's users try application
- Let Juan try it
    - Former "teacher" at Hials
    - Strong
    - Quote from Juan saying "This is good"

Write down selling points:

- It there enough evidence?
- Fix critical bugs

An A means something special!
Need to be without problems and a wow-factor.

We need to sell the product to get a good grade!

Include marketing in both presentation and in the report!

What is OUR contribution? Write in short what it is all about

NEXT MEETING:

Selling points

- This is our focus

Uke 14

Skip to end of metadata

- Created by Ole-Martin Bratteberg, last modified on Apr 08, 2016

Go to start of metadata

Date

08 Apr 2016

Participants

Ole-Martin Bratteberg

Per-Olav S. Eikrem

Jens Vingen

Girts Strazdins

Goal

It is been a few weeks since last meeting, but we have been doing good.

What has been done?

Week 12

Developed the SQL module further

Decided what to implement during the next couple of weeks.

Decided on important issues like how the menus should look like, design, database design, real-like application

Week 13

Finished the basic of the SQL

Started the login module

Researching Broken Authentication

Tried making menu from MapRoute, but didn't work as good as we wanted.

Tried importing the previous menu into the new project, but it didn't quite work

Confluence was down, so didn't have the opportunity to write logs and retrieve things..

Wrote some pages on the thesis

Monday

Started with the HidddenContent jquery (never writte anything in jqury, and little experience with javascript)

Decided that we didn't want to much information on each of the pages, because we wanted the application to be more real-life-like.

Trimmed down the SQL module

Researching parsel rendering and user control for the menu

Worked with the login funcionality

Tuesday

Found out how to use bootstrap with Asp:Menu Controller

Reseach cookie and session management for the login-module.

Went through a JavaScript tutorial/learning platform on codeacademy to learn more JavaScript

Wednesday

Began using Scrum for real

The menu is finished, so we needed to push and pull the commits so that everyone had the changes (with a lot of errors ofc)

Took some more tutorals on the javascript course

Thursday

Finally finished the HiddenContent jquery, which will be used on all the modules that need it.

Fixed some error in the SQL module which made the application crash.

Wrote on the bachelor thesis

Started with the reasearch on insure object refereaces, but ended up with reasearching windows authentication system,

Friday

What to show?

- Area to database, database to menu (jens)
- Hack to enable bootstrap styling on asp:menu (jens)
- Injection module (pero)
- SQL module (olem)

What to do next week

- Login / user management
- Peephole documentation
- Module: insecure object references

Questions to Girts?

- The hack to enable bootstrap styling, reuse of code
- In general, correct usage of code snippets found online

Meeting

Woow-factor

- Real people testing
- GitHub
- Scientific article
  - Tool for teaching security (and .NET? ^^)

Scientific question

- Your direct competitors
  - How are we better?
- Tools which are in the whole life-cycle
  - Programming language, .NET, jquery, bootstrap, javascript, css, html
  - Things we could have used, but didn't - WHY?
  - Why have we chosen
- Let other try the app

- Students with testing and question
- Juan

COMPETETIVE SOLUTION

Make a summorize table at the end OR start

Uke 15

Skip to end of metadata

- Created by Ole-Martin Bratteberg, last modified on Apr 15, 2016

Go to start of metadata

Date

15 Apr 2016

Participants

Per-Olav S. Eikrem

Ole-Martin Bratteberg

-

Summary

Added some design to the application.

What has been done?

Monday

Found out that we needed to build an unvalidated redirct module and started reasearching that.

Decided how the insecutre object module should be and research customer user management.

Looking into the default login vs making our own login example. Pros and cons with both suggestions.

Tuesday

Built the unvalidated redirect module.

Worked on the insecure object module

Partly decided on using the built in login. Main problem is that modifying it beyond configuration is a bit hard. How to remove salting to make hash simpler to crack, how to give feedback when username is correct and password is incorrect etc.?

Wednesday

Started working on the design of the application, had to do some reasearch on what we needed to include in the solution and in the application.

Misconfigured the defualt login and implemented refleced XSS after login has been done.

How to remove salt on the MS login

Thursday

Discarded the previous design and started over with a new one. This is currently in use.

The upload file module was not as vulnerable as we would like, so we made a remake / new one which is highly vulnerable.

The menu is going to get some kind of drag-and-drop grading (jquery) which makes configuring the menu easier.

Friday

Meeting notes

Looking into the content for our bachelor thesis report.

After the meeting will plan the next sprint and write on the bachelor thesis.

checking how to make default login more vulnerable.

What to show?

Unvalidated redicect

Frontpage / design - thoughts?

Implemented modules in a more real life scenario kind of way. (reflecte xss)

What to do next week

- Ole-Martin is going to make some documentation pdf with content from the information security course. This is going to be filled in under the documentation section on the front page.
  Fix bugs in the design
  Comment code that is already written by me and fix information input in the hiddenContent.
  Look at the area template design with Jens
  Going to be in Gdansk from saturday to tuesday, so it is going to be a reduced week for me.
- Merge databases and try to make modules connect in a more natural way, i.e doing a succesful SQLinjection should give you email and password of users registered, making it possible to log in. XSS should be able to steal session id cookies etc.
- Testing and documenting what vulnerabilities we got and how to exploit them - put some information in documentation.

Questions to Girts?

Should we use another template when writing our bachelor thesis?

Scientific report?

Design? I want to change colors

Meeting

Abstract

Introduction and problemstates (1 or 2 sections depending on content)

Related work - litterature studies and other tools - how is our better or different?

Acedemic paper review - articles on feks Google scholar

------------------------------ Line of what have been done before and what should be done afterwards

Purposed solution - design rules, goals, artchitecture - First product type or just a part of the solution - General idea and purpose of the project - Modularity

Evaluation - Analytical - Table (of features) - Features of the project (not just by ourself, but others) If something have three solution, descripe all with cons and pros and tell WHY we chose the one we did

Uke 16

Skip to end of metadata

- Created by Ole-Martin Bratteberg, last modified on Apr 22, 2016

Go to start of metadata

Date

22 Apr 2016

Participants

Ole-Martin Bratteberg

-

Goal

Weekly meeting to show progression and general discussions about the project

What has been done?

Monday

Gdansk

Jobb

Android

Tuesday

Gdansk
Jobb

Anroid

Wednesday

Didn't feel so good

Jobb

Looking into how to disable salt + hash on default login

Thursday

"Meeting" with Girts about the thesis structure.

Changed some of the design and colours in the frontpage and wrote about the design in the thesis.

Edit menu functionality

Decided on making a custom login

Made successfull login with md5 hash which is simple to crack with online crack tools.

Friday

Writing this report

Write more on the thesis and plan the next week/sprint.

Decided on reworking the default login to use session ID or membership to control when a user i logged in or not.

What to show?

The new colours / design. Look better now. Not the "warning" red colour.

What to do next week

Module template, if we feel like we could make it now. Since we have the menu in place, we should be able to do it.

Make some documentation from the information security subject. Create them as pdfs and uploads them to the documentation.html.

More active with the bachelor thesis.

Questions to Girts?

Materials

"*[Here describe the practical equipment that is used to display the results in the report. There is here a detailed list of the equipment that is used (multimeter, scoops etc.), measurement methods used, as well as a detailed description of any alignment/lab setup/measure the alignment. If the equipment is part of the product to be developed under the project, describes this under the chapter results.]"*

We haven't used any practical equipement other than computers, keyboard and mouse.

Does this mean application like Jira, Confluence, etc. in our case?

Uke 17

Skip to end of metadata

- Created by Ole-Martin Bratteberg, last modified by Jens Vingen on Apr 29, 2016

Go to start of metadata

Date

29 Apr 2016

Participants

Ole-Martin Bratteberg

Jens Vingen

Per-Olav S. Eikrem

Goal

Weekly meeting to show progression and general discussions about the project

What has been done?

Monday

Started focusing more on the bachelor thesis report

Simplified database structure, connection strings cleanup

Design fixes

Tuesday

Bachelor report

Merge Databases into one

Wednesday

Bachelor report

Thursday

Bachelor report

Added SSL

Database changes to add user accounts

Friday

Bachelor report

Design fixes

What to show?

What to do next week
Questions to Girts?
Which bachelor report frontpage

- Is the application rich enough?
- **Presentation:** Make video showing a scenario or do a live demonstration?

Meeting

Girts har en rapport, avvente godkjenning på den

Uke 18

Skip to end of metadata

- Created by Ole-Martin Bratteberg on May 06, 2016

Go to start of metadata
Date

06 May 2016

Participants

Ole-Martin Bratteberg

Jens Vingen

Per-Olav S. Eikrem

Girts Strazdins

Goal

Weekly meeting to show progression and general discussions about the project

What has been done?

This week we have been pretty much writing on the thesis, but Jens have been adding accounts and money transfers

What to show?

Money transfers from one account to another

What to do next week

Small bugs with money transfer

Other small bugs we find

Questions to Girts?

Could you read thorugh the thesis as it is now and give us some feedback on what's good and what's not?
What to include and what we should remove?

The article to NISK, what should be the content of this article? What could be our contribution to this conferance?

Uke 19

Skip to end of metadata

- Created by Ole-Martin Bratteberg, last modified by Per-Olav S. Eikrem on May 13, 2016

Go to start of metadata

Date

13 May 2016

Participants

Ole-Martin Bratteberg

Per-Olav S. Eikrem

Jens Vingen

Girts Strazdins

Goal

Weekly meeting to show progression and general discussions about the project

What has been done?

We have been working on the report, documentation, fixed some bugs and made a virtual machine with the application on it.

What to show?
What to do next week
Questions to Girts?

Problem to be adressed?

Example problm:

With society becoming more and more digitalized, the need for proper security of information is rapidly increasing.
With a threat picture that is in constant change, we want to address the issue by making a modular and easily expandable application to ensure it stays relevant.

8.3 Attachment 3: Meeting with Encripto AS

Date

29 Feb 2016

Participants

Per-Olav S. Eikrem

Ole-Martin Bratteberg

Jens Vingen

What have we done, and what should we do?

What has been done	Notes
Write with a few lines what we have done	• Great amount of time spent: • Learning Visual studio and MVC structure • SQL strings vs Entity Framework (ORM) vs LINQ. Parameterization • How to actually make the "modules" • Areas • Portable areas • Multiple projects within project • Multiple projects in solution • Shared data as nuget-packages ▪ DLLs vs running from Visual Studio • Resourcing vulnerabilities to certain .NET versions https://www.cvedetails.com/version-list/26/2002/1/Microsoft-.net-Framework.html ▪ How Git works, making minimal of errors during the project. ▪ Researching .NET framework and C# • Researching existing solutions (WebGoat, DVWA...) • We have made a framework that gives the user control of which modules should be shown in the application. The user can add/edit/delete links as he whishes. We have not yet ended on a solution for the modular program (multiple project solution / muliple area etc.) • We have made a simple XSS module showing Stored XSS with an almost functioning restore database button (deletes the database but fails at building it up again)

What has been done	Notes
What do we want to show?	Demonstrate what we have and what we are planning to do now. Multiple projects in solution "Tag-menu" **Why multiple projects in solution:** ▪ Full flexability for module developers. They can choose which .NET framework to use and MVC/Webforms. ▪ Could clone existing projects and include in system ▪ Could easily include multiple "real life" applications **Why not multiple projects in solution:** ▪ Harder to implement in "main" project 1. Different layouts ▪ Complexity Short explanation of thoughts Show planned Menu
Questions to Juan	Login at start of application or at each module that may require a login to demonstrate the vulnerability? For a web app utilizing Entity Framework or LINQ-to-SQL, do you know if these could be vulnerable to SQL injection? Is the default filtering in .NET sufficient protection against XXS? In our module we have disabled validation and store the data as Html.Raw() Which vulnerabilities would you reccomend focusing on? If we should simulate CSRF, which approach would you recommend? Banking, Employee Salary raise etc?

What has been done	Notes
	* How should we implement he bad site that injects the vulnerable code? Is it relevant to involve .NET 2.0? Should we focus on the newest version available? Do they have different vulnerabilities?

Meeting

Hvordan deploye om jeg laste rned fra GitHub?

Forskjell på dll og åpen kildekode

1. Dll kan dekompileres
2. Gi kompilert og ukompilert versjon av kode

.NET har muligheten for å kjøre kode on the fly
IIS

Mappe med modul/instrukser

Velg en måte
Prøv
Finn begrensninger

Syntaz webform/razor

- Prinsippet er samme
- C# vs Java (buffer overflow)

LINK er sikkert

EF var han usikker på

.NET versjon
2.0 xp
3.5 Win 7 og nyere

Login

- CSRF
- Tid/Respons

SQL

1. Bruk SQL commands
2. LINQ er sikkert, så anbefal det

Modul som benytter API som henter informasjon fra databasen,

SQL

Command injection

SMTP injection

Fil opplasting

- Ikke godt sikret
- Kjøre egne filer
- http./................/uploads/myfile.html
- zip fil
    - Må laste opp og dekomprimeres
    - Kan unzippes via command injection
    - zip.bomb
        - 1 PB er ca 100kb
        - Denial of disk
        - dd (verktøy)

        - Inputfile = / dev / zero outputfile = ...
- OWASP Unresricted file uploads

Planen fremover:

- Lage en god plan
- Dele OWASP top 10 i 3 og utvikla modul

Kjøre på linux

mono?

Windows lisens

LEGG UT KILDEKODE og disclaimer om lisensbruk

8.4 Attachment 4: File structure

```
Folder PATH listing
Volume serial number is AE05-DF7F
C:.
|   About.aspx
|   About.aspx.cs
|   About.aspx.designer.cs
|   AreasHandling.cs
|   Bank.csproj
|   Bank.csproj.user
|   BankAccount.cs
|   BankAccountType.cs
|   BootstrapMenu.cs
|   Bundle.config
|   ControllerExtensions.cs
|   Default.aspx
|   Default.aspx.cs
|   Default.aspx.designer.cs
|   favicon.ico
|   Feedback.cs
|   Files.cs
|   Global.asax
|   Global.asax.cs
|   MainMenu.cs
|   packages.config
|   PeepholeModel.Context.cs
|   PeepholeModel.Context.tt
|   PeepholeModel.cs
|   PeepholeModel.Designer.cs
|   PeepholeModel.edmx
|   PeepholeModel.edmx.diagram
|   PeepholeModel.tt
|   Project_Readme.html
|   SiteMenu.ascx
|   SiteMenu.ascx.cs
|   SiteMenu.ascx.designer.cs
|   Startup.cs
|   TableTemplates.css
|   test.html
|   test2.html
|   test3.html
|   Transaction.cs
|   Users.cs
|   ViewSwitcher.ascx
|   ViewSwitcher.ascx.cs
|   ViewSwitcher.ascx.designer.cs
|   Web.config
|   Web.Debug.config
|   Web.Release.config
|
+---Account
|       AddPhoneNumber.aspx
|       AddPhoneNumber.aspx.cs
|       AddPhoneNumber.aspx.designer.cs
|       Confirm.aspx
|       Confirm.aspx.cs
|       Confirm.aspx.designer.cs
|       Forgot.aspx
|       Forgot.aspx.cs
|       Forgot.aspx.designer.cs
|       Lockout.aspx
|       Lockout.aspx.cs
|       Lockout.aspx.designer.cs
|       Login.aspx
|       Login.aspx.cs
|       Login.aspx.designer.cs
|       Manage.aspx
|       Manage.aspx.cs
|       Manage.aspx.designer.cs
|       ManageLogins.aspx
|       ManageLogins.aspx.cs
|       ManageLogins.aspx.designer.cs
|       ManagePassword.aspx
|       ManagePassword.aspx.cs
|       ManagePassword.aspx.designer.cs
|       OpenAuthProviders.ascx
|       OpenAuthProviders.ascx.cs
|       OpenAuthProviders.ascx.designer.cs
|       Register.aspx
|       Register.aspx.cs
|       Register.aspx.designer.cs
|       RegisterExternalLogin.aspx
|       RegisterExternalLogin.aspx.cs
|       RegisterExternalLogin.aspx.designer.cs
|       ResetPassword.aspx
|       ResetPassword.aspx.cs
|       ResetPassword.aspx.designer.cs
|       ResetPasswordConfirmation.aspx
|       ResetPasswordConfirmation.aspx.cs
|       ResetPasswordConfirmation.aspx.designer.cs
|       TwoFactorAuthenticationSignIn.aspx
|       TwoFactorAuthenticationSignIn.aspx.cs
|       TwoFactorAuthenticationSignIn.aspx.designer.cs
|       VerifyPhoneNumber.aspx
|       VerifyPhoneNumber.aspx.cs
|       VerifyPhoneNumber.aspx.designer.cs
|       Web.config
|
+---App_Data
|       Peephole.mdf
|       Peephole_log.ldf
|
+---App_Start
|       BundleConfig.cs
|       IdentityConfig.cs
|       RouteConfig.cs
|       Startup.Auth.cs
|       WebApiConfig.cs
|
+---Areas
|   +---CustomerInfo
|   |   |   CustomerInfoAreaRegistration.cs
|   |   |
|   |   +---Controllers
|   |   |       DefaultController.cs
|   |   |
|   |   +---Models
|   |   \---Views
|   |       |   web.config
|   |       |
|   |       +---Default
|   |       |       Details.cshtml
|   |       |       Edit.cshtml
|   |       |
|   |       +---Shared
|   |       \---Usersdd
|   +---Employee
|   |       EmployeeAreaRegistration.cs
|   |       Search.aspx
|   |       Search.aspx.cs
|   |       Search.aspx.designer.cs
|   |
|   +---Feedback
|   |   |   DataAreaRegistration.cs
|   |   |
|   |   +---Controllers
|   |   |       CommentController.cs
|   |   |
|   |   +---Models
|   |   \---Views
|   |       |   web.config
|   |       |
|   |       +---Comment
|   |       |       Comments.cshtml
|   |       |
|   |       \---Shared
|   +---LoanApplication
|   |       Application.aspx
|   |       Application.aspx.cs
|   |       Application.aspx.designer.cs
|   |       FileUploadAreaRegistration.cs
|   |
|   +---Login
|   |   |   Login.aspx
|   |   |   Login.aspx.cs
|   |   |   Login.aspx.designer.cs
|   |   |   LoginAreaRegistration.cs
|   |   |   Logout.aspx
|   |   |   Logout.aspx.cs
|   |   |   Logout.aspx.designer.cs
| | | Register.aspx
| | | Register.aspx.cs
| | | Register.aspx.designer.cs
| | | Success.aspx
| | | Success.aspx.cs
| | | Success.aspx.designer.cs
| | | Welcome.aspx
| | | Welcome.aspx.cs
| | | Welcome.aspx.designer.cs
| | |
| | +---Controllers
| | +---Models
| | \---Views
| |     | web.config
| |     |
| |     \---Shared
| +---PeroTesting
| | | PeroTesting.aspx
| | | PeroTesting.aspx.cs
| | | PeroTesting.aspx.designer.cs
| | | PeroTestingAreaRegistration.cs
| | |
| | +---Controllers
| | +---Models
| | \---Views
| |     | web.config
| |     |
| |     \---Shared
| +---Redirect
| | | RedirectAreaRegistration.cs
| | |
| | +---Controllers
|   |   |       RedirectController.cs
|   |   |
|   |   +---Models
|   |   |       RedirectModel.cs
|   |   |
|   |   \---Views
|   |       |   web.config
|   |       |   _ViewStart.cshtml
|   |       |
|   |       +---Redirect
|   |       |       Index.cshtml
|   |       |       Redirect.cshtml
|   |       |
|   |       \---Shared
|   |               _Layout.cshtml
|   |
|   +---RouteTest
|   |   +---Controllers
|   |   \---Views
|   |       \---Default
|   +---TransferFunds
|   |   |   TransferFundsAreaRegistration.cs
|   |   |
|   |   +---Controllers
|   |   |       BankAccountsController.cs
|   |   |       TransactionsController.cs
|   |   |
|   |   +---Models
|   |   |       BankAccountModel.cs
|   |   |
|   |   \---Views
|   |       |   web.config
| |     |
| |     +---BankAccounts
| |     |       Create.cshtml
| |     |       Delete.cshtml
| |     |       Details.cshtml
| |     |       Edit.cshtml
| |     |       Index.cshtml
| |     |
| |     \---Transactions
| |             Create.cshtml
| |             Delete.cshtml
| |             Details.cshtml
| |             Edit.cshtml
| |             Index.cshtml
| |
| +---_MVCTemplate
| | |  MVCTemplateAreaRegistration.cs
| | |
| | +---Controllers
| | |       MVCTemplateController.cs
| | |
| | +---Models
| | |       MVCTemplateModel.cs
| | |
| | \---Views
| |     |  web.config
| |     |
| |     \---MVCTemplate
| |             Index.cshtml
| |
| \---_WebformTemplate
|           Default.aspx
|       Default.aspx.cs
|       Default.aspx.designer.cs
|       WebformsTestAreaRegistration.cs
|
+---bin
| |  Antlr3.Runtime.dll
| |  Antlr3.Runtime.pdb
| |  AspNet.ScriptManager.bootstrap.dll
| |  AspNet.ScriptManager.jQuery.dll
| |  EntityFramework.dll
| |  EntityFramework.SqlServer.dll
| |  EntityFramework.SqlServer.xml
| |  EntityFramework.xml
| |  Microsoft.AspNet.FriendlyUrls.dll
| |  Microsoft.AspNet.FriendlyUrls.xml
| |  Microsoft.AspNet.Identity.Core.dll
| |  Microsoft.AspNet.Identity.Core.xml
| |  Microsoft.AspNet.Identity.EntityFramework.dll
| |  Microsoft.AspNet.Identity.EntityFramework.xml
| |  Microsoft.AspNet.Identity.Owin.dll
| |  Microsoft.AspNet.Identity.Owin.xml
| |  Microsoft.AspNet.Web.Optimization.WebForms.dll
| |  Microsoft.CodeDom.Providers.DotNetCompilerPlatform.dll
| |  Microsoft.CodeDom.Providers.DotNetCompilerPlatform.xml
| |  Microsoft.Owin.dll
| |  Microsoft.Owin.Host.SystemWeb.dll
| |  Microsoft.Owin.Host.SystemWeb.xml
| |  Microsoft.Owin.Security.Cookies.dll
| |  Microsoft.Owin.Security.Cookies.xml
| |  Microsoft.Owin.Security.dll
| |  Microsoft.Owin.Security.Facebook.dll
| |  Microsoft.Owin.Security.Facebook.xml
| | Microsoft.Owin.Security.Google.dll
| | Microsoft.Owin.Security.Google.xml
| | Microsoft.Owin.Security.MicrosoftAccount.dll
| | Microsoft.Owin.Security.MicrosoftAccount.xml
| | Microsoft.Owin.Security.OAuth.dll
| | Microsoft.Owin.Security.OAuth.xml
| | Microsoft.Owin.Security.Twitter.dll
| | Microsoft.Owin.Security.Twitter.xml
| | Microsoft.Owin.Security.xml
| | Microsoft.Owin.xml
| | Microsoft.ScriptManager.MSAjax.dll
| | Microsoft.ScriptManager.WebForms.dll
| | Microsoft.Web.Infrastructure.dll
| | Newtonsoft.Json.dll
| | Newtonsoft.Json.xml
| | Owin.dll
| | Peephole.dll
| | Peephole.dll.config
| | Peephole.pdb
| | System.Net.Http.Formatting.dll
| | System.Net.Http.Formatting.xml
| | System.Web.Helpers.dll
| | System.Web.Helpers.xml
| | System.Web.Http.dll
| | System.Web.Http.WebHost.dll
| | System.Web.Http.WebHost.xml
| | System.Web.Http.xml
| | System.Web.Mvc.dll
| | System.Web.Mvc.xml
| | System.Web.Optimization.dll
| | System.Web.Optimization.xml
| | System.Web.Providers.dll
| |   System.Web.Providers.xml
| |   System.Web.Razor.dll
| |   System.Web.Razor.xml
| |   System.Web.Webpages.Deployment.dll
| |   System.Web.Webpages.Deployment.xml
| |   System.Web.Webpages.dll
| |   System.Web.Webpages.Razor.dll
| |   System.Web.Webpages.Razor.xml
| |   System.Web.Webpages.xml
| |   WebGrease.dll
| |   WebMatrix.Data.dll
| |   WebMatrix.Data.xml
| |
| \---roslyn
|         csc.exe
|         Microsoft.Build.Tasks.CodeAnalysis.dll
|         Microsoft.CodeAnalysis.CSharp.dll
|         Microsoft.CodeAnalysis.dll
|         Microsoft.CodeAnalysis.VisualBasic.dll
|         Microsoft.CSharp.Core.targets
|         Microsoft.VisualBasic.Core.targets
|         System.Collections.Immutable.dll
|         System.Reflection.Metadata.dll
|         vbc.exe
|         VBCSCompiler.exe
|         VBCSCompiler.exe.config
|
+---Content
| |   bootstrap.css
| |   bootstrap.min.css
| |   custom.css
| |   fotorama.css
|   |   menu.css
|   |   Site.css
|   |
|   \---Font
|           BebasNeue.otf
|
+---Controllers
|       DefaultController.cs
|       DocumentationController.cs
|       FrontpageController.cs
|       HomeController.cs
|       MainMenusController.cs
|
+---fonts
|       glyphicons-halflings-regular.eot
|       glyphicons-halflings-regular.svg
|       glyphicons-halflings-regular.ttf
|       glyphicons-halflings-regular.woff
|
+---img
|       bank1.jpg
|       bank2.jpg
|       bank3.jpg
|       bank4.jpg
|
+---jquery
|       demos.css
|       jquery-2.2.3.min.js
|
+---jqueryui
|   |   index.html
|   |   jquery-ui.css
| | jquery-ui.js
| | jquery-ui.min.css
| | jquery-ui.min.js
| | jquery-ui.structure.css
| | jquery-ui.structure.min.css
| | jquery-ui.theme.css
| | jquery-ui.theme.min.css
| |
| +---external
| | \---jquery
| |         jquery.js
| |
| \---images
|         ui-icons_444444_256x240.png
|         ui-icons_555555_256x240.png
|         ui-icons_777620_256x240.png
|         ui-icons_777777_256x240.png
|         ui-icons_cc0000_256x240.png
|         ui-icons_ffffff_256x240.png
|
+---Models
|     IdentityModels.cs
|
+---NewFolder1
+---obj
| \---Debug
|     | Bank.csproj.FileListAbsolute.txt
|     | Bank.csprojResolveAssemblyReference.cache
|     | DesignTimeResolveAssemblyReferences.cache
|     | DesignTimeResolveAssemblyReferencesInput.cache
|     | Peephole.csproj.FileListAbsolute.txt
|     | Peephole.csprojResolveAssemblyReference.cache
|   |  Peephole.dll
|   |  Peephole.pdb
|   |  TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs
|   |  TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs
|   |  TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs
|   |
|   +---edmxResourcesToEmbed
|   |      MyModel.csdl
|   |      MyModel.msl
|   |      MyModel.ssdl
|   |      PeepholeModel.csdl
|   |      PeepholeModel.msl
|   |      PeepholeModel.ssdl
|   |
|   \---TempPE
|          MyModel.cs.dll
|          MyModel.Designer.cs.dll
|          PeepholeModel.cs.dll
|          PeepholeModel.Designer.cs.dll
|
+---Properties
|   AssemblyInfo.cs
|   Settings.Designer.cs
|   Settings.settings
|
+---Scripts
| |  bootstrap.js
| |  bootstrap.min.js
| |  fotorama.js
| |  hiddenContent.js
| |  jquery-1.10.2.intellisense.js
| |  jquery-1.10.2.js
|   |   jquery-1.10.2.min.js
|   |   jquery-1.10.2.min.map
|   |   jquery-ui.min.js
|   |   jquery.cookie.js
|   |   modernizr-2.6.2.js
|   |   respond.js
|   |   respond.min.js
|   |   _references.js
|   |
|   \---WebForms
|       |   DetailsView.js
|       |   Focus.js
|       |   GridView.js
|       |   Menu.js
|       |   MenuStandards.js
|       |   SmartNav.js
|       |   TreeView.js
|       |   WebForms.js
|       |   WebParts.js
|       |   WebUIValidation.js
|       |
|       \---MSAjax
|               MicrosoftAjax.js
|               MicrosoftAjaxApplicationServices.js
|               MicrosoftAjaxComponentModel.js
|               MicrosoftAjaxCore.js
|               MicrosoftAjaxGlobalization.js
|               MicrosoftAjaxHistory.js
|               MicrosoftAjaxNetwork.js
|               MicrosoftAjaxSerialization.js
|               MicrosoftAjaxTimer.js
|               MicrosoftAjaxWebForms.js
|           MicrosoftAjaxWebServices.js
|
+---Userfiles
|       This file is just for show.pdf
|
\---Views
    |   web.config
    |   _ViewStart.cshtml
    |
    +---Default
    |       Index.cshtml
    |
    +---Documentation
    |       Index.cshtml
    |
    +---Frontpage
    |       Index.cshtml
    |
    +---Home
    |       Index.cshtml
    |
    +---MainMenus
    |       Create.cshtml
    |       Delete.cshtml
    |       Details.cshtml
    |       Edit.cshtml
    |       Index.cshtml
    |
    \---Shared
            RazorView.aspx
            RazorView.aspx.cs
            RazorView.aspx.designer.cs
```

Site.Master

Site.Master.cs

Site.Master.designer.cs

_Layout.cshtml

_SiteLayout.cshtml

8.5 Attachment 5: NISK article

Peephole – ASP.NET Information Security Teaching Tool

Ole-Martin Bratteberg
Jens Vingen
Per-Olav Eikrem
Girts Strazdins
NTNU in Ålesund

Abstract

Information security is an ever ongoing process. As the technology evolves, so does the vulnerabilities and attacks. The amount of information available online is growing intensively and so does the need for secure web applications. Web application security starts with the knowledge of the developers. Peephole aims to be a tool to aid in teaching how to code more secure web application. There are many applications with similar functionality for PHP/Java application, but no good solution for ASP.NET which is the target for our project. We strongly believe in "learning by doing" which has influenced the design of Peephole. Vulnerable application, hints, explanations, solutions. Peephole aims to be up to date, easily extensible (both Webforms and MVC, shared layout). Gather other web application projects within the Peephole solution.

INTRODUCTION

In our fifth semester of studying Bachelor of Engineering in Computer Science we had a course about Information Security. The teacher Juan J. Güelfo at Encripto AS expressed the need for a teaching tool that demonstrated the vulnerabilities for the ASP.NET framework.

When learning information security, it is important to understand the underlying technologies that the vulnerabilities are a product of. One of the best ways to learn is to actually exploit the application and look at the code to understand how it is all connected. ASP.NET stands for 15.8 % [14] of all known server-side languages which makes up a significant number of systems running the technology amplifying the demand for Peephole developed with and for the Microsoft platform.

Problem to be addressed

To ensure best possible security for web applications, education for the developers plays a major part and education needs practice. With a threat picture that is constantly changing, how do we best create the most comprehensive, useful and over time current information security teaching tool for ASP.NET.

To achieve the desired result, these are the most important aspects to consider:

- Distribution
- Architecture
- Extensibility
- Implemented vulnerabilities

Constraints

The Peephole solution will not provide a complete teaching environment with a full set of vulnerabilities nor information about such vulnerabilities. Instead, we focus on providing the best possible foundation for worldwide cooperation that over time could become the most comprehensive and current information security teaching tool for ASP.NET.

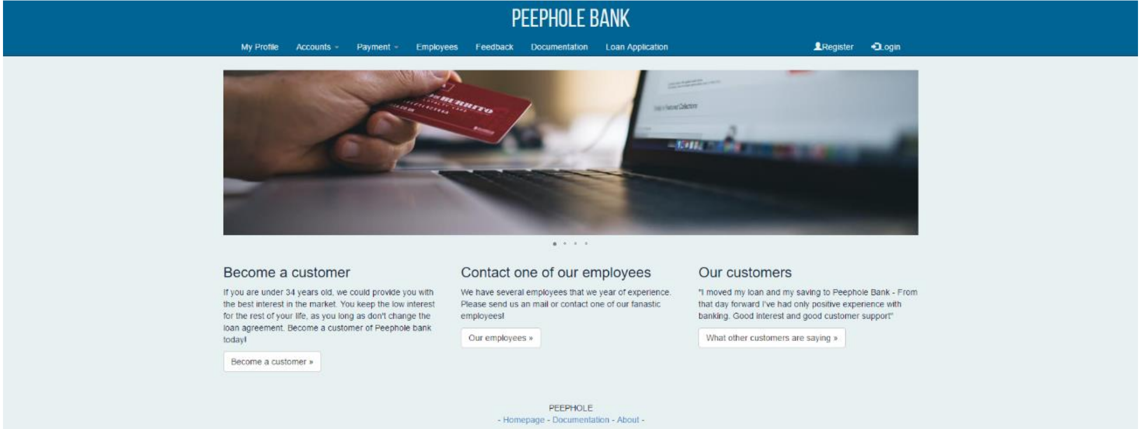
MATERIALS AND METHODS

When developing an intentional vulnerable web application, we had to decide which vulnerabilities we wanted to focus on. We did large amounts of research on what was the most common vulnerabilities in web applications and found that covering the OWASP Top 10 list would be a good start for our modular solution.

The application has been sent to testing to a known security expert and master students which have specialized in information security and computer security.

RESULTS

Our application has been built with current .NET technology, which offers an environment not common for web security training, but which has a very important market share in web application development. The application is modular with the possibility to extend with both new Areas within the same project and with Multiple Projects in the same solution. We have facilitated the application for both WebForms and MVC development and support most developers in ASP.NET. The vulnerabilities we have prioritized implementing are from the OWASP Top 10 [68] list. This is a list of the most critical web application security flaws there are.
We wanted the application look like a real bank as you can see in Figure .



Become a customer

If you are under 34 years old, we could provide you with the best interest in the market. You keep the low interest for the rest of your life, as you long as don't change the loan agreement. Become a customer of Peephole bank today!

Contact one of our employees

We have several employees that we year of experience. Please send us an mail or contact one of our fanastic employees!

Our customers

"I moved my loan and my saving to Peephole Bank - From that day forward I've had only positive experience with banking. Good interest and good customer support"

What other customers are saying »
PEEPHOLE
- Homepage - Documentation - About -

Figure 8.1: Screenshot of the Peephole Bank. This is the first of many real-like application Peephole could potentialy contain.

We had some core requirements for our application before starting to develop:

- Written in: ASP.NET, C#, MSSQL
- Hosted: Virtual Machine or Visual studio
- Price: Free

We designed the application to look and feel like a banking application so that it resembles real life scenario and not just a “click module one – XSS, module two – SQL, etc.”.

We wanted some security experts to test out the Peephole Bank. One of them was our assigner from Encripto AS and the other one was a Master Student from Guangdong University of Technology (GDUT), China. They both found vulnerabilities and our assigner gave us some feedback:

> "Peephole is an excellent project that takes web security training to a next level, since it tackles many of the unsolved challenges seen until now. This application has been built with current .NET technology, which offers an environment not common for web security training, but which has a very important market share in web application development."

TABLES AND GRAPHS

Table showing similar applications up against our solution.

🟩 Yes 🟥 No ▨ N/A

Application name Feature	Peephole	DVWA	Webgoat	WebGoat.NET	Hera	Hack.ME	NOWASP	Webscarab	bWAPP
Open source	🟩	🟩	🟩	🟩	🟥	🟥	🟩	🟥	🟩
Free	🟩	🟩	🟩	🟩	🟥	🟩	🟩	🟩	🟩
Updated within last year	🟩	🟩	🟩	🟥	▨	▨	🟥	🟥	🟥
English language	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Does not require internet connection	🟩	🟩	🟩	🟩	🟥	🟥	🟩	▨	🟩
Real-life application	🟩	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥
Includes OWASP Top 10	🟩	▨	🟩	🟩	▨	▨	▨	🟥	🟩
Over 50 vulnerab.	🟥	🟥	🟩	🟥	🟩	🟩	🟥	▨	🟩
Modular	🟩	🟥	🟩	🟥	🟥	🟥	🟥	🟥	▨
Guidance	🟩	🟩	🟩	🟩	🟩	🟥	🟥	🟩	🟩
Written in	ASP.NET C# MSSQL	PHP MySQL	Java	ASP.NET C# MSSQL	▨	▨	PHP JS	▨	PHP MySQL

Table 1: Comparison between existing solutions and our proposed solution

As we can see in Table 1, the biggest differences are that Peephole is a Real-Life application and that it is used the ASP.NET framework. It does not have over 50 vulnerabilities now, but in the future it could grow. Since Peephole support multiple projects the possibilities are unlimited. The application that has the most in common with Peephole is WebGoat, but in reality there are big differences between them. It is not a real-life application and it is written in Java, which is a much smaller platform than ASP.NET [14].

DISCUSSION

When developing in ASP.NET there are mainly two ways of writing code: WebForms and Razor (MVC). Since we are developing a modular application, we want to target as many developers as possible. We weighed the options we had which was to stay with either WebForms or Razor and have more time to build the application itself, or to use some extra time to implement both in our solution and support a broader audience of possible developers. We came to the decision that implementing both made more sense in the long run since the application is made modular and easily extensible.

When we looked into making the application modular, Visual Studio and ASP.NET projects come with a few built in possible ways to do modularity. The two main methods we have discussed in the context of development were “Areas” and “Multiple Project Solution”. Areas are simply subfolders within the project which contain all the files necessary for building individual parts of a project, where Multiple Project Solutions offer complete customization of project type and file structure and works as a different project within the same solution. Having Areas in a project meant that we could extend a web application with standalone parts, however, having Multiple Project Solution meant that other developers could make completely different applications in our solution, opening up for huge extensions. For example, we made a Bank application to demonstrate the vulnerabilities. Some vulnerabilities might be best demonstrated under different circumstances, which then maybe requires a different project to do so, and with Multiple Project Solution that can be done easily.
We decided to implement both ways of modularity since they both offered great advantages to our solution. Our solution now contains two projects and multiple areas for handling the content. We nest all the different projects together on a main start up page which lists all projects available in the solution.

CONCLUSION

Since we are making an Open Source application, we want people to get the files. We made a GitHub repository which anyone can clone and develop or play with. If you are familiar with Webforms or with MVC, you could easily use whatever you want, since the application supports both. If you have a great idea for a project, it is easy to extend the Peephole with a complete new project. Peephole includes all the OWASP Top 10 vulnerabilities out of the box.

ACKNOWLEDGMENTS

Thanks to:
Our supervisor Girts Strazdins
The person who made this project possible; Juan J. Güelfo at Encripto AS.

8.6 Attachment 6: Backlog from JIRA

This does not include subtasks, only mainstasks

Main task	Date created
Thesis - Architecture structure - ViewEngine	27/Apr/16
Thesis - Vulnerability - Unvalidated Redirect	27/Apr/16
Thesis - Vulnerability - Insufficient Transpart layer transportation	27/Apr/16
Thesis - Vulnerability - Failure to resrict URL access	27/Apr/16
Thesis - Vulnerability - Insecure cryptographic storage	27/Apr/16
Thesis - Vulnerability - Security Misconfiguration	27/Apr/16
Thesis - Vulnerability - Cross-site Request forgery	27/Apr/16
Thesis - Vulnerability - Insecure Direct objefct references	27/Apr/16
Thesis - Vulnerability - Broken authentication	27/Apr/16
Thesis - Vulnerability - XSS	27/Apr/16
Thesis - Vulnerability - Injection	27/Apr/16
Thesis - Modular	27/Apr/16
Thesis - Architecture structure	27/Apr/16
Make video / presentation for Encripto	26/Apr/16
Write about the name "peephole"	22/Apr/16
Change a current db to be the main db and delete the others. Fix stored procedures	22/Apr/16
Document source code	21/Apr/16
Add content/documentation for each page	21/Apr/16
Add hiddenContent to the pages that needs it	21/Apr/16
Add some new pictures (stock photos) to the fotorama	21/Apr/16
Menu sorting	21/Apr/16
Write about design in thesis	21/Apr/16
Make design documentation	21/Apr/16
Unvalidated redirect	15/Apr/16
Inscure Direct Object References	06/Apr/16
jQuery that saves checkbox state	06/Apr/16
Simple hashing on passwords	06/Apr/16
Information page	06/Apr/16
New login version 3	06/Apr/16
Menu control	06/Apr/16
Create menu	06/Apr/16
Documentation module template	06/Apr/16

Final decition on how to do modules	26/Feb/16
Write more on the bachelor thesis	26/Feb/16
PEEPHOLE-17 Both?	26/Feb/16
PEEPHOLE-17 WEb form?	26/Feb/16
PEEPHOLE-17 MVC?	26/Feb/16
What kind of strucure are we going for?	26/Feb/16
Module template	22/Feb/16
Bachelor thesis	22/Feb/16
PEEPHOLE-11 Hvordan bruke flere .NET versjoner	18/Feb/16
Implement set-up method	12/Feb/16
Find out which questions should be answered in the thesis	12/Feb/16
Try out Conflucence and JIRA	12/Feb/16
Finalize planned design	12/Feb/16
Make a module layout template	12/Feb/16
Make functional SQL module	12/Feb/16
Make functional XSS module	12/Feb/16
Reset module db from settings	12/Feb/16
Module guideline example	12/Feb/16

8.7 Attachment 7: Documentation

Peephole Documentation

This is the documentation paper for the Peephole solution. In this document you can find information on how to extend Peephole Bank and also how to add additional projects into Peephole solution.
Peephole is created as a bachelor project at NTNU Aalesund, Norway. We wanted to create a legal testing environment for information security issues for the ASP.NET platform.
If you have any questions, please feel free to contact us.

VIRTUAL MACHINE SETUP

Requirements

The requirements for the host computer variates depending on the which OS the VM client is running. Below are the minimum requirements for resources that needs to be free and available to the VM:

VM running Windows 8.1

Available disk space	> 20 GB
RAM	> 2 GB
CPU logical cores	Half of total

VM running Windows 10

Available disk space	> 20 GB
RAM	> 4 GB
CPU logical cores	Half of total

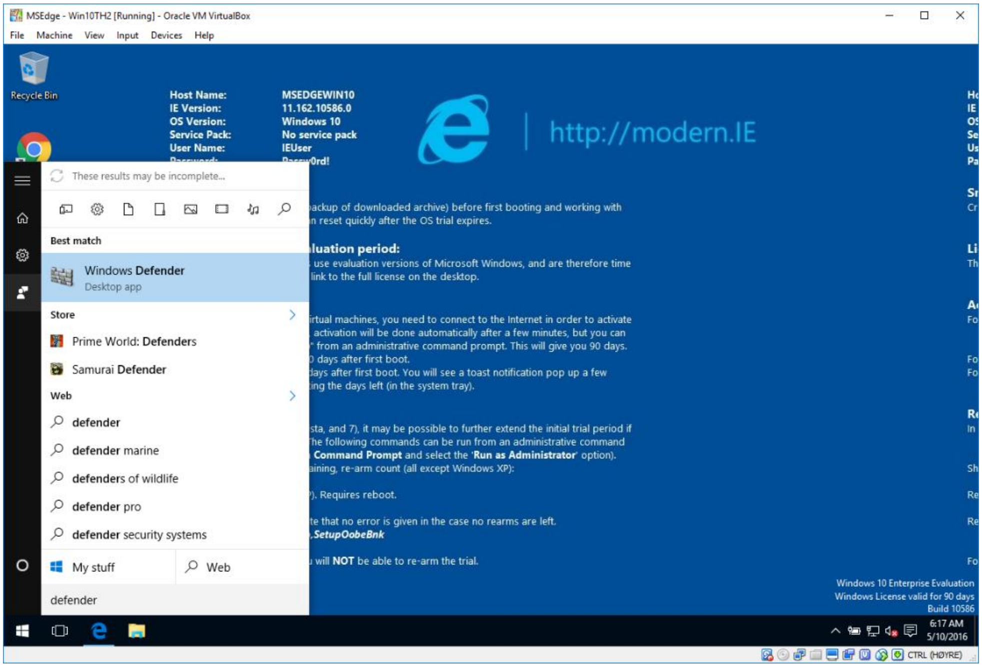
Installation

1. If you plan to run the Peephole project from existing Windows installation, please proceed to step **Feil! Fant ikke referansekilden.**.

2. Download Virtual Machine from https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/windows/

3. Unzip downloaded VM and import in Virtualization software. For VirtualBox, change CPU to 2, memory 4096.

4. Once booted, disable Windows Defender Real-time scan (For performance, can be skipped)



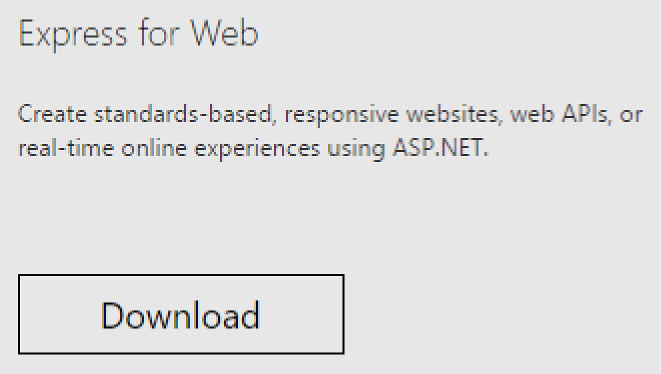
MSEdge - Win10TH2 [Running] - Oracle VM VirtualBox

Host Name:	MSEDGEWIN10
IE Version:	11.162.10586.0
OS Version:	Windows 10
Service Pack:	No service pack
User Name:	IEUser

http://modern.IE

Windows 10 Enterprise Evaluation
Windows License valid for 90 days
Build 10586

5. Inside the VM, go to https://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx, scroll down on the page and download Visual Studio Express for Web

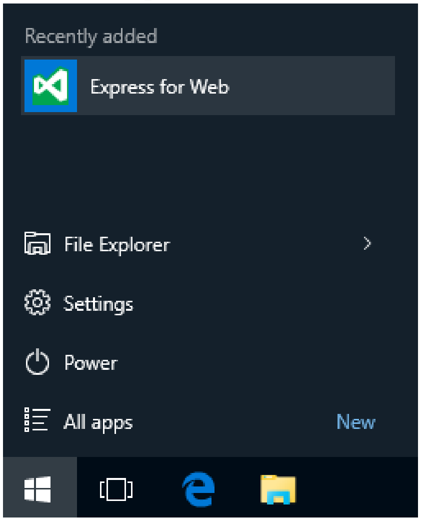


Express for Web

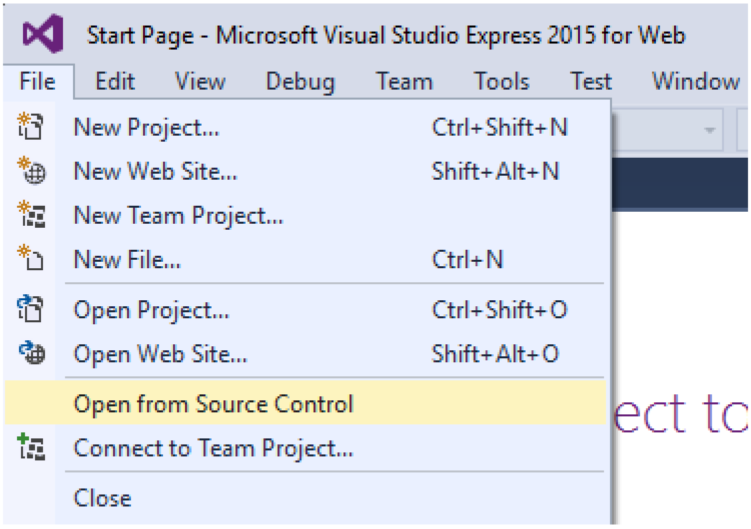
Create standards-based, responsive websites, web APIs, or real-time online experiences using ASP.NET.

6. If you plan to run the Peephole project from existing Visual Studio installation, please proceed to step **Feil! Fant ikke referansekilden.**.

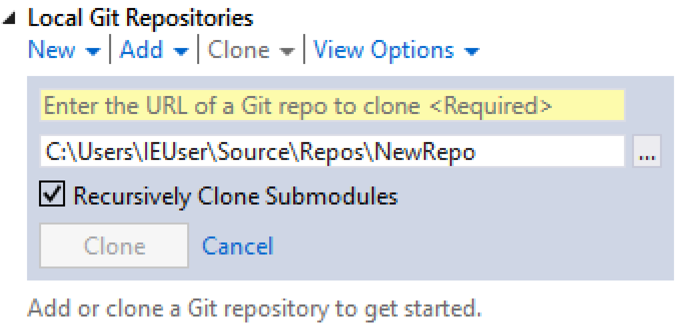
7. Install Visual Studio. This is a time consuming operation, approximately 1 hour depending on system performance.
8. Start -> Express for Web



9. File -> Open from source control


Start Page - Microsoft Visual Studio Express 2015 for Web
File
Edit
View
Debug
Team
Tools
Test
Window
New Project... Ctrl+Shift+N
New Web Site... Shift+Alt+N
New Team Project...
New File... Ctrl+N
Open Project... Ctrl+Shift+O
Open Web Site... Shift+Alt+O
Open from Source Control
Connect to Team Project...
Close
ect tc

10. On the right side, select "Clone"



Enter the URL of a Git repo to clone <Required>

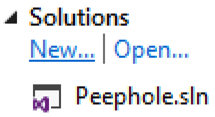
C:\Users\IEUser\Source\Repos\NewRepo

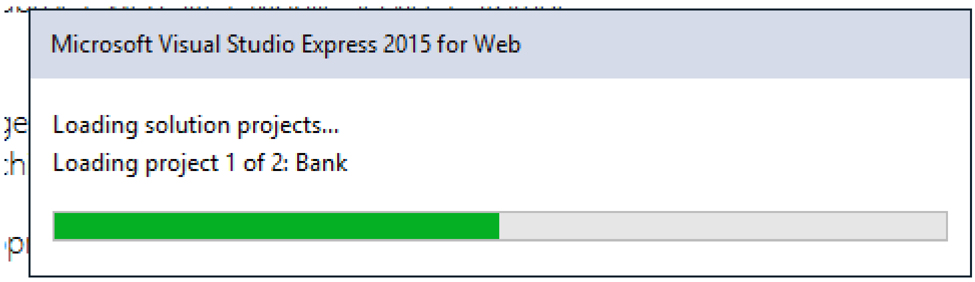
☑ Recursively Clone Submodules

Add or clone a Git repository to get started.

11. Enter the Peephole repository
    https://github.com/Urdar/Peephole.git

12. Select Peephole.sln and the project will load



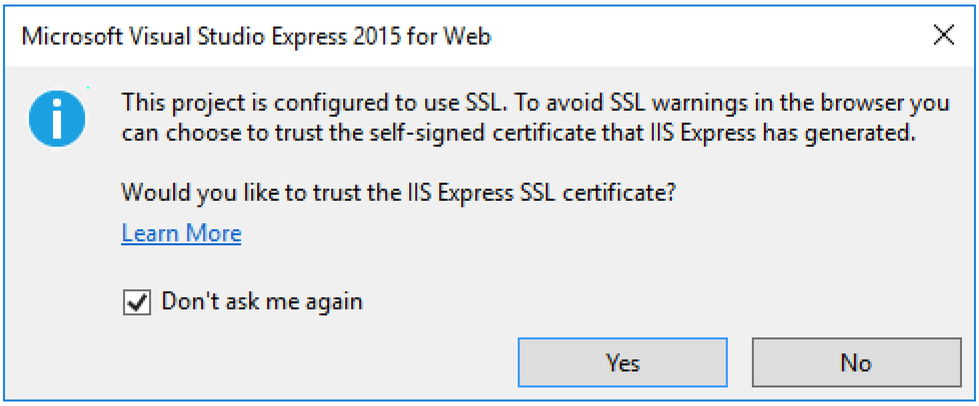


Microsoft Visual Studio Express 2015 for Web

Loading solution projects...

Loading project 1 of 2: Bank

13. On the first run, click Yes to create a local SSL certificate



Microsoft Visual Studio Express 2015 for Web

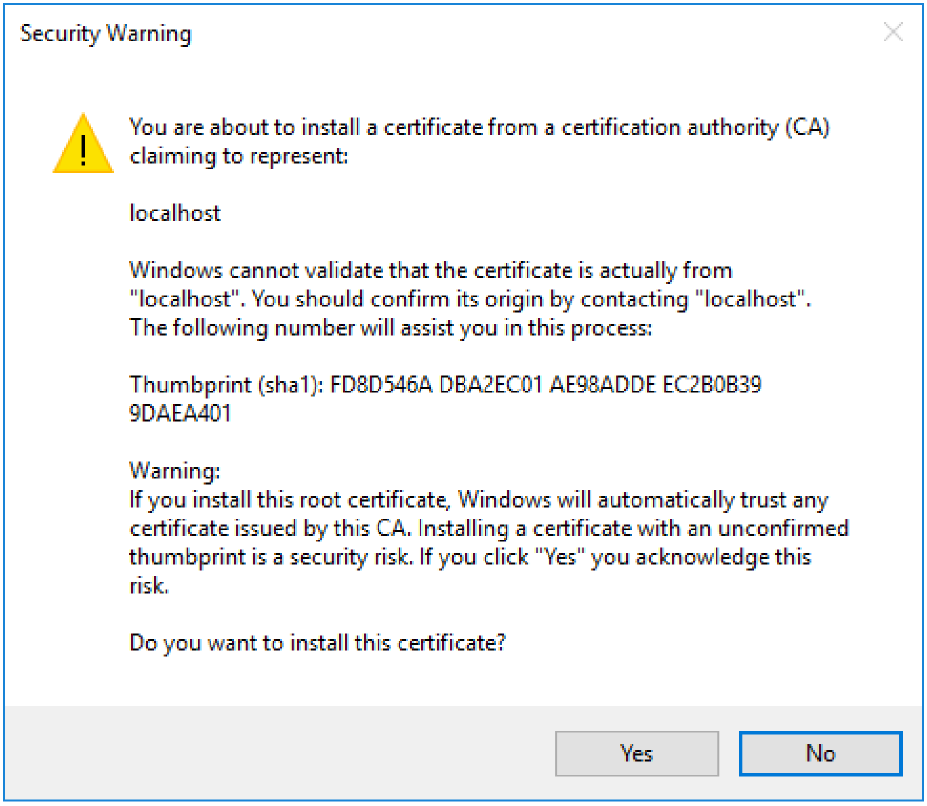
This project is configured to use SSL. To avoid SSL warnings in the browser you can choose to trust the self-signed certificate that IIS Express has generated.

Would you like to trust the IIS Express SSL certificate?

Learn More

☑ Don't ask me again

14. And Yes again



Security Warning

You are about to install a certificate from a certification authority (CA) claiming to represent:

localhost

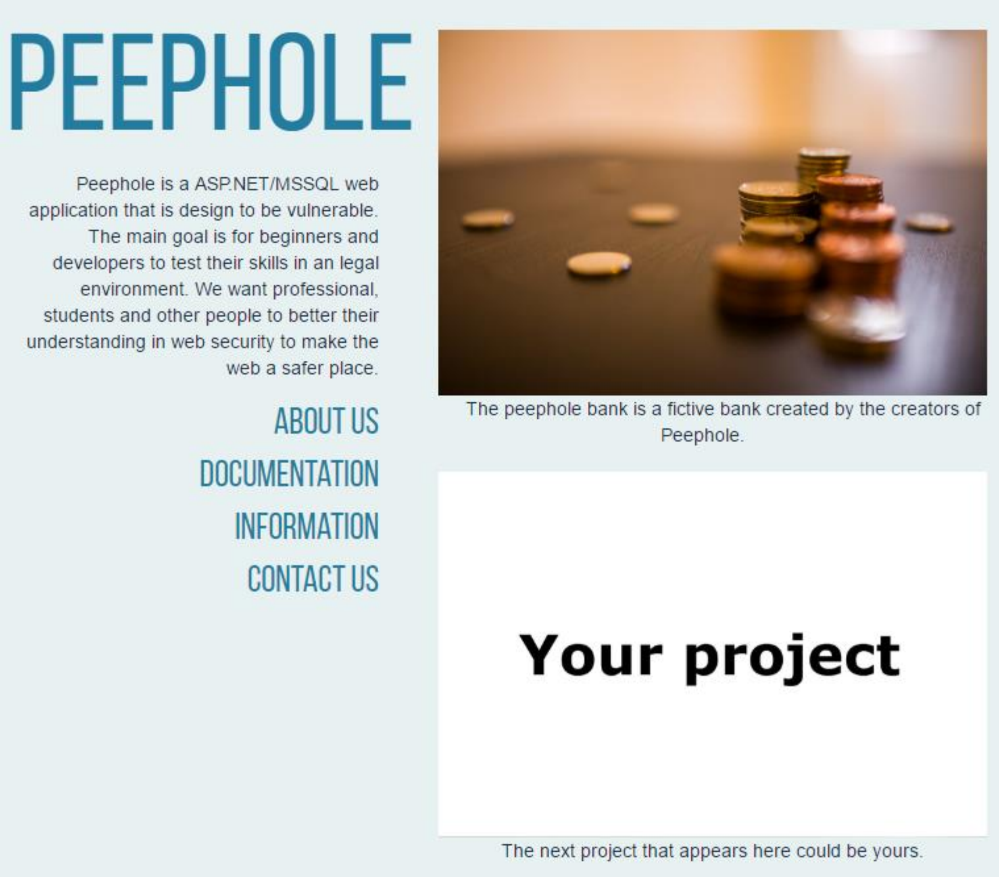
Windows cannot validate that the certificate is actually from "localhost". You should confirm its origin by contacting "localhost". The following number will assist you in this process:

Thumbprint (sha1): FD8D546A DBA2EC01 AE98ADDE EC2B0B39 9DAEA401

Warning:
If you install this root certificate, Windows will automatically trust any certificate issued by this CA. Installing a certificate with an unconfirmed thumbprint is a security risk. If you click "Yes" you acknowledge this risk.

Do you want to install this certificate?

15. A browser will be opened showing the Peephole Frontpage



PEEPHOLE

Peephole is a ASP.NET/MSSQL web application that is design to be vulnerable. The main goal is for beginners and developers to test their skills in an legal environment. We want professional, students and other people to better their understanding in web security to make the web a safer place.

The peephole bank is a fictive bank created by the creators of Peephole.

Your project

The next project that appears here could be yours.

EXTENDING PEEPHOLE SOLUTION

Create new project

The solution can be extended with additional projects.

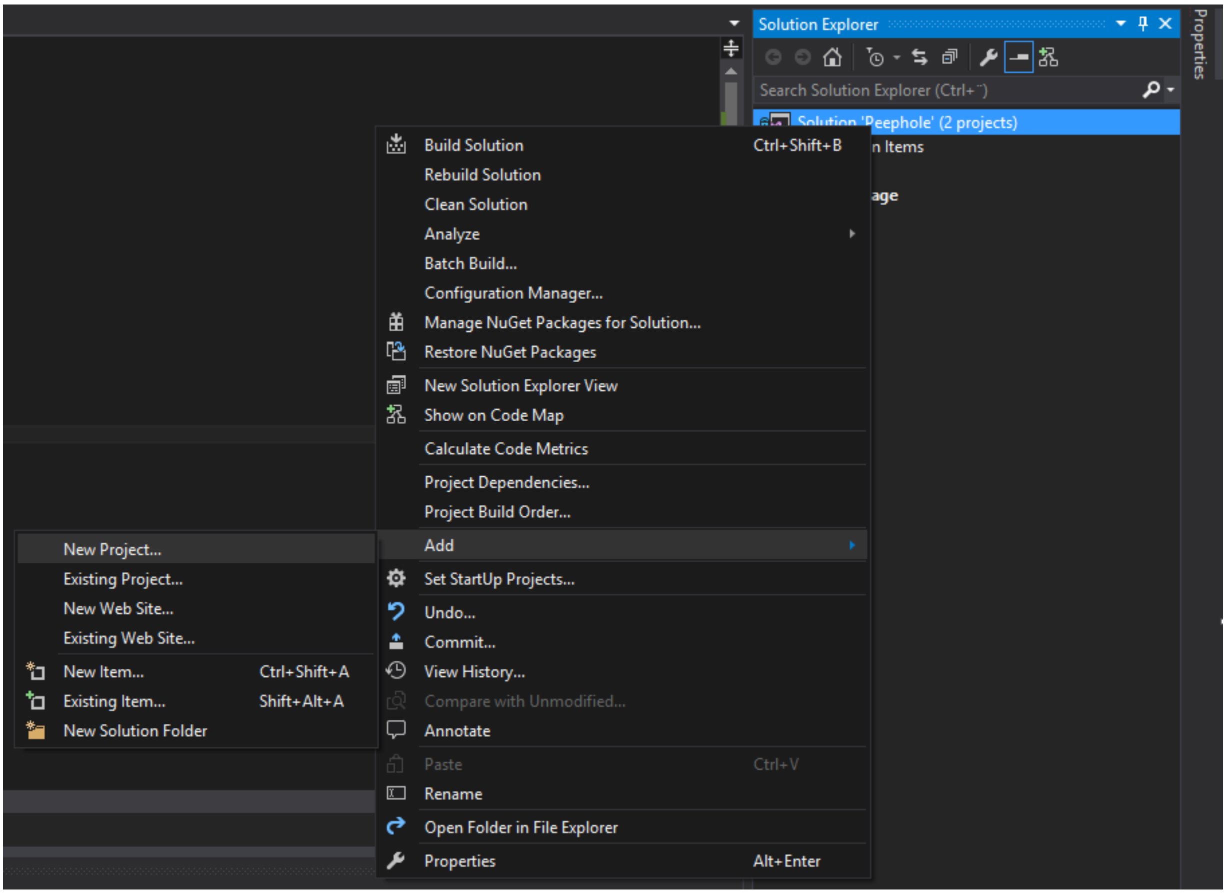


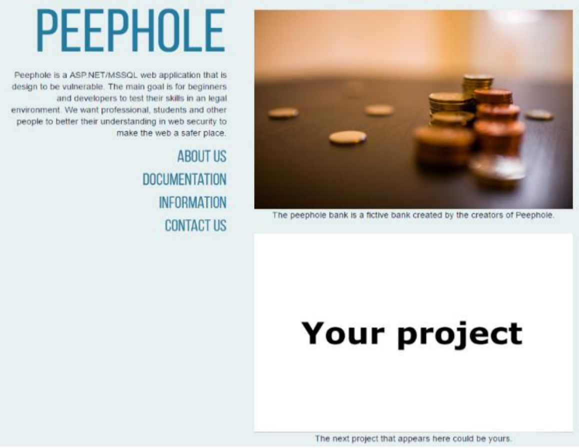
Figure 8.2: Adding Project To Solution

Step by step:

1. Right click the solution
2. Choose “Add”
3. Choose “New project”
4. Select the type of project you want to create
5. Change the application settings
    a. Right click the project
    b. Choose “Properties”
        i. *Application:* set the namespace
        ii. *Web:* set the localhost address and start-up page)
6. For accessibility, add a link to the new project from the Frontpage-project ()

Adding link to project in the Frontpage

When adding a new project to the solution you should also make a link to that project in the Frontpage. The Frontpage is the start-up project for the solution and functions as a hub where projects are accessed.



PEEPHOLE

Peephole is a ASP.NET/MSSQL web application that is design to be vulnerable. The main goal is for beginners and developers to test their skills in an legal environment. We want professional, students and other people to better their understanding in web security to make the web a safer place.

ABOUT US

DOCUMENTATION

INFORMATION

CONTACT US

The peephole bank is a fictive bank created by the creators of Peephole.

Your project

The next project that appears here could be yours.

Figure 8.3: How the frontpage and the different project looks like

Each of the application have one a clickable picture each that shows what kind of application it is. When hovering over the picture, you will get the application name and a short description of the project. When adding a new project, you will have to either edit or add a few links of HTML code to make a new picture for your project. In this code you also have to edit the header, project name and a description of the of the project.

In the Bank application

Header: Peephole

Project: Bank

Description: The Peephole Bank is a fictive bank created by the creators of Peephole.

Figure 8.4: Project image when hoovering

```
        <div class="content col-md-9 col-sm-9 col-lg-9">
                 <a href="http://localhost:3988/" class="portfolio-box">
                     <!-- Add image-->
                     <img src="img/bank.jpg" class="img-responsive" alt="">
                     <div class="portfolio-box-caption">
                         <div class="portfolio-box-caption-content">
                             <div class="project-category text-faded">
                                 <!-- Header-->
                                 Peephole
                             </div>
                             <!-- Project name-->
                             <div class="project-name">
                                 Bank
                             </div>
                         </div>
                     </div>
                 </a>
                 <span class="description col-md-12 col-sm-12 col-lg-12">
                  The Peephole Bank is a fictive bank within the Peephole solution.
                 </span>
             </div>
```

Figure 8.5: Code to create project image in Frontpage / Index.html

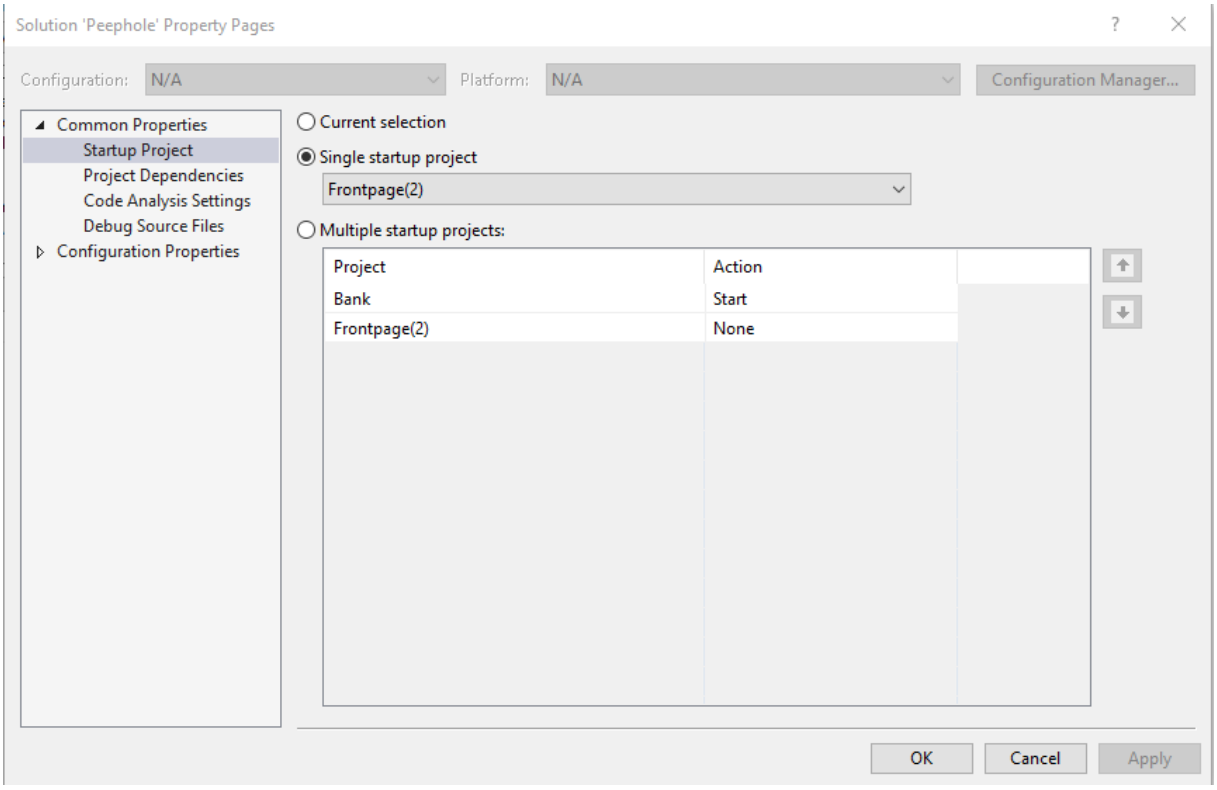
Feil! Fant ikke referansekilden. shows the code that makes a new project picture with the text needed. This uses some of the bootstrap framework to style the pictures and the structure.

If you want to use another image, this is also where this is edited. It is possible to add an image from the web, but it is preferable to save the image locally.

Default Project in Solution

In the solution you can also choose which project should run when you start the solution.

1. Right click the solution
2. Choose “properties”
3. Choose “Startup Project” under “Common Properties”
4. Now you can choose which project should start when you press the Start button in Visual Studio



Configuration: N/A

Platform: N/A

○ Current selection

● Single startup project

Frontpage(2)

○ Multiple startup projects:

Project	Action
Bank	Start
Frontpage(2)	None

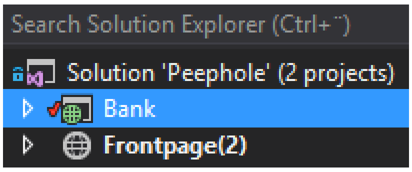
Figure 8.6: Choosing which project should start by default

EXTENDING PEEPHOLE BANK

Peephole Bank is the first project in the Peephole Solution.

Adding modules / Areas

Extending "Peephole Bank" with Areas



Search Solution Explorer (Ctrl+¨)

Solution 'Peephole' (2 projects)

Bank

Frontpage(2)

1. View the file structure of the project
2. Right click on "Areas"
3. Choose "Add"
4. Choose "Area"
5. Write desired area name, which must be unique.
6. The new Area will have MVC structure by default. If Webforms is to be preferred,

MVC Area Specifics

To use shared design in Site.master, in your controllers, you have to replace

```
return View();
```

with

```
return this.RazorView();
```

Webforms Area Specifics

If you intend to use Webforms for the new Area, please delete the folders "Controllers", "Models" and "Views". These are only used when using the MCV structure.
Leave the *AreaRegistration.cs file intact. The AreaRegistration file will be adapted into registering the Area into the Peephole database / menu.
An example Webforms area is provided in Bank/Areas/_WebformsTemplate.

Adding the Area/Module To Menu

In the *AreaRegistration.cs file you need to add a line of code to make it appear the in the menu. Beneath the code below

```
context.MapRoute(
            "MVCTemplate_default",
            "MVCTemplate/{controller}/{action}/{id}",
            new { action = "Index", id = UrlParameter.Optional }
         );
```

Add this line of code:

```
AreasHandling.insertInMenu(context.AreaName);
```

The result would be:

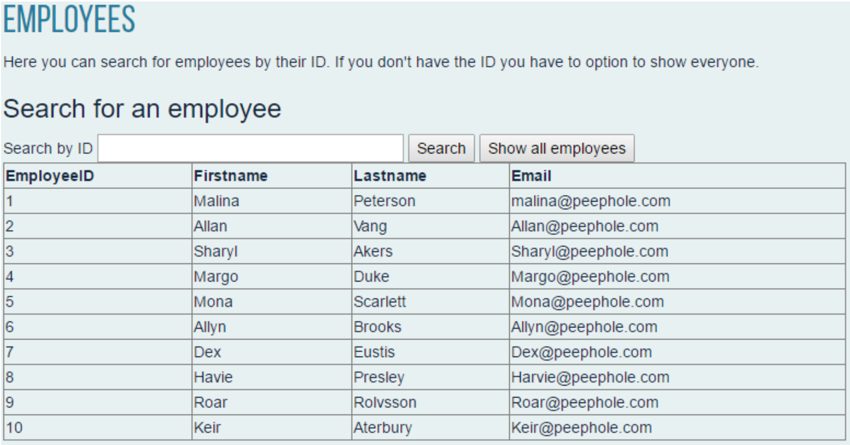
```
public override void RegisterArea(AreaRegistrationContext context)
      {
         context.MapRoute(
            "MVCTemplate_default",
            "MVCTemplate/{controller}/{action}/{id}",
            new { action = "Index", id = UrlParameter.Optional }
         );
AreasHandling.insertInMenu(context.AreaName);
      }
```

This applies to both MVC and Webforms Areas.

USING THE APPLICATION

Injection

The injection in our Bank application is an SQL-injection done when you search for employees, but you insert a *malicious* query which the database interprets like a query to do something else than searching.



EMPLOYEES

Here you can search for employees by their ID. If you don't have the ID you have to option to show everyone.

Search for an employee

EmployeeID	Firstname	Lastname	Email
1	Malina	Peterson	malina@peephole.com
2	Allan	Vang	Allan@peephole.com
3	Sharyl	Akers	Sharyl@peephole.com
4	Margo	Duke	Margo@peephole.com
5	Mona	Scarlett	Mona@peephole.com
6	Allyn	Brooks	Allyn@peephole.com
7	Dex	Eustis	Dex@peephole.com
8	Havie	Presley	Harvie@peephole.com
9	Roar	Rolvsson	Roar@peephole.com
10	Keir	Aterbury	Keir@peephole.com

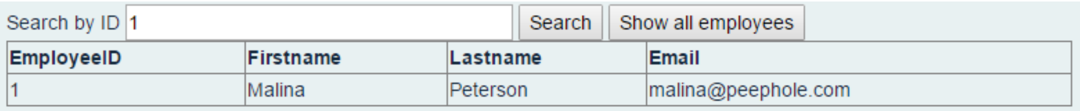
Screenshot 5: Show all employees

When using the “Employee”-site normally, you will get a list of all the employees with their ID, firstname, lastname and email. This is what you would except from this kind of site.

Since this website is vulnerable the search-field could be inserted with malicious queries which makes the database change or display information which we have intended. This is because the code is poorly written.

```
var sqlString = "SELECT * FROM Users WHERE ID = " + ID;
```

This is how the query on the server-side looks like. This means that if you write the number “1” in the search-field, you would return only “Malina Peterson” like this:



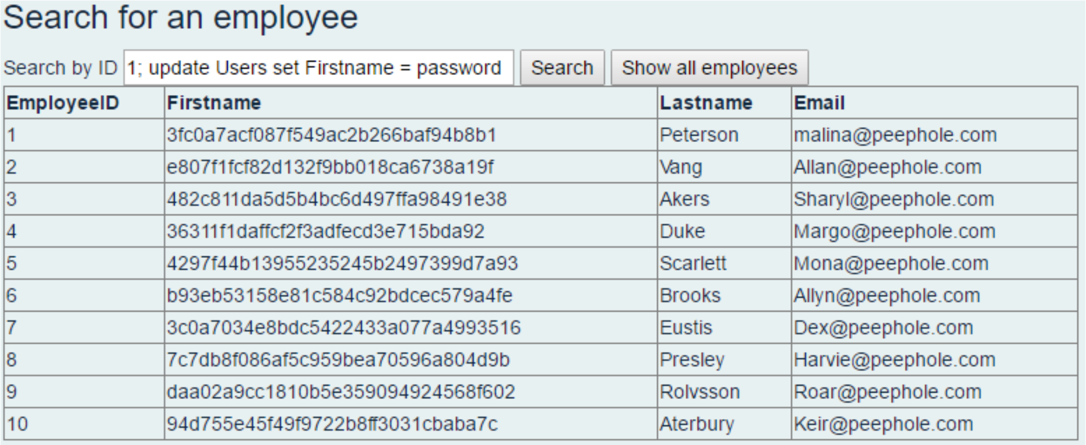
Search by ID 1

EmployeeID	Firstname	Lastname	Email
1	Malina	Peterson	malina@peephole.com

Screenshot 6: Search for specific employee

The database would read the following as `SELECT * FROM Users WHERE ID = 1`

But if you find out names of the other columns in the table, the attacker can retrieve anything from that database.



EmployeeID	Firstname	Lastname	Email
1	3fc0a7acf087f549ac2b266baf94b8b1	Peterson	malina@peephole.com
2	e807f1fcf82d132f9bb018ca6738a19f	Vang	Allan@peephole.com
3	482c811da5d5b4bc6d497ffa98491e38	Akers	Sharyl@peephole.com
4	36311f1daffcf2f3adfecd3e715bda92	Duke	Margo@peephole.com
5	4297f44b13955235245b2497399d7a93	Scarlett	Mona@peephole.com
6	b93eb53158e81c584c92bdcec579a4fe	Brooks	Allyn@peephole.com
7	3c0a7034e8bdc5422433a077a4993516	Eustis	Dex@peephole.com
8	7c7db8f086af5c959bea70596a804d9b	Presley	Harvie@peephole.com
9	daa02a9cc1810b5e359094924568f602	Rolvsson	Roar@peephole.com
10	94d755e45f49f9722b8ff3031cbaba7c	Aterbury	Keir@peephole.com

Screenshot 7: SQL-injection. Firstname is swapped with password

The database would read the following as SELECT * FROM Users WHERE ID = 1; UPDATE Users SET Firstname = Password

Now the attacker has the passwords hashed. Since this is hashed with MD5, cracking it would take seconds by using a rainbow table.

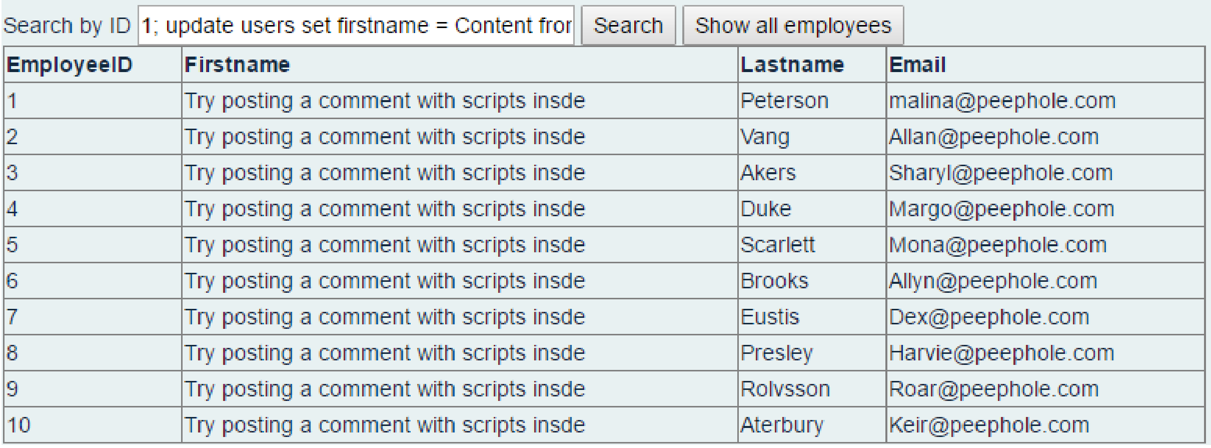
This could be fixed by checking if the input is other than numbers in this example (if the search is done by text, filter the text by symbols like ;'@-.

```
var positiveIntRegex = new Regex(@"^0*[1-9][0-9]*$");
        if (!positiveIntRegex.IsMatch(ID))
        {
           //Code
        }

        else {
           //Code
        }
```

This code allows only number to be put into the search-field, making an SQL-injection harder to do.

Since all the tables is stored on the same database, it is possible to retrieve information from other tables because they are in the same database and you can input anything into the search field.



EmployeeID	Firstname	Lastname	Email
1	Try posting a comment with scripts insde	Peterson	malina@peephole.com
2	Try posting a comment with scripts insde	Vang	Allan@peephole.com
3	Try posting a comment with scripts insde	Akers	Sharyl@peephole.com
4	Try posting a comment with scripts insde	Duke	Margo@peephole.com
5	Try posting a comment with scripts insde	Scarlett	Mona@peephole.com
6	Try posting a comment with scripts insde	Brooks	Allyn@peephole.com
7	Try posting a comment with scripts insde	Eustis	Dex@peephole.com
8	Try posting a comment with scripts insde	Presley	Harvie@peephole.com
9	Try posting a comment with scripts insde	Rolvsson	Roar@peephole.com
10	Try posting a comment with scripts insde	Aterbury	Keir@peephole.com

Screenshot 8: Content from another table

There also a lot of new ways to display information from a database. One of these way is using ORM which is a technique converting data between type systems and object-oriented programming languages. The most commonly used in ASP.NET is Entity Framework[30]. This eliminates the need for most of the data-access code for the developers and that means it is less vulnerable because of the lack of human mistakes.

EF is one of the most secure ways of displaying and using database information, but there is other ways:

- LINQ[31]
- Parametrized queries[32]
- Regular expression[33] (which is used in the example code above)

All these have been documented by Microsoft with tutorials and examples.

Cross-Site Scripting (XSS)

In the OWASP Top 10 report XSS made it to a third place and a second place in WhiteHats report. Based on these reports we chose to implement the XSS vulnerability in our web application.

As mentioned above, there are three types of XSS. In our web application we chose to implement two of them; Reflected and Stored XSS. This decision was made due to time limitations and that we already strongly represented XSS in our application.

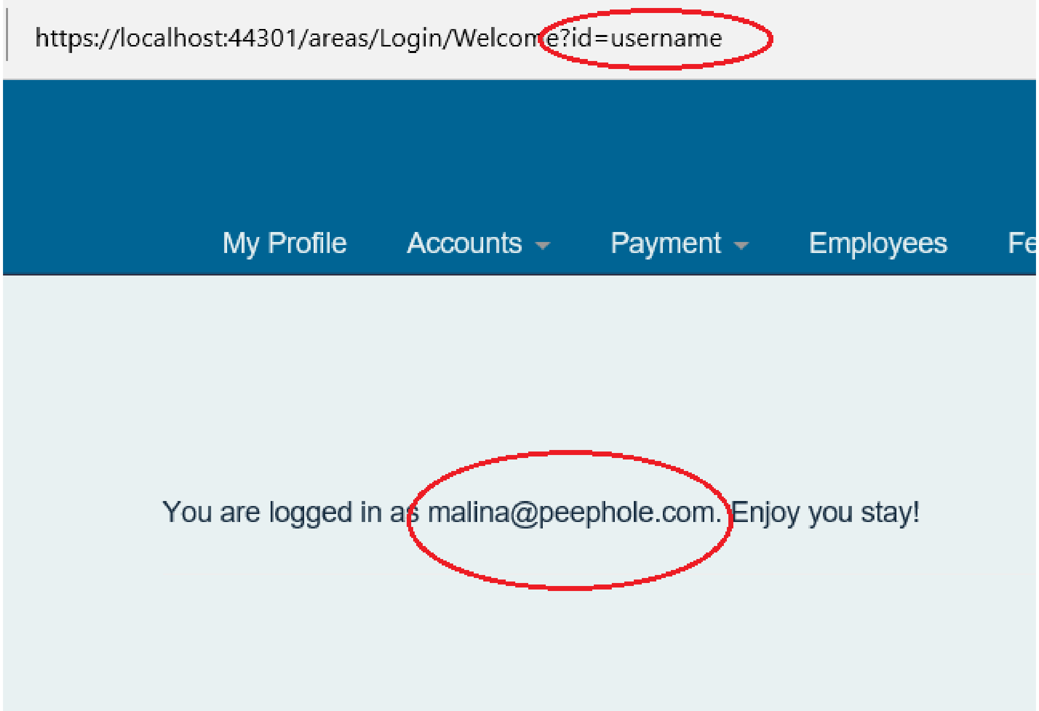
We are making the application look more like a real example and less like a “go-to-vulnerability” kind of application. The user is presented with links and functionality usually found in a bank, so we had to find natural points to implement the vulnerabilities. Reflected XSS can be found and only accessed after a user has logged in. After the user has been authenticated he is redirected to a welcome page, this is where we show a static welcome message along with the username in the form of a variable seen in the URL. Look at the below picture for an example:

[30] http://www.asp.net/entity-framework

[31] https://msdn.microsoft.com/en-us/library/bb907622%28v=vs.100%29.aspx?f=255&MSPPError=-2147217396

[32] http://www.asp.net/web-forms/overview/data-access/accessing-the-database-directly-from-an-aspnet-page/using-parameterized-queries-with-the-sqldatasource-vb

[33] https://msdn.microsoft.com/en-us/library/ms972966.aspx

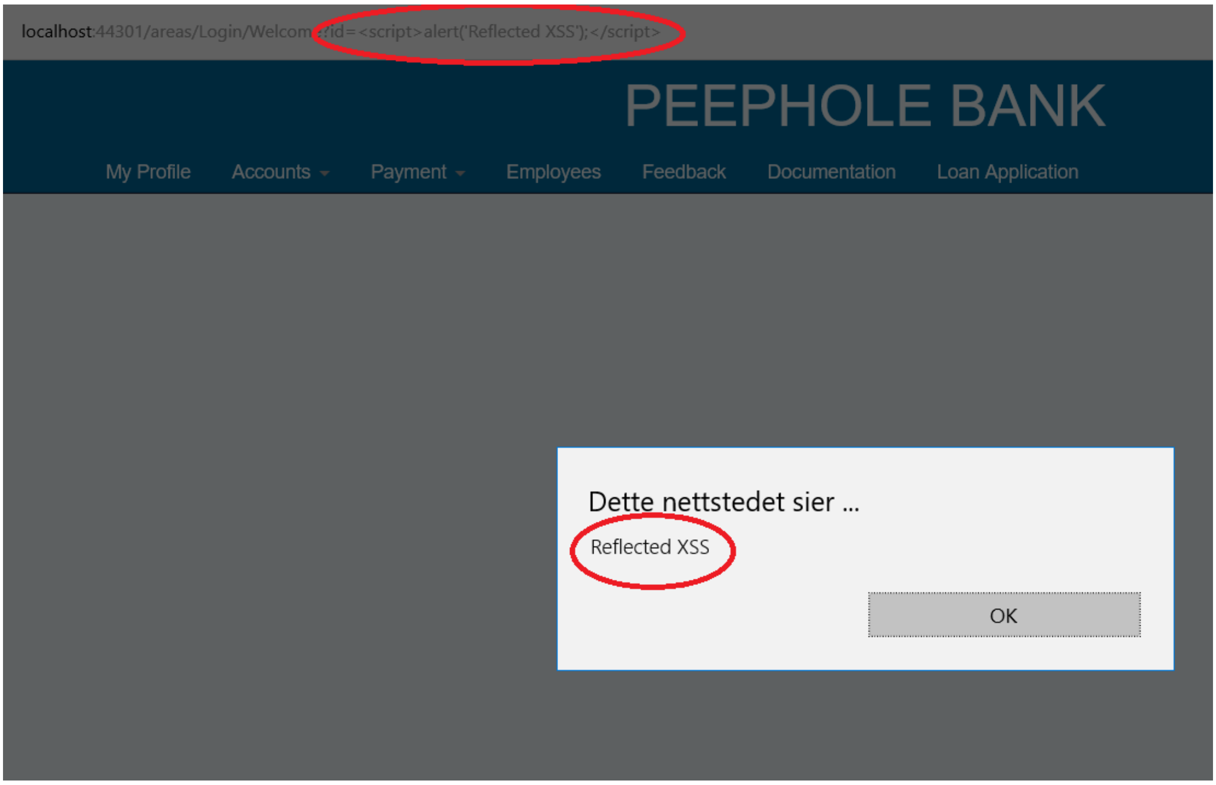


https://localhost:44301/areas/Login/Welcome?id=username

You are logged in as malina@peephole.com. Enjoy you stay!

Screenshot 9- Screenshot showing that the username is reflected back on the site through a variable in the URL

In order to exploit this vulnerability, you can put malicious code directly into the URL field. Most of the big browsers have their own filtering of such vulnerabilities by checking the URL input and filtering out specific input. Some browser however, does not filter such input. Among those are Internet Explorer and Microsoft Edge. This is why the examples are shown in Microsoft Edge.



localhost:44301/areas/Login/Welcome?id=<script>alert('Reflected XSS');</script>

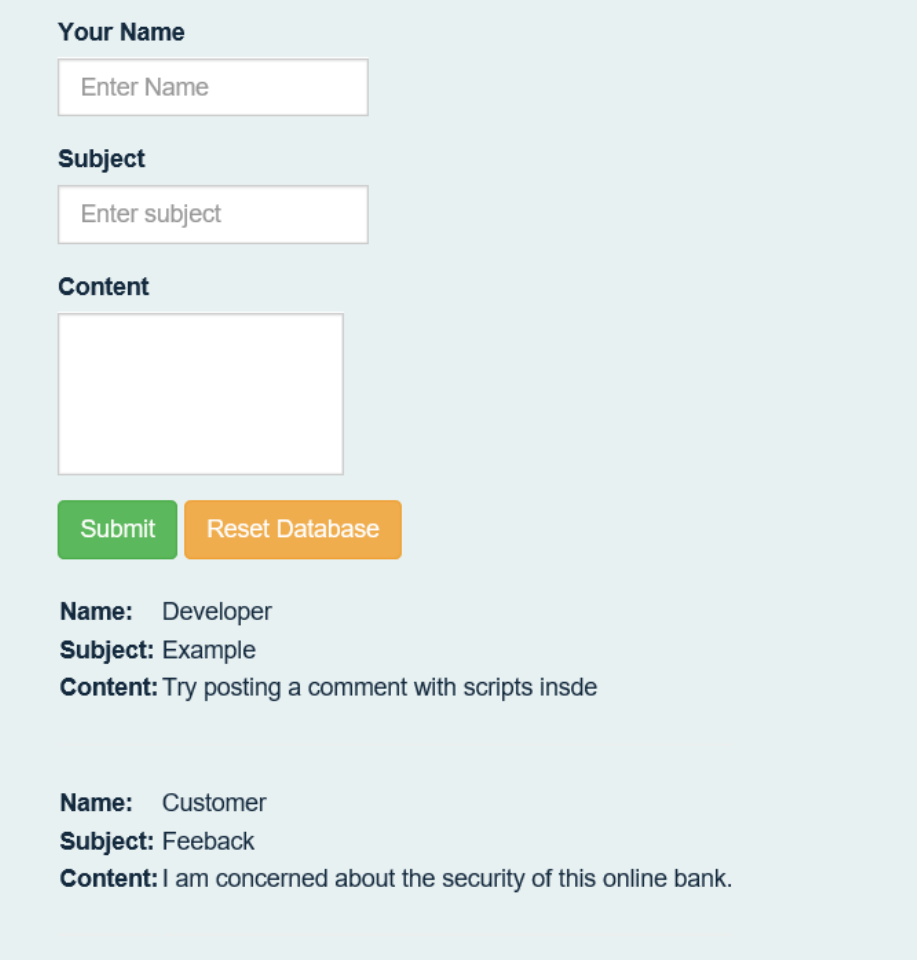
Dette nettstedet sier ...

Reflected XSS

Screenshot 10- Screenshot showing what happens when we change the variable in the URL to a script

In the picture above we can see that the reflected XSS was successful. The script used here and that can be seen some of in the URL is <script>alert(‘Reflected XSS’);</script>. This is just a simple alert box, but with such a vulnerability you can do an enormous amount of damage. You can for example steal credentials in non-HTTPOnly cookies, send requests to a server with the user's credentials, make the user download content, display a password input, log keystrokes, and send the result to a site of your choosing. These are just a few of the things you can do with reflected XSS. Performing the same example in for example Google Chrome would require to either disable the filtering itself. This however, does not mean it is completely secure. The input can be encoded in a way not yet known to the filter implemented in the browser for example, and the script will run. Protecting against XSS is mainly the developers' responsibility, but most of the big browsers have acknowledged the vulnerability and taken steps to help reduce the risk.

The stored XSS can be found under customer feedback in our application, this is where customers and visitors can leave a comment about the bank or their experience with the bank service. The user input is handled and displayed back to the web application without any kind of validation. Here is an example of a normal comment:

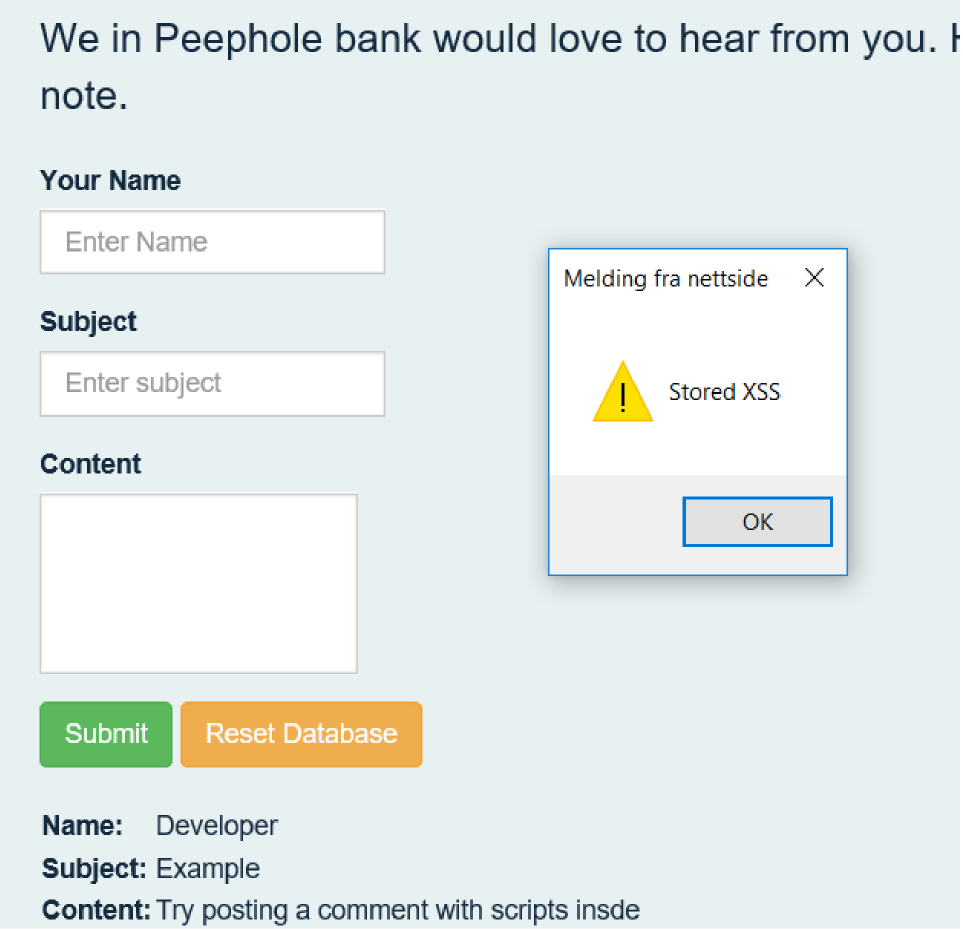


Name: Developer
Subject: Example
Content: Try posting a comment with scripts insde

Name: Customer
Subject: Feeback
Content: I am concerned about the security of this online bank.

Screenshot 11- A normal comment form displaying comments back on the site.

In the picture above we see two normal comments with no harmful code. We also see a button called "Reset Database". This is implemented so that users can try out the vulnerability without concern for breaking the application, as it will be restored to a default state once you push the reset database button. In the next picture we will see what happens when we use the same script as we did in the reflected XSS but with different text inside: <script>alert('Stored XSS');</script>



We in Peephole bank would love to hear from you. I
note.

Name: Developer
Subject: Example
Content: Try posting a comment with scripts insde

Screenshot 12- Here we make a comment with a script. The result is that the script is run on the site.

Here we can see that we get the same result as in the reflected XSS. This time the script is stored into a database and then displayed back onto the site. This means that everyone visiting this specific page after the exploit has been done successfully will run the script.

To prevent XSS in .NET, there are two main countermeasures:

- Constrain input
- Encode output

The .NET framework provides some methods to help prevent XSS, both for constraining input and for encoding output. Using the provided methods where they are needed should help secure against XSS. In ASP.NET version 4.5, Microsoft included AntiXSS based methods to help developer protect against XSS, whereas earlier versions of .NET have an AntiXSS library. Fully preventing cross site scripting is harder than it may seem. OWASP has a list of over 80 vectors that can be targeted using XSS.

Following is an example of XSS on a big social network:
A few years ago there was an XSS vulnerability on Myspace, which a user found out about and created the XSS worm called Samy [69]. MySpace had a filter for “javascript” and a lot of “<tags>” to protect against XSS, but these could be bypassed by altering the format of the words, like “javascript = java\nscript” would not be filtered. The script itself was not that harmful, but worked

as a great example of how dangerous such a vulnerability could be. The XSS worm carried a payload that would display the string “but most of all, samy is my hero” on a victim’s MySpace profile page. When a user viewed that profile page, the payload would then be replicated to their own profile page which resulted in a continuing distribution of the worm. Within 20 hours of its release, over one million users had run the payload, making Samy the fastest spreading virus of all time.

Broken authentication and Session Management

Broken authentication and session management encompasses security flaws to all aspects of user authentication and session management, including but not limited to: Password strength, password storage, weak session IDs, accessible account lists on the website and browser caching.

This vulnerability is represented on several points in our application. When making a default ASP.NET application, Microsoft has a default login with strong password requirements. These requirements included special characters, letters, password length of 8+ characters and upper and lower case characters. For the sake of the application, we chose to make our own custom register and login to better show the security threat with weak passwords, insufficient hashing and storing of passwords along with weak protection. With our weak password requirements, the user can make passwords that are easily guessed as seen in this list of most common passwords of 2015[34]. This list also indicates that there are still many sites out there with a weak password policy.

Insecure Direct Object References

Insecure Direct Object Reference is implemented as a part of the user being able to edit their profile from “My Profile” page accessible from the main menu.

An attacker would create an account, access the edit profile page and inspect the packet sent when submitting the form. The attacker will easily spot the user ID being submitted and suspect the possibility for editing other users accounts simply by changing the ID. The packet can be replayed for ID 1, 2, 3, N, setting the password to whatever he prefers. Once the password has been changed the attacker can simply log in to the victims account and transfer the available money to a bank account of choice.

We have made a YouTube which contains how this works[35]

Cross-Site Request Forgery

Cross-Site Request Forgery (CSRF) is a highly relevant and underestimated vulnerability that we have implemented or rather, not added protection against, on the account money transfer form. The defence against this in .NET is a helper called "AntiForgeryToken". This helper generates a unique ID for each form, which is impossible for an attacker site to guess. This means that when an

[34] http://www.computerworld.com/article/3024404/security/worst-most-common-passwords-for-the-last-5-years.html

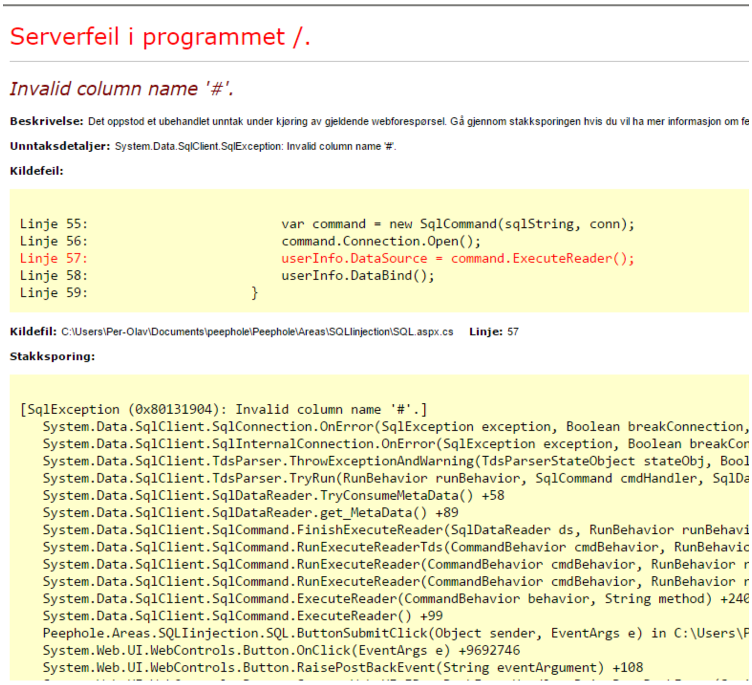
[35] https://youtu.be/jWGMPw2JizI?t=3m22s

attacker site tries to submit a form with the victim's permissions, the targeted site will see that the form comes from an untrusted source since it lacks the unique ID.

Even though our application is vulnerable against CSRF we do not yet have a good way of simulating it. To simulate CSRF you need to have an attacker site, which we have not prioritized due to the restricted development time we have spent on the application.

Security Misconfiguration

Security misconfiguration is a term that describes when any one part of our application stack has not been hardened against possible security vulnerabilities. In OWASP Security misconfiguration is listed at number 5 of their top 10 most critical web application security flaws. When developing an application in .NET as we did, ASP.NET applications can be configured to produce debug binaries which may be extremely helpful when developing, however, when releasing an application, such a configuration can give extremely helpful information to attackers since it displays direct information about the backend of the system. An example to this is improper error handling. When an application crashes without handling the exception, it will display a stack trace and information not meant for users. This information may prove very useful for an attacker wanting to learn more about the system. Our application produces such an error when you try to search for a user and search with special characters.



Serverfeil i programmet /.

Invalid column name '#'.

Beskrivelse: Det oppstod et ubehandlet unntak under kjøring av gjeldende webforespørsel. Gå gjennom stakksporingen hvis du vil ha mer informasjon om fe

Unntaksdetaljer: System.Data.SqlClient.SqlException: Invalid column name '#'.

Kildefeil:

```
Linje 55:                         var command = new SqlCommand(sqlString, conn);
Linje 56:                         command.Connection.Open();
Linje 57:                         userInfo.DataSource = command.ExecuteReader();
Linje 58:                         userInfo.DataBind();
Linje 59:                     }
```

Kildefil: C:\Users\Per-Olav\Documents\peephole\Peephole\Areas\SQLIinjection\SQL.aspx.cs **Linje:** 57

Stakksporing:

```
[SqlException (0x80131904): Invalid column name '#'.]
   System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection,
   System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakCor
   System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Bool
   System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDa
   System.Data.SqlClient.SqlDataReader.TryConsumeMetaData() +58
   System.Data.SqlClient.SqlDataReader.get_MetaData() +89
   System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavi
   System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavic
   System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior r
   System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior r
   System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior, String method) +240
   System.Data.SqlClient.SqlCommand.ExecuteReader() +99
   Peephole.Areas.SQLIinjection.SQL.ButtonSubmitClick(Object sender, EventArgs e) in C:\Users\F
   System.Web.UI.WebControls.Button.OnClick(EventArgs e) +9692746
   System.Web.UI.WebControls.Button.RaisePostBackEvent(String eventArgument) +108
```

Screenshot 13- Showing the error produced by incorrect input when searching for users

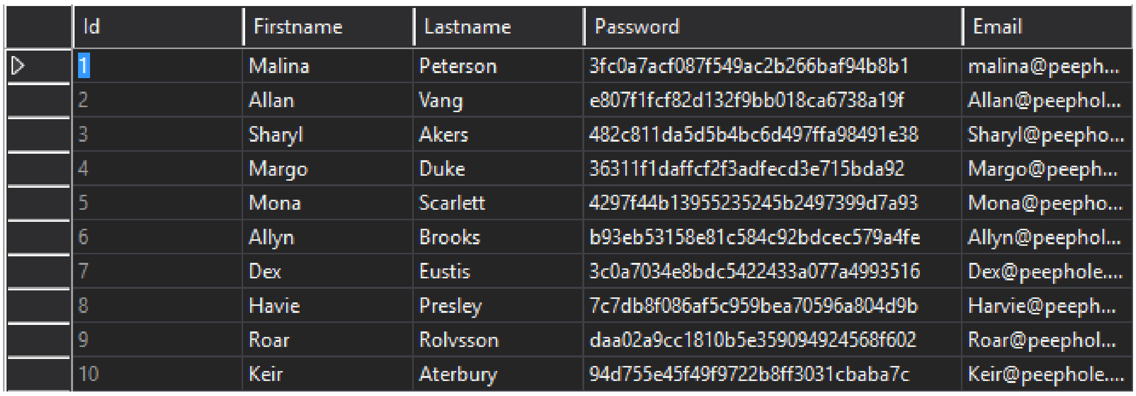
From the error produced above we can see that we got direct contact with the database in the system, this gives us a good indication that the system is vulnerable to SQL-injection.

Insecure Cryptographic Storage

In our project we have used MD5 as the hashing algorithm. This means that that we are using a hashing algorithm that should not be used to protect confidential information (like passwords). It should not be used because it is unsalted and it is fast to decrypt. MD5 is also vulnerable to collision, that means two inputs producing the same hash. In theory that means if the password is "123" that gives a specific hash output, but another string, let's say "Asjdaskdbv879" could give the same hash output and the attacker login to account that has the password "123" with "Asjdaskdbv879".

There is also MD5 rainbow tables available online, which makes cracking MD5 hashes incredible easy, for instance www.md5cracker.org.

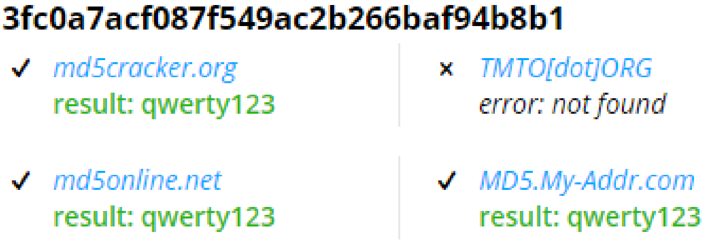
If you do an SQL injection like we have done in 4.3.2 Injection, you will get the hashed password as an output. Insert one of those hashes into MD5cracker.org and you will have the password in clear text. After you have the password, you can proceed to logging in with the username and password.



	Id	Firstname	Lastname	Password	Email
▷	1	Malina	Peterson	3fc0a7acf087f549ac2b266baf94b8b1	malina@peeph...
	2	Allan	Vang	e807f1fcf82d132f9bb018ca6738a19f	Allan@peephol...
	3	Sharyl	Akers	482c811da5d5b4bc6d497ffa98491e38	Sharyl@peepho...
	4	Margo	Duke	36311f1daffcf2f3adfecd3e715bda92	Margo@peeph...
	5	Mona	Scarlett	4297f44b13955235245b2497399d7a93	Mona@peepho...
	6	Allyn	Brooks	b93eb53158e81c584c92bdcec579a4fe	Allyn@peephol...
	7	Dex	Eustis	3c0a7034e8bdc5422433a077a4993516	Dex@peephole....
	8	Havie	Presley	7c7db8f086af5c959bea70596a804d9b	Harvie@peeph...
	9	Roar	Rolvsson	daa02a9cc1810b5e359094924568f602	Roar@peephol...
	10	Keir	Aterbury	94d755e45f49f9722b8ff3031cbaba7c	Keir@peephole....

Screenshot 14: The passwords is stored in the database as hashed passwords. This got to be secure, right?

If we try taking the first hashed password into MD5cracker.org we will see the what the password actually is.
As I mentioned this works for MD5, but there are also rainbow tables for a lot of other hashing algorithms such as SHA1 and LM hash.



3fc0a7acf087f549ac2b266baf94b8b1

✓ md5cracker.org
result: qwerty123

× TMTO[dot]ORG
error: not found

✓ md5online.net
result: qwerty123

✓ MD5.My-Addr.com
result: qwerty123

Screenshot 15: The result of the MD5 crack. This took less than a second.

The best solution for hashing now is the SHA3, which was released in August 2015 by NIST and FIPS. This is more secure because cracking it takes long a long time and there is no collision. Microsoft has not yet created a SHA3 hashing method, but they have one for SHA2-256[36].

Failure to Restrict URL Access Jens

Failure to restrict URL access is implemented several places in Peephole Bank.

My Profile

[36] https://msdn.microsoft.com/en-us/library/system.security.cryptography.sha256%28v=vs.110%29.aspx?f=255&MSPPError=-2147217396

When accessing “My Profile” the user is directed to URL https://localhost:44301/CustomerInfo/Default/Edit/1

An attacker can simply change the last number in the URL, a number representing user ID, to edit some other person’s account

Edit bank account

The URL presented to the user when editing a bank account is https://localhost:44301/TransferFunds/BankAccounts/Edit/3994045019

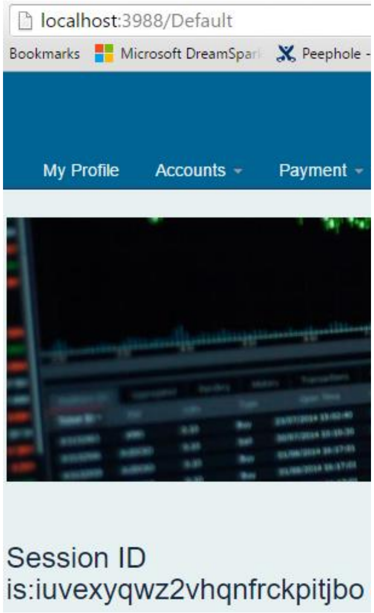
While not being the most serious vulnerability, an attacker could change the bank account name and type by guessing another person’s bank account number in the URL. As all account increment by 1 from 3994.04.5000 other people’s account numbers are easily discoverable and the information could be used for instance to increase the success rate of a spear phishing attack.

Insufficient Transport Layer Protection

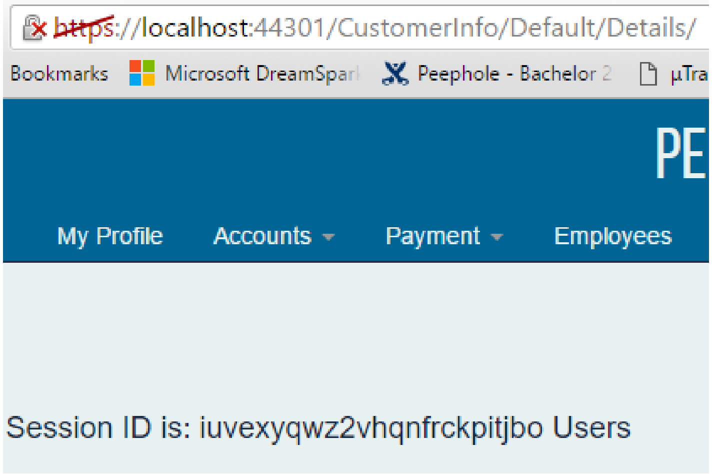
SSL is only partly implemented in Peephole Bank. As the user first enters the bank the session ID is created and transmitted in cleartext over unprotected HTTP. The transmitted information does not get encryption from HTTPS until the user logs in. The session ID remains the same leaving a severe risk of session hijacking.

The attacker could simply capture the transmitted packets when the victim is connected to an unencrypted public Wi-Fi and get the hold of the session ID. Once the victim has authenticated against the bank the attacker holds a valid, authenticated session and can perform whatever actions the victim is authenticated to do.

Two-factor authentication or any other level of secure user authentication would not help as long as the authenticated session ID is up for a grab.



Session ID is:iuvexyqwz2vhqnfrckpitjbo



Session ID is: iuvexyqwz2vhqnfrckpitjbo Users

Screenshot 16: Session ID when is logged in, HTTPS

Screenshot 17: Session ID when user we have logged in, HTTP

Unvalidated Redirects and Forwards

Once this site is loaded it just redirect to another site and this pretty much happens instant. That means that the user does not even notice it.
What the site does is getting a single parameter from the URL which is redirects to. Here an example
http://localhost:3988/Redirect/Redirect/Redirect?url=http://www.ntnu.no
What this does is first accessing the Redirect-module, checking the URL for the parameter called "url" (…?url=http://www.ntnu.no) and redirect to that page. The code that does this is simple:

```
string url = Request.QueryString["url"];
    Response.Redirect(url);
```

But how can this be used by an attacker's points of view? Because the parameter is in the URL, the attacker can change this before sending it to the victim. If the victim trusts the site that redirects (in this case localhost:3988), he/she probably would not check the whole URL. If the attacker changes the parameter to "http://www.malware.com" the victim most probably would not notice.
http://localhost:3988/Redirect/Redirect/Redirect?url=http://www.malware.com
From the victim's perspective, this link is pointing towards localhost:3988 and not a malicious site.

The safest way to make this non-existent is using making a redirect page for each redirect you want to do. Instead of requesting the redirecting-URL as a parameter from the URL, you will put the redirecting URL directly into the Redirect function like this:

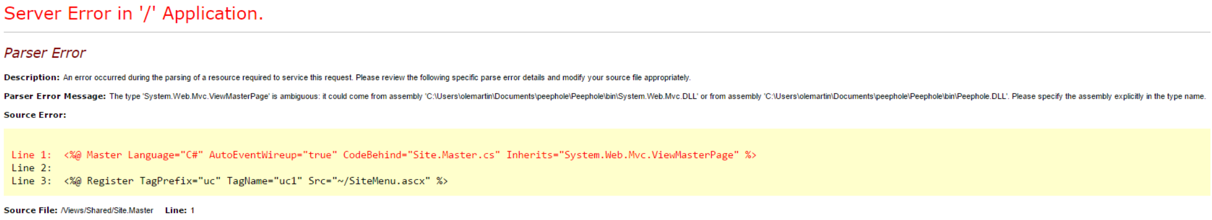
```
Response.Redirect("http://www.safeURL.com");
```

By doing it this way, the attacker cannot change the URL to his advantage.

KNOWN BUGS

Site.Master changes

When changing the Site.Master (which contains the structure for the web application), the designer changes the namespace automatically. Why this happens we do not know (guessing we have about 20-30 hours of research, testing and failing on this, but we have not found a way to fix it).
Below you can see what needs to be changed every time there is a saved change in Site.Master. If we are not doing this, the application would not run and you will get this error:



Server Error in '/' Application.

Parser Error

Description: An error occurred during the parsing of a resource required to service this request. Please review the following specific parse error details and modify your source file appropriately.

Parser Error Message: The type 'System.Web.Mvc.ViewMasterPage' is ambiguous: it could come from assembly 'C:\Users\olemartin\Documents\peephole\Peephole\bin\System.Web.Mvc.DLL' or from assembly 'C:\Users\olemartin\Documents\peephole\Peephole\bin\Peephole.DLL'. Please specify the assembly explicitly in the type name.

Source Error:

```
Line 1:  <%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site.Master.cs" Inherits="System.Web.Mvc.ViewMasterPage" %>
Line 2:
Line 3:  <%@ Register TagPrefix="uc" TagName="uc1" Src="~/SiteMenu.ascx" %>
```

Source File: /Views/Shared/Site.Master **Line:** 1

Screenshot 18: Error message when changing the Site.Master

How to fix this:

1. This happens when changing the Site.Master file located in ~/Views/Shared/Site.Master. When changing this file, the Site.Master.designer.cs updates automatically. Originally it should be
   `namespace Peephole {`
   But changes to
   `namespace System.Web.Mvc {`
   when we edit the Site.Master.Master file.
2. To fix this, go to the designer file and change the System.Web.Mvc to Peephole and it should work.

8.8 Attachment 8: Database tables

```
CREATE TABLE [dbo].[BankAccount] (
   [AccountNumber] BIGINT       IDENTITY (3994045000, 1) NOT NULL,
   [AccountType]   INT          NOT NULL,
   [UserID]        INT          NOT NULL,
   [AccountName]   NCHAR (30)      NULL,
   [Balance]       DECIMAL (18, 2) NOT NULL,
   PRIMARY KEY CLUSTERED ([AccountNumber] ASC),
   CONSTRAINT [FK_BankAccount_AccType] FOREIGN KEY ([AccountType]) REFERENCES
[dbo].[BankAccountType] ([Id]),
   CONSTRAINT [FK_BankAccount_User] FOREIGN KEY ([UserID]) REFERENCES
[dbo].[Users] ([Id])
);


CREATE TABLE [dbo].[BankAccountType] (
   [Id]   INT       IDENTITY (1, 1) NOT NULL,
   [Type] NCHAR (10) NULL,
   PRIMARY KEY CLUSTERED ([Id] ASC)
);


CREATE TABLE [dbo].[Feedback] (
   [Id]      INT           IDENTITY (1, 1) NOT NULL,
   [Name]    NVARCHAR (MAX) NULL,
   [Subject] NVARCHAR (MAX) NULL,
   [Content] NVARCHAR (MAX) NULL
);


CREATE TABLE [dbo].[Files] (
   [fileID]       INT          IDENTITY (1, 1) NOT NULL,
   [filenameName] VARCHAR (100) NOT NULL,
   [filePath]     VARCHAR (100) NOT NULL,
   [Createdby]    VARCHAR (100) NOT NULL,
   [CreatedDt]    DATETIME      NOT NULL,
   [Updatedby]    VARCHAR (100) NULL,
   [UpdatedDt]    DATETIME      NULL,
   [Active]       BIT           NOT NULL,
   PRIMARY KEY CLUSTERED ([fileID] ASC)
);


CREATE TABLE [dbo].[MainMenu] (
   [Id]          INT           IDENTITY (1, 1) NOT NULL,
   [ShowInMenu] BIT           DEFAULT ((1)) NULL,
   [AreaName]   VARCHAR (100) NULL,
   [LinkText]   VARCHAR (100) NOT NULL,
   [ParentId]   INT           NULL,
   [URL]        VARCHAR (100)  NULL,
   [Priority]   INT           NULL,
   PRIMARY KEY CLUSTERED ([Id] ASC)
);


CREATE TABLE [dbo].[Transaction] (
   [Id]          BIGINT         IDENTITY (1, 1) NOT NULL,
   [Amount]      DECIMAL (18, 2) NOT NULL,
   [FromAccount] BIGINT         NOT NULL,
   [ToAccount]   BIGINT         NOT NULL,
   [Timestamp]   DATETIME2 (7)   NOT NULL,
   PRIMARY KEY CLUSTERED ([Id] ASC),
   CONSTRAINT [FK_Transaction_From] FOREIGN KEY ([FromAccount]) REFERENCES
[dbo].[BankAccount] ([AccountNumber]),
   CONSTRAINT [FK_Transaction_To] FOREIGN KEY ([ToAccount]) REFERENCES
[dbo].[BankAccount] ([AccountNumber])
);


CREATE TABLE [dbo].[Users] (
   [Id]        INT           IDENTITY (1, 1) NOT NULL,
   [Firstname] NVARCHAR (50)  NOT NULL,
   [Lastname]  NVARCHAR (50)  NOT NULL,
   [Password]  NVARCHAR (32) NOT NULL,
   [Email]     NVARCHAR (50)  NOT NULL,
   PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

8.9 Attachment 9: Feedback

Feedback 1

Nedlastingen pågår :-)
Videoen ser veldig bra ut. Applikasjonen er jo kjempesårbar, hehe.

Jeg har ikke hatt tid til å se nærmere på applikasjonen i praksis enda. Det har vært en hektisk uke med arbeid både på dag og kveldstid.
Gleder meg til å teste VMen og gi dere en tilbakemelding.

Gratulerer med arbeidet så langt.

Hei alle sammen,

Jeg har sett nærmere på applikasjonen, og den ser bra ut. Gratulerer med arbeidet så langt :-)
Jeg har utnyttet noen sårbarheter, og de fungerte bra.

Sender her en tilbakemelding på noen punkter som kan forbedres.
Håper at dette kan hjelpe:

... Things we needed to fix ...

Mail from Juan:

"Teaching and learning web security is a difficult matter, since applications deployed in the real world cannot be used as training platforms.

Vulnerable web applications designed for web security training are great resources that help closing this gap.
However, these kinds of applications tend to be predictable, very unrealistic, and they have important limitations.

For example, vulnerable web applications available these days are very focused on open source technologies, such as PHP and MySQL.
This limitation gives teachers and students a very narrow set of possibilities, with a short-sighted view of web security and platform-specific vulnerabilities.

Peephole is an excellent project that takes web security training to a next level, since it tackles many of the unsolved challenges seen until now.
This application has been built with current .NET technology, which offers an environment not

common for web security training, but which has a very important market share in web application development.

Peephole also provides an attractive banking environment, which allows students to explore and test their web security skills, without expecting predictable results.

Finally, it is important to mention that Peephole offers a very exciting development platform, which may allow other students to go deeper in web security.
This could be shaped in the development of new vulnerable modules, extending existent functionality, or even learning how to implement recommended solutions for fixing vulnerabilities from a programming perspective.

Med vennlig hilsen / Kind regards
Juan J. Güelfo

CEO & Lead IT Security Consultant
Encripto AS - Information Security.

Feedback from Master Student

Have you heard of any of these application before?

- *WebGoat*

What is you first impression of Peephole?
When I first see the Peephole I feel the UI is very comfortable and many modules.

Have you found any vulnerabilities?
I found some vulnerabilities as follow:
1.reflected XSS (vulnerability parameter is id) url:
http://localhost:3988/Areas/Login/Welcome?id=%3Cembed%20src=..%20%3E
2.Stored XSS (vulnerability parameters:Name 、Subject、Content url:
https://localhost:44301/Feedback/Comment/Comments
3.SQL Injection (POST Injection / vulnerability parameter is ctl00$MainContent$input) dump database url:https://localhost:44301/Areas/Employee/Search
4.Unrestricted File Upload （get webshell） Upload path: ../../Userfiles/xxx.asp url ：
https://localhost:44301/Areas/LoanApplication/Application
5.Modify any user password （traversal UserID） URL ：
https://localhost:44301/CustomerInfo/Default/Edit/12
6 View any Bankaccount details (traversal Bankaccount Number) url ：
https://localhost:44301/TransferFunds/BankAccounts/Details/3994055030

7.Modify any Bankaccount AccountType & AccountName (traversal Bankaccount Number)
url:https://localhost:44301/TransferFunds/BankAccounts/Edit/3994055029
8.CSRF url: https://localhost:44301/Feedback/Comment/Comments .....etc.

Do you miss any vulnerabilities?
Yes

What do you think about the design? (From 1 til 10)
8

If you could change anything, what would it be? Why?
I think I could change the the module of loan application ,do some restrict like file format or add content checking. because do that increasing the diffcult as well as adds to the Knowledge points and diversity

Any additional feedback?
Error page leakage information (throw exception)

Date: 2016-05-18

8.10 Attachment 10: Cooperation agreement

Denne avtalen gjelder samarbeid mellom følgende organisasjoner/parter:

1. Jens Vingen
2. Per-Olav Eikrem
3. Ole-Martin Bratteberg

Avtalen gjelder samarbeid om planlegging, gjennomføring og rapportering for følgende tiltak:

Slik som vi har forstått skal vi rullere på «rollene» som vi har i løpet av prosjektet, slik at alle sitter igjen med samme kunnskap.
Alle på gruppen har et ansvar for å ta kontakt med kontaktlærer omtrent hver 14 dag.

Arbeidstider vil være fra 08.00 til 16.00 hver dag (vi har og noen timer på tirsdag og onsdag som alle på gruppen deltar i). Helger vil bli brukt ved behov, men hovedsakelig vil vi bruke ca. 8 timer hver hverdag til jobbing med bachelor.

Mål for samarbeidet:

Målet for samarbeidet er å utarbeide en bacheloroppgave på best og mest mulig effektiv måte for alle involverte.
Vi skal kunne vise til samarbeidsavtale ved uenigheter, for i dette prosjektet skal alle jobbe tilsvarende lik arbeidsmengde/timer.

Ansvarsfordeling – planlegging og gjennomføring av tiltaket:

Vi kommer til å ha flere roller i prosjektet, blant annet prosjektleder, Scrum-master, Git-master, rapporterings-sjef, rapport-sjef, m.m. Her kommer vi nok til å rullere litt hvem som har ansvar, slik at alle har mulighet for å lære det samme.
Vi må være i kontakt med kontaktlærer minst hver 14 dag, og dette er også noe som en eller flere av oss tar ansvar for å sette opp et avtalt møte.

Alle skal møte hver dag til avtalt tid og være på skolen avsluttende «skoledag».
Ved sykdom eller andre akutte saker skal vedkommende si ifra så snart som mulig.
Ved reise/planlagt bortetid skal vedkommende si ifra tidligst mulig.

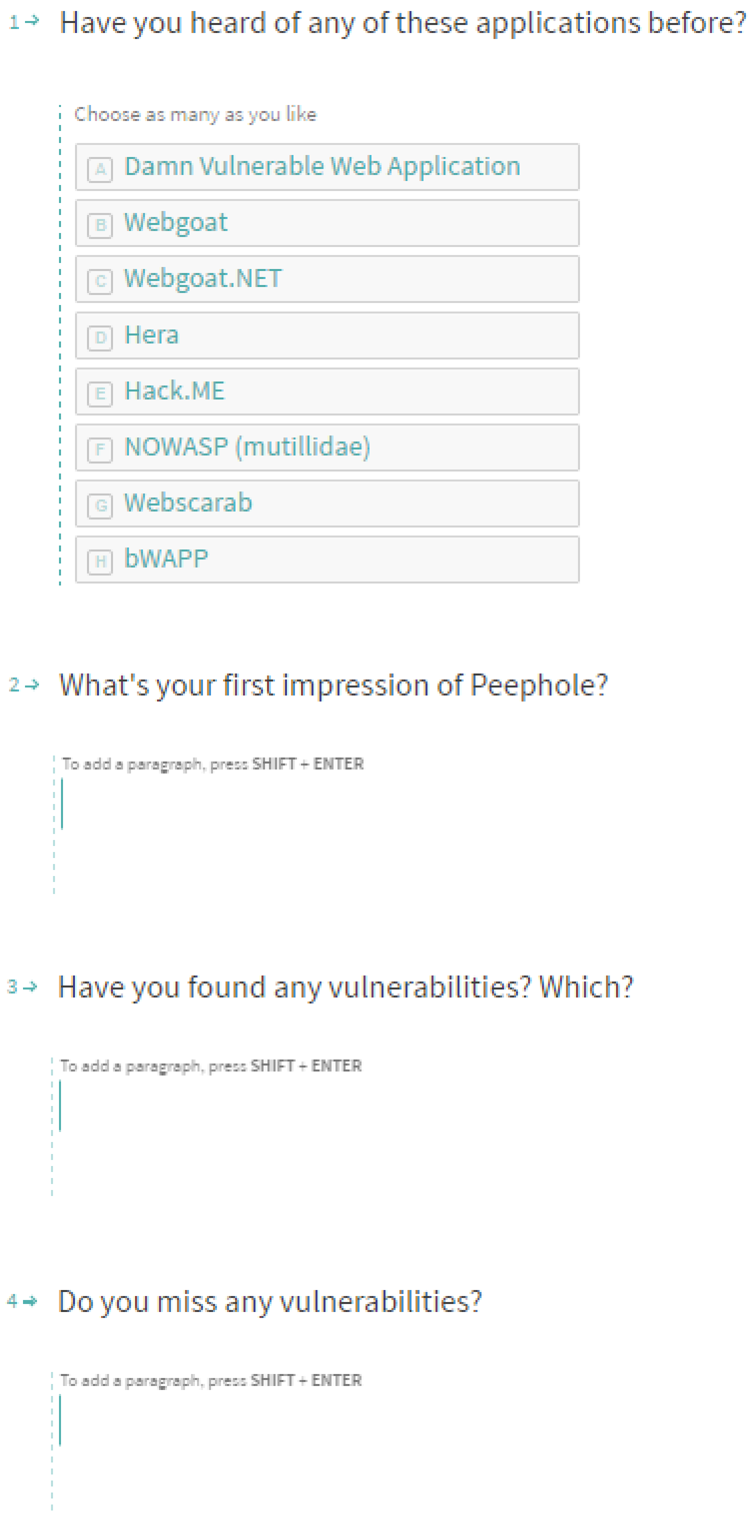
Samarbeidsavtalens varighet:

Fra januar 2016 til slutt av bachelor prosjekt mai/juni 2016

Informasjonsplikt

Alle parter forplikter seg til å informere de andre om alle forhold som kan ha innvirkning på gjennomføringen av tiltaket.

8.11 Attachment 11: Typeform



1 → Have you heard of any of these applications before?

Choose as many as you like

- A Damn Vulnerable Web Application
- B Webgoat
- C Webgoat.NET
- D Hera
- E Hack.ME
- F NOWASP (mutillidae)
- G Webscarab
- H bWAPP

2 → What's your first impression of Peephole?

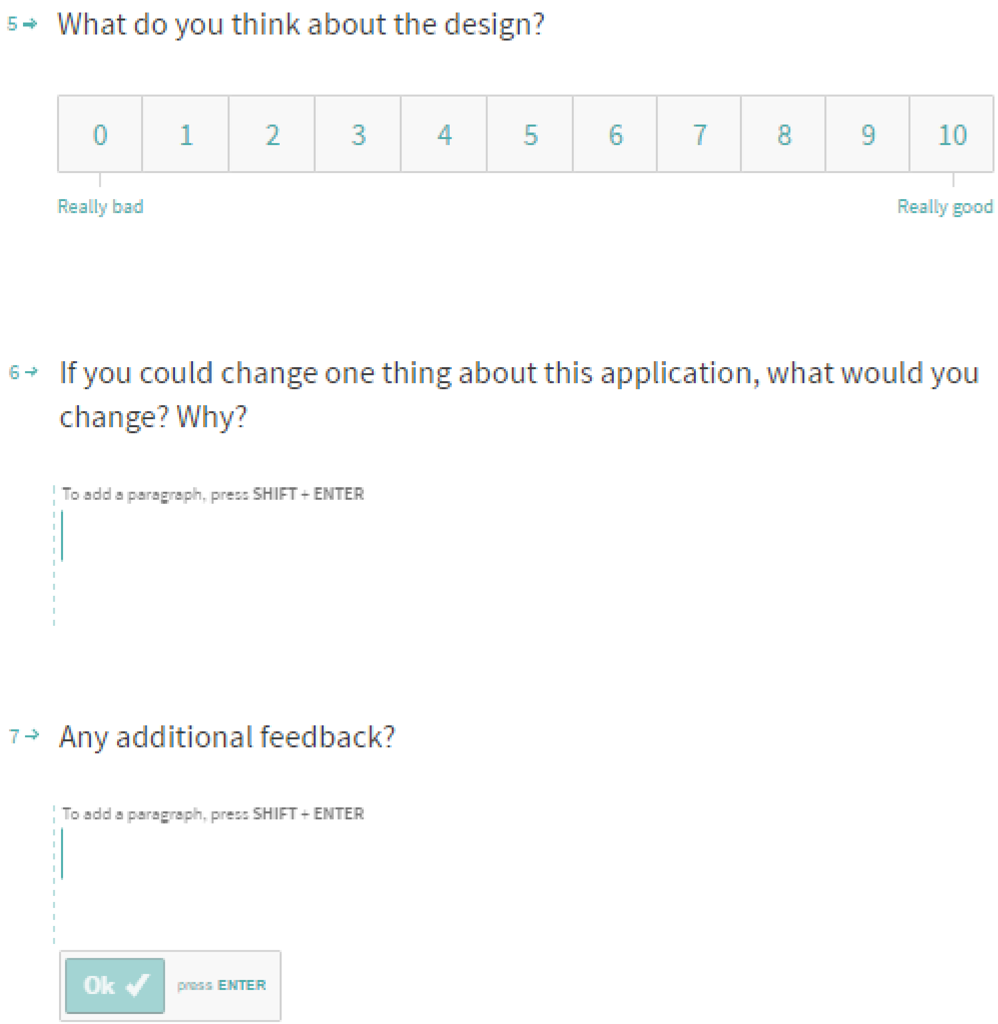
To add a paragraph, press SHIFT + ENTER

3 → Have you found any vulnerabilities? Which?

To add a paragraph, press SHIFT + ENTER

4 → Do you miss any vulnerabilities?

To add a paragraph, press SHIFT + ENTER



5→ What do you think about the design?

0 1 2 3 4 5 6 7 8 9 10

Really bad — Really good

6→ If you could change one thing about this application, what would you change? Why?

To add a paragraph, press SHIFT + ENTER

7→ Any additional feedback?

To add a paragraph, press SHIFT + ENTER