Creating secure software

Runar Moen

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2013

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Creating secure software

Runar Moen

Abstract

Security in applications has for more than 25 year been one of the biggest challenges in software development, and as a consequence earlier response to improve overall quality and security of applications has failed. There have been some improvements, but it seems that old errors are repeated every time a new platform is introduced.

Application security is sometimes confused with software related to security, but it is about having security in all types of software. Today applications are often produced in the fastest and cheapest way, with no or little focus on security. The goal with this thesis is to collect and arrange information development projects can use to improve the security in the applications they create. This will include technical information about secure coding, and also for assuring secure development process. Frameworks with world class target groups such as best practice from sources like Microsoft, specially Microsoft Security Development Lifecycle, OWASP, and others will be examined, compared and used for recommendations.

The main audience is small to medium teams, with 1 to 20 developers where it is normal for one person to have several roles, and where all will be responsible for the quality of the code. However, the techniques and processes will also be possible to scale up to larger teams.

The resulting material includes training material, blog articles and this thesis with background information.

Summary

Like other parts of building software, there is no silver bullet[1] to magically fix all security problems. The only way to get good security is by hard work and by keeping focus on quality and security. Tools and frameworks can only help to some extent. Then tools have to be supported by developers writing good code, business stakeholders and analysts knowing how to define business rules for security, and testers that are creative when testing for security. Software keeps getting more and more complex, and complexity makes all these parts harder, so we have to keep improving in order to have a chance to get ahead of the bad guys. Different software has different requirements for security, but all software has a responsibility for protecting the users.

In this thesis we have looked at ways for improving the security in web-applications. We have found sources about the need for web-application security, and worked on how to spread awareness and knowledge about the subject. A training program was created and used to train a development team, and a blog was created as a way for spreading knowledge and awareness, and to receive feedback on the content. The produced material can be used as a starting point for teaching development teams about security and get them started with securing their products.

Contents

1 Introduction

In the beginning computer security was more about securing the large mainframes so real life bugs and rodents was kept away from them[2]. When we started to connect computers together in networks pranksters[1] and researchers started to create, what later was named, viruses and worms. As the timeline on Wikipedia[3] shows the number of new malware[2] has steadily increased over the years. After the part of Internet that is known as World Wide Web, now popular called only Internet[4] was created and become popular, web pages started to be the target for attacks.

The functionality and complexity of Internet applications has continued to increase. We are now living in an exciting era with a rapid change in how we communicate and trade. This possibility was enabled by Internet and by all the different devices we connect to it. Fast development creates an increasing demand for software developers, also with the help of some new tools it is possible to create applications and web-sites without much training and specific knowledge. A side effect of this it is risk that the proceeding trend with improving security in software might stop and even reverse.

The attacker's toolset is also growing, and it keeps getting easier to use scripts and applications to attack web-sites and applications. While developers have to know how security work, attackers like normal users, do not need to have any special knowledge rather than knowing how to start a program and typing an address to launch an attack. This is why this level of attackers has been given the name "Script Kiddies". Attackers do also have, what is called, unlimited time, while developers most often are on a budget giving limited time. So developers will need any help they can get with securing their work. The material produced as a part if this thesis can be used to help developing teams to make conscious decisions for the security level in the different parts of security, based on their needs. Instead of being limited by ignorance and not knowing what to secure and how to implement the security requirements.

Security is divided into the different parts:

- Confidentiality
- Integrity
- Availability

[1] A person found of playing pranks, is more known in relationship to phreaking and exploiting phone systems
[2] Software that is intended to damage or disable computers and computer system

- Authenticity
- None-repudiation

Creating secure software has one major goal to achieve - to make sure that the software in a good way integrates all this parts.

With this thesis we will try to close the gap between those who know and care about software security, and those that do not know enough to care about security. It is written as part of technology track, but could also have been from the management track in the master's degree. It fits under technology since most of it is about how technology leaders and technical peoples with security interests can help train and motivate other technical peoples about security. This is tried to be done without downplaying the importance of having security in all levels of the development process, and that security is owned by top management. The intention, and methods, used here is to have a bottom up approach where technical peoples will be trained and made aware and then can get help with convincing the upper management that security is important and should have focus. Having it the other way around, where it would have been about how management can get the technical peoples to care about security will probably have fitted better in the management track.

2 Topics

First we can define what a web-application is. In our case we can consider any web site that interacts with users as web-applications. Web-applications can be anything from online banking, to web-sites that allows users to log in and post a comment on a waffle recipe. This does also cover online web-site generators and blogging software. It can even include smart-phone apps. The main difference between desktop applications and web-applications is that the data is accessible from anywhere in a web-application. This is a large convenience for the users, since they do not have to care about installing any software, taking backup and moving the data around when changing computers. The largest drawback is that again the data is accessible from anywhere, meaning that anyone anywhere can try to get access to it. Vulnerabilities related to this are mostly harmless on desktop applications since you do not harm others if you manage to break into your local database or try to use the application to hack your own computer. On the other hand with web-applications where several registered and not registered users interact with the same database and application it can be possible to attack others if there are vulnerabilities that can be exploited.

This thesis covers topics in training and awareness of computer security, especially related to software and applications available on Internet. A significant part of the material produced is introduction to create awareness and interest among software creators. There have been some proposals for regulations to force companies to produce more secure software[5]. However we think creating awareness and curiosity among developers will help more than creating regulations. Regulations will set a minimum level and it is a chance that it will be used as the yardstick to measure against. We think showing consequences of attacks and ways how to attack is a good way to trigger interest, and that giving this information an introduction course will give developing teams a good start in the work on securing software.

The training is mostly at an introduction level covering the most common problems in modern web software. OWASP top 10 vulnerabilities list have been the main influence for finding which parts to include. The list is created using the OWASP Risk Rating Methodology[6], more on risk rating in the section about risk and cost. Using this methodology did result in 4 risk factors used to calculate the order of the Top 10, 3 Likelihood Factors, and 1 Impact Factor. These factors are:

- Likelihood of an application having the vulnerability (Prevalence)
- Likelihood of an attacker discovering that vulnerability (Detectability)
- Likelihood of an attacker successfully exploiting that vulnerability (Exploitability)

- Typical technical impact if that vulnerability is successfully exploited (Impact)

Text quoted from the OWASP page for project methodology[6]. The OWASP Top 10 Project collects prevalence data and have ranked them using ranking from the risk rating to decide the candidates to include and the ordering of them, and after a review period the final list is published. The goal with the list is to create awareness and inform on where to start to focus on improving the security in web-applications. This makes their list an easy choice for using it as basis for finding topics to include in the background information for this thesis. The OWASP Top 10 list is not the complete list of all possible vulnerabilities, but it is a good place to start.

The list of resources and additional material is used as an introduction together with suggestions on how to create a short training program and how knowledge of the system and a risk assessment should be used to find where to start the training. General web application security and connecting security and quality is also a binding factor in the published material.

Cloud computing is often mentioned in current literature, also in security. We have not included any special parts about securing cloud applications, since cloud is more about operation and how the applications is hosted and do not make large differences for how it is developed. Web software is using the server client model both when hosted by the company using the application and when it is run from servers hosted by others. The only difference can be in case with applications that are only available on a local intranet, but they will still have to be secured since many attacks are done by insiders.

2.1 Problem Description

Before the Internet attackers could only hack software they had access to, now when everything is getting connected it is possible for attackers to try breaking into systems located anywhere. Internet does not have borders or opening hours. Putting software online makes it possible for anyone to attack it. Many are not aware of this, and do not see the implications of a successful break-in. The amount of new applications on Internet keeps rising, and to be able to compete new products are rushed to the market every day. With this demand for getting the products out to the market it is easy to cut corners and take shortcuts; also the security training and knowledge among the developers might vary. Several research papers indicate that the software industry still have a long way to go before we achieve the quality and security levels our users deserve. Some try to link quality and security[7], while others conclude that our industry still shows signs of immaturity[8].

The economy connected to web-applications is also increasing, and an increase in economy can also attract more criminals. Attacking software on Internet is also relatively safe compared to other criminal activities. The criminals can be located in countries that have different rules and no extradition treaty with the country where the attacked software is located. We have to create more secure software to be able to make the cost and the risk for the criminals to be higher than what they are gaining from exploiting software. In many cases exploiting the software is only

a step on the way of attacking the users, using drive-by techniques where vulnerable sites are exploited to include payloads that try to exploit the browser or browser add-ons the sites visitors have.

There are several reasons why applications and web sites are attacked. Some of the reasons are:

- Hacktivism, where web-sites are hacked for political reasons. Hacktivism is from the words hack and activism. A hacktivist is much like a hacker, to main difference is why he hacks. This is the digital version of putting up flyers, arranging demonstrations and tagging a wall with political statements.
- Industry or military espionage, this is the digital version of traditional espionage that has been conducted as long as there have been industry and military. This can also be seen as a part of what is known as Information Warfare.
- Using the hacked sites as springboard to attack users. This might be the most serious threat to most web-applications. It will be no visual indication of the hack, at least not before browsers put up a warning telling your site include harmful code and recommend users to not visit it.
- Harvest usernames and passwords. Many users use the same username and password on several sites, so attackers can attack less secure sites to find usernames and passwords so they can try them on more secure sites. The passwords can also be used to update dictionaries used for dictionary attacks for login and finding secured passwords.
- Showing off, and attacking for fun. Also known as phreaking, which was more used to exploiting phone systems to get free calls, or for the lulz meaning for the fun where lulz is plural for LOL[1]

Web-applications have to withstand all this kind of attacks, every day and every night, from all over the world. Developers do have a limited time and budget, while attackers can have an abundance of time. It do not cost much to have a computer, or a network of computers, running scripts against a site for hours or days, and how much time an attacker will use depends on the price he gets when succeeding. It is enough for an attacker to find one serious vulnerability to get access, while developers have to find all to be able to remove them. So developers need all the help they can get in the work on creating more secure applications.

Traditional security measures do also not help when coming to web-application security. Firewalls blocks traffic that is not allowed, but it assumes that the applications that are allowed access behaves and do what it should do. However the attacks are against the application to make it do other things than what it is supposed to do. Antivirus both client side and server side looks for known signatures to block harmful software, so this do also assume that the web-application do what it should do. The antivirus software will then only be useful defending against attackers

[1] Laughing Out Loud

trying to install harmful software on the compromised machines. We can encrypt data in rest on the server and in transit, but it has to be decrypted for the users to do anything with it. Network vulnerability scanners do also not help much; it can detect server misconfiguration but not most vulnerabilities in the software.

2.2 Justification, motivations and benefits

Even though there are a lot of information on how to create secure software we keep getting new software with questionable security. Collecting and organizing some of this information in an easy to find and understand way can help getting developers started in creating more secure software. The main motivation for writing this is to secure the users from compromised web sites. Attacks against web-sites do in many cases not only harm the company behind the site, they can also compromise other sites hosted on the same server and it can, and in many cases will, be used to launch attacks against visiting users.

According to a report from the web-security company WhiteHat security "86% of all tested web-sites had at least one serious vulnerability[2], and most of the time far more than one -56 to be precise."[9] They are also reporting a decline in the number of vulnerabilities, but show that we still have a long way to go. Another finding they have made is that of the companies they have surveyed 57% provide some amount of software security training for their programmers, and that these organizations experienced 40% fewer vulnerabilities. They do also report that 23% of organizations answered that they had experienced a data or system breach as a result of an application layer vulnerability. This shows that putting focus on security does help, it does not directly show if it is the actual training that helps or if it is that companies providing security training also sets higher standards for security. It is likely that it is a combination of the two, so doing security training should be a part of a larger security strategy.

The knowledge and training program created as a part of this thesis will also be used in an in-house training for several of the teams in the company where I have my day job. But since both the program and how it is made is available on Internet it will be possible for others to use the same program, and make adjustments to it to make it fit more to their needs. This can then help making our user information safer when companies we trust our data with create more secure software.

2.3 Planned contributions

A blog was created to increase awareness and give basic knowledge about creating secure web-applications[10]. The blog has two main themes, one is to get developers and other related to software production aware and interested in securing their work. The other main part is to give

[2]Serious vulnerabilities are defined as those in which an attacker could take control over all, or some part, of the website, compromise user accounts on the system, access sensitive data, violate compliance requirements, and possible make headline news.

an introduction to secure internet enabled software, including a training program. See more on the program in another section. The blog has not been heavily marketed, some posts were shared on Facebook[3], some were advertised via Twitter[4] and some were published on Hacker News[5]. Totally have there been around 1500 page-views by mid-October, not including own visits. The statistics for visitors do include search-crawlers[6] and probably a number of spambots[7]. The posts with most views are those that were published on Hacker News, even when none of them did make it to the main page. It seems like the technical stories were more shared on Google +1, and did get more likes on Facebook. But the dataset is too small for getting a conclusive result. The technical posts do also seem to get more late views by visitors finding it from links in other posts and through a search engine. The continued work after the thesis is finished will show if this visitor and sharing distribution is continued. The page-views are mainly distributed between 10 countries, creating a list that on October 13th is:

United States	558
Norway	368
China	54
Sweden	47
India	42
Belarus	40
Russia	39
Germany	33
Israel	27
United Kingdom	24

Table 1: Visitors per country

The relative high number from Norway is because the author is Norwegian, the high numbers from Sweden and Belarus can be attributed to the author having colleagues in both countries. The main search engines are located in United States and are contributing to the high number of visitors from USA.

The training program will in different versions be used to train developing teams in the company where the author works. As the part of the training updated secure coding guidelines will be created for the different teams. It will be done to give the participants more ownership of the guidelines and to help with creating awareness in the projects.

This thesis is not about researching new knowledge or creating something unique; it is about finding ways for spreading awareness and to get developers started in securing the software they create.

[3] facebook.com
[4] twitter.com
[5] news.ycombinator.com
[6] A search-crawler, spider or bot is a search engine's automated program that crawls web sites by following href links to index the pages for usage by the search engine
[7] A spambot is an automated computer program designed to assist in the sending of spam, they are also used for visiting blogs hoping the blog owner will check the referrer site

3 Secure Software

It helps knowing what secure software is in order to be able to create secure software. Secure software is a part of information security, even when it is not security software. The software will still have to follow the main principles for information security. To get a holistic view on the security it will have to be included in all parts of the process, it is not enough to focus on it during development and maintenance. The architecture and deployment are also very important for security. The architecture and deployment of the software most be treated so it is possible to follow the demands for the required level on the main security principles from the requirements and specifications. Different systems will have different demands for security in the different principles, however it is important to consider the different part and give an assessment on them. Some of this like confidentiality and integrity is more obvious than the others, since they should be enforced in all kind of applications, the others might be more relevant for software that requires a higher level of security.

3.1 Confidentiality

The information transferred between client and server and the information stored on the server should not be available for others than the intended recipients. For instance the evaluation a teacher sets on a pupil should only be available to the users that it have been published to, like the student, other teachers or staff connected to the student and guardians if the student is not of legal age. Other students should not be able have access to this information it in the system. Other systems like medical records will have other demands, there it could be requirements saying that IT staff and others with access to servers also should not have access to the information, and that it should be logged by whom and when the information have been accessed. The requirements for confidentiality will then both cover authentication and authorization. Where authentication is verifying who the user is, this is normally done through user login with username and password, maybe also including a two factor component like a code from a SMS message. Authorization is giving the right access to the right user, after the user is authenticated, saying which content the user is authorized to get access to. There are several ways for ensuring confidentiality, and the measures should be taken based on the requirements for the specific application. In some cases using HTTPS, even with a self-signed certificate users will have to accept in their browsers might be enough. In other cases it might be required to use a VPN[1] to make a secured connection to get access, or even use a computer that is not connected to a network.

[1] Virtual Private Network

3.2 Integrity

Data stored should not be changed by anyone unauthorized or undetected. A pupil should not be able to change the teachers evaluation, neither in the system or while the assessment is transferred from the client to the server. It should also not be possible for anyone that is not authorized to change data through the application or directly on the servers. Users should be able to trust the data they see, they should trust that the content is right and that it is from the person they think it is from. A student should be sure that the grades shown on the page belongs to the student and is set by the teacher. More important places, or maybe not for a student, can be that the pharmacy must be sure that the prescription is to the right patient. This do in some parts overlap with authenticity, the difference is that integrity is more about the connection between entities, not who it is from. Integrity can be hold in several ways, spanning from using foreign key constraints in the database and using HTTPS for sending the data, through having backups and do integrity checks on the code using checksums, to store the data on a server in a faraday cage protecting it from an EMP[2]

3.3 Availability

Availability does mostly affect user experience, but it can also be more critical. It can for instance be that the intake system for schools is down because of DDoS[3] attack making it unavailable for applicants to apply, or it can be that the testing software is unavailable when a test is scheduled. This might seem like a problem for IT operations, and not part of developing the software, but how it is developed can influence how easy it is to protect. One example is that functions to start heavy jobs or synchronizations are not available through an open web-site. It can also be to introduce fault-tolerance especially when integrated with 3rd party systems. Availability can also be ensured by allowing load balancing and supporting redundancy, and if needed alternative datacenters. So if one server or center have problems, or is under a DDoS attack another server or center can take its place.

3.4 Authenticity

Some data require that the validity is guaranteed to be from the claimed sender, so it is a proof of identity. This can for instance be when a doctor sends an electronic prescription to the pharmacy. The pharmacy need to know that the prescription is from the doctor. Authentic can be verified through authentication, so the level of ensuring the authenticity correlates with the authentication security. Most systems require a username and password, this is the normal traditional way, and the security is only as strong as the password and how secure the password is stored. Many users selects simple passwords that are easy to remember, and easy to guess. It is possible to get higher level of security by introducing a second factor, making it a two-factor or multi factor

[2]Electromagnetic Pulse
[3]Distributed Denial of Service

authentication system, by adding a physical object like a code generator, key card or a mobile phone to get a one-time code to add with the password. Higher level of authenticity assurance can also be achieved by setting up a trusted circle using PKI[4], which will be a smaller version of how TLS/SSL[5] certificates is used in web-browsers, where the browsers accepts certificates signed by any root certificate which they already have in the list of trusted certificates.

3.5 Non-repudiation

In some cases it is important that users cannot deny having sent data. The level of prof for who did send information is linked to authenticity and authentication. It is also possible to use a digital signature as prof for who did create the data. This is handled by more secure software, so out of the scope for our work on securing normal web-applications.

[4]Public Key Infrastructure
[5]Transport Layer Security/Secure Sockets Layer

4 Related Work

Several books, articles, courses and online material cover the same topics. This thesis is not about finding new ways for secure development, it is created to help spread the knowledge and to be a starting point for learning about secure development. The range of available information on and related work on software security spans from the simplest web pages on specific topics, like SQL injection, and all the way up to full educations, like Masters of Information Security degrees. The problem is not to find information about the topic, the problem is to navigate the vast size of the information and find what is relevant and finding where to start. It is easy to spend so much time on reading how to write secure code and how to secure applications that it will not be time left to implement it.

Some of the most relevant work and influences are:

- OWASP[11], the Open Web Application Security Project. It is a worldwide not-for-profit charitable organization focused on improving the security of software[11]. They have an extensive list of resources, libraries and information on all parts of web application security. They do also have several books, which can be downloaded for free and arrange several application security conferences in different locations. The downside with their web site is that it is so much information that it is easy to keep reading instead of start implementing.
- Mozilla secure coding guidelines[12], is guidelines for security in web-applications and web-services developed my Mozilla.
- MS SDLC, Microsoft Security Development Lifecycle[13] is based on Microsoft's focus on secure development practices which they have incorporated in their development and provide resources for how others can do the same.
- SANS institute and their resources[14]. SANS have training programs and online resources to spread awareness and knowledge about computer security.

Books and articles that were directly used are cited through this thesis, but many others have been used for inspiration and help with the different parts. One problem with application security might be the vast size of information that could be found when looking, especially compared with how little that is in mainstream media. It is speculated that the news effect of normal security issues is close to nothing since it is so normal, and it only is published in mainstream media when it affects large enterprises. This can lead to some taking higher risks since they estimate the chance to come in the media so low that a security breach will not harm the bottom line

much. So a part of the remedy can be to help making the information more accessible for all, not only for the security community. In other words, the size of available related work should not hinder creation of more work and trying to get more attraction to the subject.

5 Choice of Methods

The main aim with this work is to help improve the overall security among web-sites and -applications. The chosen methods for this are to use information and examples to create more awareness and trigger curiosity to learn more. We think a large impact can be gained by making developing teams more aware and interested in security. Also understanding why security is important is a better incentive for focusing on security rather than laws and regulations.

We did choose to include four closely related parts in this thesis, where all are ways for spreading knowledge and awareness about creating secure web-applications. The different parts were selected to be included to cover more of the security process and to attack the problem from different angles. All parts is connected and do partly overlap each other. First the projects have to see the need for security and be aware on which parts that is most important to start focusing on, and then the participants have to learn what they need to do to achieve the desired security. And last they need to implement the new security knowledge in their work. Writing about a topic or teaching others is a good way of getting more knowledge about the topic. After improving the security it is time to go back and find the next focus point and continue the process.

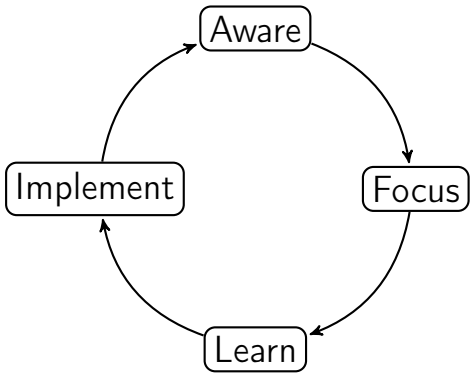

Figure 1: Security as a process

We did apply qualitative exploratory research methodology to create content for the different parts, and have included some quantitative methods for analyzing the result of applying the information. We started with analyzing the current state of the art and related work to find a starting point on which parts to focus on and start with. We did find which parts that was most important to spread awareness on, and used that to create a training program and write blog posts about awareness and introduction to web application security. The feedback on both the

blog and training program has then been used for improvements and inspiration in the continued work.

The research in this thesis is mostly a qualitative exploratory research. There are no definitive answers and no new knowledge. Writing secure code is not clearly definable, since there is no definitive answer to what secure software is. There are different levels of security; modern software is too complex for anyone to find all security problems and fix them. The most important part might be to make conscious decisions on the required level of security for each development project, and even for the different components in a project. Exploratory research in this case can also be described as secondary research where available literature is reviewed and used as background. Qualitative approaches like informal discussions with members of development teams and feedback on training and published blog-posts are also used. This approach was selected since it will not be precise to set a hypothesis related to proofing that some or most software on Internet is insecure. The available dataset is so large that it will be close to impossible to get a reliable dataset representative for all available applications. It will also not help much in finding what is the problems applications have in common and why they have them. It was better to use existing sources like the OWASP top 10 list[15] and Mozilla secure coding guidelines[12] to find common patterns in vulnerabilities and how to avoid them, since they already have used a lot of time on finding common vulnerabilities on a large dataset. Their selection of data-points might not be large enough to qualify as empirical research, but it is a good place to start exploratory research from.

6 Methodology

Research methodology is traditionally separated into two main parts, qualitative and quantitative research. Qualitative research is about understanding behavior and reasons, questions to be asked and gathering data. But the result is not measured in a quantifiably way, it is more concerned with why and how. It is the quality, not the quantity of samples that is important. On the other hand in quantitative research the researcher tries to measure the result in an empirical way having statistics and calculations to back up the result. Quantitative research relies on high quantity of samples to create statistics and to show trends. Exploratory research can be used to formulate a more precise problem, for instance by reviewing available literature or having informal discussions with stakeholders. This thesis is divided into four parts covering different areas and approaches related to increasing the security in developed applications.

6.1 Need for web application security

This part is mostly described in the chapter Justification, motivations and benefits. Here we did use qualitative exploratory research to find background information about common vulnerabilities and their frequency and impact to find which parts to focus on. Statistics from several sources was used to show how much of a need there is for more secure web applications, to get some numbers backing up what can be seen in media. We have also emphasized the need for conducting risk assessment to find the requirements for security in different web-applications to find the important points to focus the security work on while developing. Results from the reviews and findings have been used when creating the rest of the content including the training program and blog.

6.2 Awareness on Web Application Security

This is mainly covered in the chapter Awareness, but is also a red thread through all the other parts of this thesis, as in the training and blog. We have analyzed existing material to find topics that will help with spreading awareness on web application security and used the findings through the other work. We have also used findings about the need for web-application security to find which parts to start the awareness focus on. Most of the initial focus is on developers and testers because of time constraints, it is planned to include more information for managers and analysts after finishing this thesis, see Appendix Continued work for plans for how to continue the work. The work in this part have not been about finding any new information, it have been about finding ways and forums for communication with stakeholders and to trigger their interest.

6.3 Teaching material development for web application security

Results from finding the need for more secure web-applications and awareness among stakeholders was used to create training material about security. The training was created with a focus on triggering an interest for learning more and to understand why it is important to create secure software. It includes some parts where the participants have to think and come up with reasons for why security is important and other parts showing simple examples of attacks and ideas about how to defend against them. More about the training can be found in the chapter Training program and in the Appendix Training Program.

A questionnaire was used after the training to try to find a qualitative result on the effect from the training. The result from the questionnaire will be used in the continued work on software security.

6.4 Experience with blogging about web application security

The blog is created in mostly the same way as the teaching material. The largest difference is that some of the feedback has come after each blog-post, so it has been easier to adjust the content for later posts. Both the direct feedback and visitor statistics can be used to quantify how the different posts have been received. The questionnaire about the effectiveness of the training did also include some questions about the effectiveness of the blog posts, to get a more qualitative result than the immediate feedback after each post. See Appendix Selected Blog Posts for the blog-posts.

7 Awareness

We propose that to be aware of the potential effects of bad security is a good way to start increasing the security. It can be easier to spend time on security when knowing how users can be affected, or that contracts can be lost. The whole team including all roles needs to be aware of how security affects the software: -

- Developers need be aware so they understand why they should create secure code
- Analysts and business owners need to be aware so they understand how to set the requirements and include it in the specifications
- Testers need to be aware so they understand why they should test the security
- Managers need to be aware so they allow for the time it takes to create secure code
- All roles should also be aware so they can run risk assessments and analyses, both on their part and participate in the overall risk assessment.

When all parts in the development team knows and understand security the requirements can be written to make it clear which parts require extra attention, and developers and testers can create scripts and attacks to test the security. And the development team can use compliance and standards as a minimum level for security, instead of setting it as the desired security level. We believe that telling that security matters is not enough to create awareness, both the blog posts and training program includes parts to create awareness and try to trigger an interest. The posts that are most related to awareness are:

- Reasons to focus on security [16]
- Web application security [17]
- Webification [18]

Other posts do also contain elements of awareness.

8 Training program

Training is a good way to learn new information, and is much more structured than reading available information found in books and on Internet. This can be illustrated on the example that schools are still important, and by the raise of available online courses from education institutions.

As a part of this thesis a training program on writing more secure web applications was created. It covers the topics and problems found during writing the thesis, and are made to create awareness and knowledge about security in development teams. The program includes parts from the OWASP Top 10 list, and and there have been made attempts to keep a balance between how attacks work and how to defend against them. The length of the training can be adjusted based on the available time. The training should be adjusted to focus most on the parts the receivers need training about, so some sort of risk assessment should be used to adjust the training. First finding the risks and then quantify them using likelihood and effect to find the most important part to focus on. Examples on how to do this are published on Creating a training program, Creating a training program part two; estimate the risks and Part three, finding what to include. This tree posts were made as part of a short introduction course, but the techniques are the same for creating the larger courses. Since the training program is created based on established best practices combined with a risk analysis it can also be used as a basis for writing secure coding guidelines for the projects.

Parts of the training program were also used in a 1.5 hour course for one development team, where developers, testers, and some analysts did participate in the training. After the training some started on writing guidelines for secure development and all did answer a questionnaire to measure the impact from the training and the blog and see what should be changed in the program. The security knowledge in the team spans from little prior knowledge about computer security to experienced penetration tester, so it is expected that the impact will vary from person to person.

The questionnaire was constructed in cooperation with Elham Vahid who is writing a master thesis about Social Business and Privacy Concerns.

The feedback shows that most participants reports some change in their perception about security, and most got a bit or a lot interested in learning more about web application security. So it is a good start to continue the security awareness and knowledge work from. See appendix for plans for how to continue this work.

The results where:

Will you say you understand more about why security is important after the training?

No change	Some Change	Big Change
1	6	0

If you have followed my blog, will you say it have changed your understanding about why security is important?

Have not read it	No change	Some Change	Big Change
4	0	3	0

Where you concerned about your privacy on Internet before the training and before you did read the blog?

Not at all	Some	A lot
0	6	1

Have the blog and training affected your view on privacy?

Not at all	Some	A lot
2	4	1

How often do you worry about online privacy?

Never	Always	Sometimes
0	3	4

Did you learn more about our security risks from the training?

No change	Some change	Big change
1	5	1

Have the blog taught you more about our security risks?

Have not read it	No change	Some change	Big change
5	0	2	0

Do you think normal web-users should know more about security and privacy online?

No	Maybe	Yes
0	1	6

Did the training teach you more about how to defend against attackers?

No change	Some Change	Big Change
1	5	1

Have the blog taught you more about how to defend from attackers?

Have not read it	No change	Some change	Big change
3	0	4	0

Have you ever tried to prevent data leakage?

No	Don't know	Yes
0	3	4

Did the training get you interested in learning more about web application security?

Not at all	A bit	A lot
2	2	3

Have the blog made you interested in learning more about web application security?

Have not read it	Not at all	A bit	A lot
5	1	1	1

What did you learn most from?

Did not learn anything	Blog	Training
1	1	5

Do you in general feel safe when using web-applications?

Not at all	A little	Very safe
2	2	3

Have the training and other information changed how safe you feel online?

Not at all	A little	Very much
2	5	0

8.1 Comments from questionnaire

8.1.1 What do you consider as sensitive data?

- Login details on different sites
- Personal information and pictures
- Any personal user data
- Any information about the user other than the one that he himself said publicly
- Payment information
- Passwords
- Personal ID number
- email
- password
- Username
- Password
- Social Security Number
- Payment details (card number, expire date, etc.)
- Also everything which normally is sensitive in different areas (tax info, patient journals, grades/exam results, etc.)
- Data the user should expect to stay his/hers private information

8.1.2 If yes to the questions about interest for learning more, what do you want to know more about first?

- First of all I want to read your blog, because for now I have no time for that
- XSS
- CSRD
- Buffer overflow

- How to prevent your web-application from having security risks
- Security related to mobile devices

8.1.3 Do you have any recommendations or suggestions for changes to the course?

- More real examples of hacks and prevents. Maybe some videos or schemas
- No, I don't think so at this moment
- Maybe some examples of how you can improve your code and way of thinking in order to make safe web-applications
- Maybe use more live examples, to visualize security faults. Can sometimes be difficult to understand what the actual problem is without good examples
- Possibility to include some real (or "real") world examples. It should be possible to quickly demonstrate a xss -> java drive by resulting in total control of a test laptop running windows.

8.1.4 Do you have any recommendations or suggestions for topics to add in the blog?

- I should start to read it. Maybe I will have recommendations later
- Security related to mobile devices

9 Quality and Security

Quality is security[7]. An important part of creating a secure system is to create a system of high quality. With the increase in complexity it is difficult or most often impractical to test all parts for the system for each change. Because of the increased complexity the importance of automated testing has continued to increase. One of the solutions to this is unit testing, and test driven development. Some prefer to start with writing the tests and others like to first create the code and then the tests. There have been some research on what is most efficient, but there is no conclusion on which way is best[19]. The important part is to test the code to see that it does what it is intended to do, and that it still does it after refactoring or changing code. Complex code is harder to read, so harder to review for security. Unit tests can also include testing for security, so that they will tell if some changes to the code create vulnerabilities. As with unit testing for correct functionality these tests will only show that what is tested is correct, it is not a substitute for other testing tools.

Both the blog and the training program include the importance of quality, and that creating clean testable code not only helps in the quality but also with security, since it is easier to find security issues when the code is easy to read. The connection between clean high quality code and security is included in many places both in the training program and in the blog posts. It is described most in the posts Security vs quality and Web application security.

10 Risk and Cost

Building security costs money, having security problems can also cost money. Security does not happen by chance, it needs to be planned and build in. How much time and effort do use on security should be based on a risk assessment. A common way to compare risk is to use the formula: Risk = Likelihood * Impact. This will tell how to prioritize the different security measures and if some none standard have to be included.

It is clear that an internal static web-page only accessible from inside the corporate network do not need the same level of security as a banking site. How large the difference is will be shown within the risk assessment. The likelihood of a security breach in a page with no user input that is only accessible from the intranet is much smaller than in a banking web-application and the impact is also much smaller. The different parts of the applications can also be assessed separately, and against different attacks. Conducting a risk assessment will also be connected with the awareness in the project, it will show which attacks are most likely to happen and what the impact of the attacks can be. On the other hand the ones doing the risk assessment need to know about security, what kind of attacks that can happen and the affect it can have on business. This shows that it is important that business process owners and managers understand the security implications so they can participate in weighting the risks with the added cost for avoiding or reducing the risks.

There exists several frameworks for risk assessments, but assessing and evaluating the different frameworks is outside of the scope for this thesis, so the recommendation is just to find and use a framework. The frameworks should not be used to create documentation that no-one is reading, they should be used as ongoing activities for thinking through the risks and how to handle them. Frameworks like CobIT [20] with Val IT [21] and Risk IT [22] are powerful tools that can be used to get a holistic view on both risk and values connected to IT investments. They might seem large and heavy for new users, and it is possible both to find lighter frameworks to start with, or to start with using only parts of the frameworks. Using them requires a more thorough work with the risk assessment, where the participants have to take more conscious decisions on the security. Using the frameworks will also show both the cost of implementing security measures and the cost of not including them. So the management can use them when deciding on projects and creating budgets.

11 Vulnerability testing

Without doing any vulnerability testing it is easy to claim that the software is released without any known security vulnerabilities. The problem is that it does most likely have plenty of unknown security vulnerabilities. One of the benefits with creating software for Internet is that it is possible to get out to a vast number of peoples, and one drawback is that the software can also be attacked by a vast number of peoples, many of them also located in different areas with different computer crime laws. It is easy for attackers to set up scripts that search for common vulnerabilities, so it will not take long after publishing before this kind of scripts starts to attack the software. So vulnerability testing should be done before releasing the software so as many as possible of the vulnerabilities can be found and removed. The testing should also continue after the software is released, changes to the software, dependent components or configuration can introduce new vulnerabilities, and new attacks can be found that can break the existing security measures. Vulnerability testing should be a part of the normal QA[1] work, and tools and experts can be included to do more testing. Development time and testing is most often limited in time and cost, attackers have all the time they want and do not have to care about budgets and release deadlines. So it is important that the conducted vulnerability testing as thorough and as total as possible. To achieve this, security testers should have knowledge about ethical hacking and penetration testing and stay updated with new attacks and tools.

Vulnerability testing can also be used to test the whole architecture in the applications. The testing should not be limited to the code written by the project, the combination of applications, server software and other components can also be vulnerable. Attackers do not only go after one application, their goal is to get access not to break an application. OWASP Top 10 2013[15] include both Security Misconfiguration and Using Components with Known Vulnerabilities among the ten most common mistakes in web application security. QA and production environments are often in theory the same, but it is too easy to get small differences, so vulnerability testing should also be used against the production environment. This testing will also show if the intruder detecting system and other preventive tools installed catches the attacks, and if notifications and response is handled in a good way.

[1] Quality Assurance

12 Static analysis, Web Application Firewalls and other tools

Static analysis is a part of testing where the code is checked for common issues, it exist tools for both code quality and security. The metrics from these tools will show where vulnerabilities are found and will show places in the code that should be checked and most often rewritten. The previous mentioned report from WhiteHat security did also include that 39% did include some amount of static code analysis on their web-applications, and that these had 15% more vulnerabilities and resolved them 26% slower than average. It did also show that 55% used a Web Application Firewall (WAF) and these had 11% more vulnerabilities and resolved those 8% slower. This shows that it is important to not depend on tools like static analysis and WAFs; they should be used as a supplement to be added on top of the other security work. Other companies like Cenzic claims that 99% of tested applications have on or more serious security vulnerabilities like cross-site scripting and SQL injection[23].

These statistics are from companies selling security scanners and training, so they are of course biased and the seriousness might be exaggerated. However this numbers shows that we still have a long way to go before we get a secure web. Static analysis can be used to check for common mistakes both in code quality and security, a CI[1] server can be set up to run analysis of the code in the same way as it is set up to run unit-tests, at regular intervals when code is checked in to the version control system. Most CI servers can be set up with predefined rules and can be configured to add rules defined by the team.

Even the statistics from companies selling tools for static analysis and web application firewalls shows so it is clear that the tools should be included as a part of defense in depth, not as the main level of security. Applications should be secured as good as possible, and the risk assessment will show when it is worth it to invest time and money in tools. Results of vulnerability scans can also be used to guide in this decision on where to focus the security work. It will only be in some special cases where using some sort of static analysis not will be worth the effort. Several free open source projects exists that can be used to start with. Web Application Firewalls can help in defending attacks until a security patch is developed, tested and set in production.

Static analysis of security could be a good addition to other static analyses and test tools used in automated code review, and used in the same way as these tools to help look for problems. Not to be the only place for testing. The results from this scans should first be checked by the development team for false positives, and all should also remember that these tools not necessarily finds all vulnerabilities.

[1] Continuous integration

13 Software Lifecycle

The software lifecycle is divided into five main parts that all have impact on security. The border between the parts is more visible in some development methodologies than in others. In waterfall development there is a distinct border between the different parts, and one part is finished before the next is started. In the other end there is agile development where all parts are done almost at the same time, and much less focus is put on the initial specifications. The different methodologies have different pros and cons, and assessing them is outside of the scope for this thesis, the focus here is on a more general level. Focusing on security in all parts are referred to as Security Development Lifecycle (SDL)[13] or Secure Software Development Life Cycle (S-SDLC)[24]. This is not a process that conflicts with waterfall, agile or any of the other development processes, it is a part put on top of any process. It is to get a more holistic security focus build into each of the different parts of the software lifecycle.

13.1 Requirements and Specifications

A software project does normally start with gathering requirements and creating specifications from them. This can be done before the development starts, or it can be done in parallel. This is the part where everything that will be included into the software will be found and described, both functional and none-functional functions is included here. Security should[1] be treated as a part of these requirements, and the desired level of security should be included in the specifications. Access right to CRUD[2] operations is also defined as part of the requirements. Any special considerations, like requiring a higher than normal level of security or special regulations that should be met must be included in the specifications, and a risk assessment including the different stakeholders should be included to decide on these requirements.

13.2 Development

Developing is the largest part of a software project; this is where the actual product is created. It is also a critical part when coming to the success of the product, the best plans cannot succeed if they are poorly implemented. The same is the case of the security in the project, the best plans and risk assessments do not help if they are not followed and incorporated into the code. Developers need to know and understand what is required to build secure software. Then need to know about attacks and how do defend from them. They should also be aware of the risk

[1] Should and could is here used instead of is and must to show the difference between recommendations and normal practice

[2] CRUD is short for Create, Read, Update, Delete

assessment to better understand why some parts are set as more important than others, and be given the time required to write secure code of high enough quality. The developers should also be part of creating guidelines for secure coding, however the CISO[3] should be included to guide the team in this work. If there is no CISO someone should be appointed with higher responsibility for security and get extra training. Application security is too important to be treated in an ad-hock manner.

13.3 Testing

Testing is the phase where the produced code is checked to see if the code do what is required and described in the specifications. The testing can be divided into two parts, automated and manual testing. Both parts should also be used to verify the security of the product. Static analyses and unit tests can be used to both check the code quality and the security, however they can only be used to check what is included in the tests. Attackers use tools for scanning for vulnerabilities and attacking applications, the same tools could be used by the development team to scan and find vulnerabilities before the product is released. Static analyses and unit testing have been mentioned before, however they are useful tools that is worth mentioning again. Modern software is so large that it is impossible to manually test all parts using all valid and invalid input, so as in other places of software development it should be automated if it could be automated. Keeping all tests and keep adding to them will give increasing test-coverage, and running the tests costs close to nothing. Not all part of testing can be automated, how much manual testing that is required for validating that the business requirements are met is out of the scope for this thesis. Manual testing can also include ethical hacking and penetration testing, both white-box[4] and black-box[5] testing can be done and it can be done manually and including tools that can be more or less automated. See also part about vulnerability testing.

13.4 Deployment

It could be discussed if deployment is a part of the development process or not, but it should be obvious that it is an important part of securing the software. It does not help how secure the software is if it requires insecure components or if the servers it is running on is not secured. It can also be discussed if included components are a part of development or deployment, in many cases it can be a part of both. It is here placed as a part of deployment for simplicity. It is also a part of the development process to create applications that could be secured in deployment, both the architecture and the involved components should be designed to make it easy to deploy and to avoid dependencies of outdated versions of components. Upgrading to newer versions of components and including patches to components is a part of maintaining the software, and could be a critical part of security. Software should have secure default settings, so that the security does not depend on remembering to set some hidden setting. It should be clear which versions of

[3] Chief Information Security Officer
[4] White-box testing is where the code is known to the tester
[5] Black-box testing is where the tester do not have access to the code, so resemble more what attackers sees

components and dependent software, like web-server software that is required and they should be kept updated and secured. Software should also be created in ways that make it possible to monitor, detect abnormalities and fail in secure ways. So overall the software architecture plays a significant part of the deployment and daily operations.

13.5 Maintenance

Maintenance, in the same way as deployment, is often overlooked when discussing the development process, especially in literature. In practice this is often the largest part, maintenance consist of two parts; bug fixing and changes to the functionality[25]. This can be a critical phase in security; changes to the code can indirectly influence other parts of the code and introduce new bugs and vulnerabilities. Unit tests can help with making it easier to check if changing code breaks other parts of the code, however it is not enough. Every change will have to be tested, and often a full system test is required after making changes. If a full system test is not feasible all tests must be run, including tests run by vulnerability scans.

When looking at the list of bug-fixes from large software products it is normal to see that fixing bugs in one place creates bugs other places, which then have to be found and fixed. This is an effect of the complexity in software, and shows the importance of controlling complexity.

Changes to functionality is a, for most, surprisingly large part of maintenance, and as the application gets older new developers will start working with it and after a while it is possible that the whole team is changed. When these changes happen it is important to have good documentation and clean code, to give as much help as possible to the ones that have to change the code so they can make sound assessments on the security implications of the changes.

It is easy to be too quick when changing code and introduce unnecessary complexity, especially when not knowing and understanding the code base, and when not having good unit-tests to check if refactoring to reduce the complexity is done correctly.

13.6 Findings and conclusion

We have found, as expected, that there is a big need for improved security in web-applications and that there is a need for more awareness and knowledge about security in development organizations. It is a big variance in how much focus different projects have on security, and even the big companies with a good focus on security have vulnerabilities. So there is a need for training and information to make development teams more knowledgeable and more aware on security. It is important that development teams have good knowledge about security and that the security is tested. The time for finding vulnerabilities is limited during development; however attackers have all the time they want when attacking.

The results from the surveys and other feedback on the training program and blog shows that teams and developers that think about the consequences of bad security, and are aware of what

it can lead to, get more interested in learning more and puts a larger focus on security. Giving training in person can be a good way of reaching out to a limited sized group, and creating a blog can reach many more. The training can be more focused in which parts are covered, while the readers can select the blog-posts they find interesting and want to read. In their own way both blogging about security and providing training can help influence development teams to create more secure software. The content should be kept up to date on current threats, and it should be updated with new content to give readers and participants a reason for coming back to learn more.

One important thing to remember is that focus on security is not something that can be done once or at one point during development, it has to be part of the overall process. Security need to have focus all the way from ordering the software through maintenance. Creating awareness and knowledge is in the same way not something that is done once, training and motivation should be included at regular intervals. It is important to also follow a process to keep the application secure and be ready for new types of attacks.

Appendices

A Continued work

The work started within this thesis is only the beginning for work on improving the security in web-applications, both in the company where I work and in general. Several blog posts are in construction or on the idea stage waiting to be written after this thesis is done. It is also scheduled to have the training for several teams and departments, and I will be working on creating secure coding guidelines for the whole company. It is also agreed that I will take a greater responsibility with working on the overall security in all our software.

B Training Program

The texts are adopted from a PowerPoint presentation with notes.

Text written in italic is suggestions and reminders.

B.1 Introduction to Secure Coding

B.1.1 Notes

Welcome to this introduction course in secure coding. This is not about writing security functions like access control or encryption; this is about securing the normal applications and web-pages you/we are working on every day. Most in this course is about web-application, however many of the attacks against web-applications can also be used against traditional software if it integrate with web-applications.

We will look at the most common vulnerabilities in web-applications and see some ways to mitigate them.

Feel free to come with questions or comments any time, an important part of this course is to create curiosity and interest.

B.2 Why?

- How many reasons for caring about web-application security can you find?

B.2.1 Notes

Have the participants come with ideas and write them on a whiteboard. Use about 5 minutes or until they don't find more reasons.

B.3 Why? Continued

- No wrong answers
- Depends on the project
- Examples

  - Protect users
  - Avoid bad press
  - Legislations
  - Avoid having the site blocked/blacklisted
  - Protect company assets

B.3.1 Notes

There is no wrong answers to why you need to focus on security, the reasons depend on the project. The important first part is to do what we did now; find reasons for why your project need it so you can find which parts to focus on.

This can also be used for a risk assessment to document the process and think more through the reasons.

The reasons listed here is only some suggestions and a way to show that it can be a large span in reasons.

B.4 Web basics

- Client - Server
- Traditional web stack
- Part run on client side
  - Potential hostile environment
- Parameters from client
  - Data from other sources

B.4.1 Notes

You do already know this, but it will not harm to go through it one more time. Web-applications uses client - server architecture with the traditional web stack of HTML, JavaScript and CSS. The JavaScript code is of course run on the client side, at least when we don't consider versions that is run on the server side. The users will have full control of the client side, and can change or disable the scripts and even change the parameters before they are sent to the server. This means that you cannot trust any of the input from the client side, and that validation on the client side is there to make a better user experience, not to do anything with the security. While we are on the topic of not trusting data from the client, you should also not trust data from web-services or any other API. You don't know what kind of validation that has been done to it, so you should do your own validation.

B.5 Quality

- Quality as part of security
- Testable code
- Include tests for illegal data
- Quality in all parts

B.5.1 Notes

This could be the part managers and testers like the most. We should not only treat security as a part of quality, but also acknowledge that quality is part of security. Bugs are code that does not work as expected, so you will not know if how the security is handled since you don't know how the function will work. Creating clean readable code will not only make it easier to find bugs, it will also find it easier to find security problems. Then the code should be testable, and you can create good unit tests, and in these tests you could also include tests with attack code to be sure that no changes open up for attacks and that the system don't start to leak information when it encounter invalid input. Security is also affected by the quality in all parts of development, from requirements and specifications, through architecture and coding to testing. Access control and if extra security is needed should be part of the requirements and be reflected in the specifications. The architecture should enable the necessary security requirements and access control requirements. Testing should also include testing of security levels and measures.

B.6 OWASP Top 10

1. Injection
2. Broken Authentication and Session Management
3. Cross-Site Scripting (XSS)
4. Insecure Direct Object References
5. Security Misconfiguration

B.6.1 Notes

The Open Web Application Security Project (OWASP) has lots of different projects working with creating resources about creating secure web applications. In this course we will focus on their top 10 list where they have listed the most common security errors in web-applications.

B.7 OWASP Top 10 continued

6. Sensitive Data Exposure
7. Missing Function Level Access Control
8. Cross-Site Request Forgery (CSRF)
9. Using Components with Known Vulnerabilities
10. Unvalidated Redirects and Forwards

B.7.1 Notes

Now you have seen a list of the parts we are going to cover, is there anyone that has a good reason for parts we should not include?

B.8 User Input

- Do not trust it

B.8.1 Notes

There is only one thing to say about user input, you should never ever trust it. This is the same as was said in web-basics, but it is important enough to be repeated. Do not trust any input from the user.

B.9 SQL injection

- Definition
- Code injection (' or 1=1; –)
- Logical bugs (like '%')
- Parameterized prepared statements
- Always use parameters

B.9.1 Notes

SQL injection is when users/attackers are able to include SQL control characters into input, and it is not properly handled.

Code injection is when characters like ' is included to break out of a string so the attacker can continue with his own code. One example is to add 'or 1=1; –. The first apostrophe will end

the current string, or 1=1 will, as you all know, make the query return all rows, and – is used to comment out the rest of the query. The two common ways to fix this is to use a library that uses parameterized prepare statement, or to use a library that uses parameterized prepared statements. Prepared statements are often mistaken as the way to fix this, but they will not help if input is concatenated into a query. It is registering the input as parameters to the query that helps against SQL injections.

Logical errors cannot be handled by libraries or parameters, they are of the type where like is used where it should not allowing users to send in wildcard characters to get around limitations that where intended.

We could have used lots of time coming up with attacks and defenses to SQL injection, but they are all ways to try to secure the application when not using parameterized prepared statements or libraries that use it.

Remember to always use parameters, even if you use a library like hibernate or another ORM. They will not be able to help if you concatenate user data into the query.

B.10 Cross site scripting (XSS)

- Both easy and difficult
- <script>alert(42)</script>
- +ADw-script+AD4-alert(1)+ADw-/script+AD4-
- </p><script...
- «SCRIPT>alert("XSS");//«/SCRIPT>
- <scr<script>ipt>alert(1)</scr</script>ipt>

B.10.1 Notes

It can be both extremely easy and very difficult to defend against XSS. The easy way is to not have any user generated input shown to other users, including not retrieving any data from other sources.

Not allowing users to do any formatting to their input, and only showing it inside of certain HTML elements is also pretty much straight forward.

Allowing users to insert HTML tags that will be used to render the sites makes defending much worse.

The problem is knowing which parts to allow and which to not allow, and following the rules on what is allowed. The first script tag should be easy to see, the next UTF-7 encoded version is not

that obvious. Most browsers do not have problem with mixed encodings anymore, so it is not that easy to get the UTF-7 version to work.

Care has to be taken when trying to remove illegal characters to "fix" the input so it will not be harmful. Extra closing tags can be used to break out of the current tag and start a new tag, in the same way as apostrophe can be used to break of strings in SQL. Removing the illegal tags might also not be enough, it should be easy to see what is left when <script> and </script> is removed from this string.

Input that will be shown without any user-generated markup should always be HTML encoded if used inside HTML, and URL encoded if it should be used as part of a URL.

B.11 Find reasons to defend against XSS

- Can you come up with anything you can do to exploit a page with XSS-vulnerabilities?

B.11.1 Notes

Use about 5 minutes and write down the reasons the participants can come up with. This is to get interaction and to get the participants to think about this.

B.12 Examples of why defend against XSS

- Steal session cookie
- Insert iframe
- XSS-worms
- Part of Cross Site Request Forgery
- Have fun with others
- Vote/like forgery

B.12.1 Notes

This is not an exhaustive list; it is only some examples of what can be done.

Stealing session cookie is only possible if the site also have broken session management.

An inserted iframe can be included to load attack code from other domains, for instance by including code that attacks browser add-ons that are vulnerable to get access to the users computers. This is also known as drive-by downloading, users get infected by visiting "trusted" sites that have been compromised.

XSS-worms, worms that spreads to user profiles on forums.

We will get back to cross site request forgery a little bit later.

Pranks against other users.

Vote/like forgery is close to cross site request forgery, but against sites that expect requests from other sites.

B.13 Headers

- X-Content-Type-options:nosniff
- X-XSS-Protection:1;mode=block
- X-Frame-Options
- X-Content-Security-Policy

B.13.1 Notes

Browsers cannot know what is intended code and what is payload. But you should give them as much help as possible. There are several headers that can be used to help with this, you should set them all.

X-Content-Type-options: nosniff is set to prevent browsers from trying to second guess the MIME type set by the server. Without it they might try to check the content to see if they should override the MIME type and change to another.

Most browsers do have some XSS protection build in, but it is possible to turn it off. Users can have different reasons for doing this, but you should not create a site that requires that it is turned off, so turn it on for your site by setting this header.

X-Frame-Options can be set with different levels, deny, same origin and allow-from URL. Set the ones that are right for you so you get control over where your application can be included in an iFrame.

X-Content-Security-Policy is an extended policy you can set on what is allowed. You can for instance disable inline JavaScript, so that all JavaScript code has to be loaded inside of the head tag.

B.14 Other defenses

- Whitelist
- Use existing libraries

- Java https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project
- PHP http://htmlpurifier.org/
- JavaScript/Node.js https://github.com/ecto/bleach
- Python https://pypi.python.org/pypi/bleach

B.14.1 Notes

Setting the security headers is not the only part that has to be done, it is only the beginning.

Some try to create blacklists with what is not allowed, I hope the examples shown a bit earlier have shown reasons for why this is not a good idea. You will never be able to find and cover all the different ways of getting around a blacklist. Attackers will have much more time than you to look for ways to insert data that will get through your list. You should create a whitelist and only allow the tags and attributes defined in it. Discharge any data that isn't allowed by your list, don't try to clean up and remove parts of tags to make them legal.

Don't be a victim of "not invented here" and try to create your own filter when you can use existing libraries.

B.15 Authentication and session management

- Secure the passwords
- HTTPS
- Session
- Use existing standards

B.15.1 Notes

As you hopefully remember, one of the main reasons for creating secure software is to protect the users. So it should be obvious that their passwords should be protected. Attackers can get hold on the passwords in several different ways. The servers can be compromised, SQL injection vulnerabilities can be used to find passwords, or a trusted insider can copy them. Clear text passwords are of course easy to steal. But it is also easy to get the clear text version of passwords hashed with a normal hash function, because hash functions are made to be fast not to protect the hashed data. If you have to store passwords you should always store them using bcrypt, scrypt or PBKDF2, they are all created for storing passwords so they are relatively slow and will not run much faster on GPU instead of CPU, as is the case with normal hashes. Passwords should have a minimum length, and can include requirements for complexity. The maximum length should only be as a defense from DoS, avoiding that someone try to post a gigabyte size password.

There are still sites that include user login that is not using HTTPS, don't be one of them. Some do also only have HTTPS for the actual login and HTTP before and after, that do only secure the password (if it is sent as a POST parameter, yes some sites allow username and passwords through GET) it will still be simple for anyone on the same network to sniff and find the session id.

You do also here have some headers that can be set to help, the HttpOnly flag should be set so that the session cookie cannot be accessed through client side scripts. AUTOCOMPLETE=OFF could also be set so that the browser will not cache the credentials. If the session is set on a HTTP site, a new session id should be created after login, or even better always create a new session id after login to be sure. Remember that the session id is the only authentication that is needed after the user have logged in. Some frameworks includes the session id in the GET string if it is not able to set the session cookie, this should always be disabled.

The rule here is as always with security; do not try to invent your own security mechanisms, use existing standards and implementations. Even large credit card companies like VISA and MasterCard can make mistakes when implementing existing standards, they did have site channel vulnerabilities in the early implementations of the chip on their cards.

B.16 Cross Site Request Forgery (CSRF)

- Requests started from other tabs
- Relatively easy to defend
- Flash vulnerability

B.16.1 Notes

Requests to your application from other sites, do only work if the user is authenticated in your application and a request is made from another site that runs in the same browser thread.

It is easy to defend against CSRF, as long as you remember to include it. You can add a hidden random value to all forms and in session, so you can compare it when to see if the post was done from your form or from another place. You can also set a header and always check that it is present, X-* headers will not be included from other domains when doing Ajax calls. The problem is that Flash had a vulnerability which allows attackers to spoof custom headers. So try to use a combination of these two techniques.

As a bonus, can anyone see why we should care about a vulnerability in Flash when we don't use any Flash on the site? (The site where the spoofed header is not ours, that is the whole problem with CSRF)

B.17 Components with known vulnerabilities

- Know your components
- Update libraries
- Stay updated on vulnerabilities

B.17.1 Notes

It is not only your code that could have vulnerabilities, all code can have it. It do not help how secure your code is if it relies on components that are not secure, or the application is only as secure as the weakest component. So what can you do?

You have to know which components you are using so you know what to check for. After finding all components you should make sure that all is updated, and make sure that they are updated as soon as possible after new releases is out.

Keep an eye open for vulnerabilities in the components so you know when you will have to update them and when you will have to create temporary fixes while waiting for official fixes to the component.

B.18 Unvalidated redirects and forwards

- Don't use redirects
- If you have to, don't involve user parameters

B.18.1 Notes

Including redirects based on user input will mean that you don't know where the redirect is going. If you need redirect based on input from users, don't use the text they have inserted. Instead use the input to look up the correct place to send redirect.

B.19 Testing

- Functional testing
- Automated testing
- Access control
- Input validation
- Penetration testing

B.19.1 Notes

There is several kind of testing that should be done in development projects, all of them should be included but it is common to forget about some of them. Functional testing is the normal testing that almost all projects do before releasing software, and it is used to verify that the functionality works as expected. Automated testing can be both unit-testing and automated functional testing, where tests are run on set intervals to check that changes done to the code don't change anything unexpected. This two types of test is most connected to quality, the next three types is more about security.

Testing of access control is borderline between security and functionality, it can be both testing that users get the access they should get and that they do not get access they should not get. It is also related to the last two types of testing we will cover now.

Input validation is to try to break the security by injecting attack code into input values, it can both be done manually by inserting data into input fields, by using proxy functionality to change data before it is sent to the server, and as a part of penetration testing. We will soon cover penetration testing a bit more.

B.20 Code Review

- Code quality
- Readable and understandable
- Find vulnerabilities

B.20.1 Notes

Code review can be used to both check code quality and to find vulnerability. Developers tend to write cleaner more readable code when they know it will be reviewed, and the reviewer should also send it back if something is not understandable. The code review should not only be used for finding quality problems but also to find security vulnerabilities and potential vulnerabilities.

B.21 Penetration testing

- Manual
- Using tools

B.21.1 Notes

This is only an introduction to creating secure applications, so we will not cover penetration testing, but it deserve some words. It requires some training to be good with penetration testing,

but as with everything else you have to start before you get experience. I recommend to first start with doing some manual tests by input manipulation, in order to understand how it works. You can start with the techniques for injection we have been talking about today and work your way up to more advanced versions. The basic recommendation is to have fun and always think; "what happens if I try that?" After you have started to understand some on how it works you can find tools and let them help you with the testing. Some (or most) of the tools can also be used by attackers, so always check with IT operations and security to find which tools you can use and to let them know about the tools.

B.22 Operation and servers (Security Misconfiguration)

B.22.1 Notes

It is easy to think of the operation of the software as IT operations problem; however development should create software that it is easy to operate in a secure manner. The software should not rely on insecure settings on the servers or on the client. Work together with operations to identify potential problem areas.

B.23 Access control and storage

B.23.1 Notes

We have already talked about testing access control, so this is only a reminder. You should be sure that users only get access to what they should get access to, don't let them in any other way. It does not help to use the most advanced 2 factor authentication if the database is open for anyone. You should also think about what you store and where, private information like passwords or credit card numbers should not be stored as clear text. Always store them using appropriate methods, passwords should be stored using a cryptographic password storing function like bcrypt or similar. When you have to store information where you will need to get the clear text back, like credit card information, be sure to handle the encryption keys in a secure way and keep the encrypted text on different servers from the rest of the application.

B.24 Encryption and security mechanisms

- Do not create own encryption
  - Do not even try to implement standards

B.24.1 Notes

It takes expert years to first create and then implement encryption mechanisms, and then other experts try to break it for years before it is suggested to use it and before it can be a standard.

So there is no chance that you will create anything that is better. Don't even think about it. Even the work of implementing existing standards is difficult and will introduce vulnerabilities. You will not only have to know how the standard works, you will also have to care about side channel attacks. Many developers have tried to implement encryption standards after reading Bruce Schneiers' Applied Cryptograph, here is part of what he later have said about the book: "Readers believed that cryptography was a kind of magic security dust that they could sprinkle over their software and make it secure. That they could invoke magic spells like "128-bit key" and "public-key infrastructure." A colleague once told me that the world was full of bad security systems designed by people who read Applied Cryptography."

B.25 Encryption and security mechanisms

- XML
- LDAP
- etc.

B.25.1 Notes

It is not enough to know about SQL injection and Cross site scripting, every format you use for data transfer might be exploited. It is possible to get XML injection, LDAP injection and any other kind where text is stored or interpreted.

C Selected Blog Posts

C.1 Creating a training program

I have decided I include all major new texts as both blog posts and parts of the web-pages, to more show the progress and which parts I am working on. As stated earlier I will make some training programs about how to improve the security in software. I was asked to also create a shorter introduction about this to use soon, before the rest is done. That made me start thinking more about how to actually create these programs, and this post will be about ways to create an efficient training program.

It is so much information about how to secure software that it is difficult to know where to start. If you want to improve the security in the software you are working on, you should use the needs for your project when doing research. If you are in a team one person could get the responsibility to do some research, learn something, and then train the rest of the team with what was learned. If you are alone you should also look at what do you need before starting to learn.

The first step is to figure out what the software is doing or going to do, and what kind of information it is going to store and show. If it do not store user credentials and handle sign in, you do not have to read up on how to securely store passwords. If it will store passwords you should find something about how to store them securely, see section that will be added about that in this page. In the same way you only have to know how to avoid security problems with a database if you are using a database, and you do not need to know much about challenges with a specific database you are not using. You could start with making a list something like this:

What	Using	Comments
Storing passwords	No	Will always use an external identity provider
Using database	Yes	Will be using a MySQL database
Programming language	Yes	We will be using PHP and JavaScript
Database connection	Yes	Have to find what is best and most secure for us
Can users communicate	Yes	All users can write something that other can see
Web-services to external API	Yes	We will connect to Facebook and Google+
Store sensitive information	No	Our risk analysis shows that none of the information we are storing is sensitive information. However we still want to protect it, so the users can control who will see what they posts
Others		Fill in all other relevant information

Table 2: Finding what the software is doing

You should of course make it based on your information.

After the list is created you can start with trying to quantify the consequences with security problems in the different parts and the likelihood that you vulnerabilities combined with the opportunity others have to exploit those vulnerabilities. After that is done you should start looking at what you could learn about those parts and see how you can teach others the same.

If this seems like risk analyses to you, it is because it is part of that. You need to know what you are protecting before you could know how to protect it. This is yet another example of a situation where techniques learned about one thing, get handy whey doing something else. This is where you can put the theory you have learned or is learning into practice. If you also start learning about the process part you will see that code security training often is the first step, for instance in `http://www.microsoft.com/security/sdl/discover/training.aspx`. More on this will be included in the process section.

The next steps with how to assess the risks and creating the actual program will follow. Stay tuned for more...

C.2 Creating a training program part two; estimate the risks

Now you should have made a risk assessment, either by using the simple version in stage one or by a more extended method like Cobit, Val IT, or similar, so you know some more about the risks to the system. The next step is to try to quantify the likelihood and effect if any of the parts is compromised. This is to get a better knowledge of where to start.

If we continue with the table created in the risk assessment, we can take the parts used and add columns for likelihood, effect and sum. In this example we will use a scale from 1 to 5, where 5 is most likely/most effect and sum = likelihood * effect

What	Likelihood	Effect	Sum
Database exploits	2	5	10
Programming language exploits	1	4	4
Database connection	2	5	10
Communication between users	3	4	12
External API	2	2	4

Table 3: Risk assessment

This is only some example numbers and they are both to compare between the different parts and to see where to put most effort. It does not say that there should not be put any effort on securing the parts with lowest score. It should be used to see where to put in most of the security testing, and where to use most time to learn about new ways to exploit the code.

C.3 Part three, finding what to include

Now that you have found where it is best to put the focus it is time to start looking up some key points about the different parts. The OWASP Top 10 project is a good place to start. We can see that Injection still is at the top of that list, and cross site scripting (XSS) is number three. So we were right in putting database and communication between users as high risk parts. Number two is authentication and session management, we did specify that we will not handle authentication ourselves. However we will have to authorize users and handle session, so we will have to keep that in mind.

We can then go on with creating a list of topics to include:

- Use parameterized prepared statement, to secure against SQL injection
- Use as little privileges as possible for the database user our application uses to connect to the database
- Use filtering, or none HTML code for the formatting users can do in input text.
- When using filtering try to use whitelist instead of blacklist
- How to test for vulnerabilities
- Automated testing, unit-testing

Now we should have enough information to start creating a short introduction for developers to be more aware on how to secure our application. The next part will be to create an approximately 10 minute presentation about this.

C.4 Security vs quality

Is there a connection between good quality and good security, or to put it another way; will better security give better quality? My answer is a definite YES! since each bug is a potential security problem. Many vulnerabilities is based on finding the edge cases that is not checked for and exploiting them. So we should be careful with putting a label on how much time we use on security when we develop, since finding and fixing security problems also is finding and fixing bugs in the code. So securing the product is to make it more robust. The main difference is the techniques and mindset used when testing and looking for the weak spots in the code.

When we start to see it this way, we could also start using unit tests to not only check that the functions works, but also check that they fail in a predictable way and don't leak unnecessary information on errors. Code that can be tested with unit-tests will also be easier check for security problems. Functions should be short, not have too many execution paths, and not have side-effects.

It is not only the code quality that affects the security, quality of specifications and architecture is of course also important. Correct access control, and correct level of encryption do also have to be enforced. This and techniques to secure the software is what this site is about, so keep reading. It is not much about general code quality, even though it is, as shown, an important part of good software security.

It is possible to argue if software with high quality always has good security, but it should not be any problem to see that software with bad quality also will be insecure.

C.5 Reasons to focus on security

It is common to hear that "our programs or web-sites are so small, so they are not a target". Or "as long as you stay in the nice neighborhood, and avoid the shady areas of illegal file sharing and adult sites, you are safe and will not get hacked or infected by virus". This is no longer the case, web-crawlers and scripts can be used to automatically search for vulnerable sites and users and exploit them. A common reason to attack a web-site is often to use it as a platform to attack the users of the site. After a site has been compromised it can be used to attack users by including scripts and hidden elements on the site. So your site might be compromised and used to infect visitors without you even know about it. You do not want to see a warning like "Reported attack site" or "This site may harm your computer." when you open up your web-page, or have users report that they get this warnings instead of your site. If you are getting this warnings, head over to `http://www.stopbadware.org/my-site-has-badware` and get some help on how to cure the infection, and then continue reading here so you can be better prepared the next time someone is trying to infect you.

There are often stories in the news about programs or applications that have vulnerabilities that have been exploited. The vulnerabilities is found in products from both big and small companies, so resources do not seem to be the common factor, the problem is more likely connected to corporate culture and prioritising. Lists of the most vulnerable software can be used to see this; Microsoft who used to dominate the lists has disappeared after they started to focus on security through the whole development lifecycle. `http://www.securelist.com/en/analysis/204792250/IT_Threat_Evolution_Q3_2012#14`. This list should not be used to say that one application is more secure than another, it is only showing which security bugs that have highest impact.

More will come in a later post, and as usual, the different sections will be put together on the web-page, where they also might be changed a bit to be easier to read.

C.6 Web application security

Modern web-application, so called web 2.0 applications, do have the same security challenges as web applications always have had and the challenges traditional client server software have.

However this might be more difficult to see when writing the code since some parts of the code is executed on the client side under users control.

First let start with what we define as a modern web 2.0 application, in this kind of web-applications a big part of the program exist and executes in the user's web-browser. So that the application looks and feels more like traditional software that is run locally. Instead of having the look and feel from old web applications, that uses forms which might be linked together by requesting new pages or frames from the server. Applications running in a browser add-on, like Flash, Silverlight, Java-applets, or ActiveX is not covered by our definition of modern web 2.0 applications. However they do also have the same vulnerability issues with being a client server application, running on a potential hostile environment. More about the meaning of a potential hostile environment will follow later.

When creating a modern web-application security consideration from both traditional web technology and from client server technology have to be considered. The client side code can work with data from the user and send it between different parts of the application, and with HTML5 it can also be stored locally between sessions. If the data should be accessible for others, or from other browser instances, they have to be stored on a server. This is the client server part of it, and where the big problems get introduced.

Let us now have a look at what is going on under the hood. The user interface is created using the traditional web tools, HTML, CSS, and JavaScript. All three parts are transferred to the client from the server, and is clear text on the client. The communication between server and client might, and should, be encrypted, but the user will see the decrypted version of the code and everything else that is transmitted. The encryption is only to secure the transport between server and client, not to secure the applications or communication from attacks by the user. It is common to use some functions to minimize the JavaScript code, and sometimes the CSS, to reduce the size of the code that have to be transmitted. This makes the JavaScript harder to read, but do not in any way increase the security. The browser uses the HTML and CSS to render web-pages, and the JavaScript is use to transform the pages to get the required interactions and to work with data.

Now we are back to a potential hostile environment. The user has access to all client side code, and can use tools to debug and change the code, and to capture and alter data sent between the client and the server. An effect of this is that data validation on the client side could only be used to make a better and faster user experience, all input validation and access control has to be done on the server to, outside of what the user can control. Let us repeat that since it is so important: Validation on the client side is to give a better user experience, security validation is done on the server side!

The main difference between traditional client-server software and modern web-applications, is that in web-applications the user have access to the client side code and to the communication, and can change how the application is working and the content and format of the transmitted data. It is almost as easy to change an id from 1 to 2 in a POST or Ajax request as it is in a GET

request.

As normal, you should not trust any user generated data, always validate and sanitize it correctly, and remember that information sent from your code on the client side is under the users control, so you have to treat it as user generated code even if it is auto generated by events in the client side code.

Keep growing stronger to fight insecure software!

C.7 Handling passwords

It seems like securing the users passwords still is a problem in many applications, this might be because developers consider the database to be a secure storage. This is wrong, you should always assume that your database will be stolen. It can be stolen by vulnerabilities in your application, or in any other software running on the server, and it can be stolen by an insider.

A quick Google search for stolen passwords gives lots of results, and the top results are about big companies. Hacking a small company will not make the top news, so it is not strange that the big ones are at the top of the list. Companies that have fallen victim for password theft include Twitter, Yahoo, LinkedIn, and Evernote. So there is no reason for you to believe that you and your applications cannot be a victim. It is up to you if your application will be strong enough to keep the users passwords safe after the database is compromised.

Securing the passwords is a part of security in depth.

C.7.1 Accepting username and passwords from a GET request

The problem with this is not actually the storing of password, but that the username and password will be in clear text in logs and can be read by anyone between the client and the server. Server logs are also normally not secured, so this is worse than storing the password in clear text. When reading this you might think that this is not used anywhere, but you will be surprised by how many sites that do not separate input from GET and POST so they will accept username and password from GET requests without having planned for it.

If you want to test if sites accept logon through GET, remember that the password you are using will be stored in server logs, so change your password after trying. You are of course not using

the same password several places, so you do not need to be reminded to change the password before you test this.

Do not accept username and password as GET parameters!

C.7.2 Storing passwords as clear text

Users are lazy, and most users are not security aware and use the same username and password combination several places. This is the main reason for us to store the passwords in a safe way.

Even if it is not a problem that someone get control over your users accounts the account information can be used to log on to other sites. If you think you have a good reason to store the passwords as clear text, think again and see if the reason you have found really is valid. If it is to be able to share the password between applications you could share the encrypted version of the password, or set up single sign on using well defined standards.

Do not try to create your own single sign on solution. As a developer your job is to build software that solves some requirements as simple as possible, reinventing the wheel all the time is not making in as simple as possible. Not having good enough security is not solving the requirements.

Do not ever store passwords as clear text!

C.7.3 Storing passwords using a homemade hash

I did not first think of this as anything that needed to be included, but when searching for documentation and background to this post I did come across several places where homemade hashes was mentioned.

The answer to this one is never ever use homemade security algorithms. Security through obscurity does not work. Why would the algorithm you are creating be more secure than the ones created by experts in computer security and mathematics and that is peer reviewed by other

experts? The algorithms created by the best in the world is not even considered to be secure before they have been analyzed by other experts for years, and even then is there sometimes found problems with them. And as mentioned about single sign on, do not waste your time on trying to reinvent the wheel.

Do not ever rely on homemade security!

C.7.4 Storing passwords as a hash

Hashing algorithms are made to be as fast as possible create a checksum of an input text, hash algorithm. Even cryptographic hash functions are not created to do anything more than that, they are only made to be better at avoiding collisions. A collision is where several texts produce the same hash.

There is nothing in these descriptions that insinuates that a hash could be used to store text in a secure way, this misunderstanding might be based on hash being called "one way encryption".

The problem with using a hash is that hashing functions are made to be fast and always give the same result for a given text, which algorithm you are using should never be considered a secret. Using MD5 on the text "password" will always produce "5f4dcc3b5aa765d61d8327deb882cf99", and you can even do a Google search with the hash and find that the text is "password". The same can be done with SHA-1 and other hashing algorithms.

Using a proper password will of course make it a bit more complicated to reverse the hash, but not much. The easiest way to crack a simple hash is to use a rainbow table where hashes are pre-calculated so that most of the work is done earlier and in a simplified description, finding the password is only a lookup in a huge table.

Another way to reverse most common passwords is to use a dictionary attack, where the cracking software is using a dictionary containing words and the common ways of "misspelling" the words by using leet speak where characters are exhanged with numbers that looks like the character.

Do not store passwords as simple hashes!

C.7.5 Storing passwords as a hash with a fixed salt

A salt is by definition supposed to be "random", random is in quotes since it really is pseudorandom, and computers are not made to do anything random so they cannot produce anything real random in an easy way. So a fixed string that is the same in all instances is by definition not realy a salt.

When using a salt the salt is appended or prepended to the password before running the hashing algorithm on the resulting string. Since the salt is supposed to be random we cannot use a hardcoded string that should be used with all passwords, so 4 and nine are not random numbers.

Using a fixed salt will leave you open for rainbow table attacks, generating one rainbow table including your salt will give the value for all passwords.

The salt is like the hashing algorithm not something you should consider a secret.

Do not store passwords with a common salt!

C.7.6 Storing passwords using a normal hash with a unique salt

Now we are getting to parts that you might read as "best practice", but we are not there yet.

Storing the password as a hash of the password plus a unique salt is a good prevention against rainbow tables.

The salt is combined with the password in the same way as when using a fixed salt, but it is created unique for each password. The salt can then be stored as part of the stored password or in another column. Attacks against this are based on the fact that normal hashing algorithms is made to be fast, and CPUs and especially GPUs are getting fast, and users are not updating password complexity according to this speed. Check the hashing speeds at `http://golubev.com/gpuest.htm` and you will see that it will not take long to crack most passwords.

Avoid storing passwords as a normal hash with a unique salt.

C.7.7 Storing passwords using password storage functions

There are several algorithms created to securely store passwords, the most known ones are probably bcrypt, scrypt, and PBKDF2.

I recommend using bcrypt if it is available on your platform, but the others are not bad choices. The advantages with bcrypt includes that it is not easy to run on GPU, and you can change the cost it will take to produce a hash with it. It does of course require a salt, and stores the salt as part of the result.

See `http://openwall.info/wiki/john/GPU/bcrypt` if you want a technical description on the problems John the Ripper have with cracking bcrupt on GPU.

With bcrypt you are setting a cost parameter that is stored as part of the result. If the selected cost is too high so it slows server response time, it is possible to update the cost for each user on the next login. And the cost can be made higher when the speed of the hardware requires it.

All of these algorithms are good enough for a normal application so use any of the other if bryct is not implemented in your language, if you are creating something that requires extremely high security you will have to do some research on the different functions so you can justify the decision in the risk analysis.

Use an algorithm created specific for storing passwords!

C.7.8 Conclusion

Use the right tool for the job. Use a screwdriver, not a hammer, to screw in a screw, and use an algorithm designed for storing passwords, not a generic hash algorithm, to store a password.

C.8 Interesting attack

I am writing about different attacks and how to protect for them, and today I did find a new attack vector that I have not seen before. The attack is called copy-paste attack, it is also known as clipboard poisoning.

This attack is aimed at technical people that paste text into a shell that can execute commands. I recon you are somehow technical since you are reading this blog, so in this case you are the target and not your normal users.

The attack is based on hiding text in a command that you copy from a webpage and paste into a shell. The hidden text can then contain any shell command that the attacker want to executed in your shell, in the same way as you expect the shell to execute any command you paste into it. It is also possible to hide text in the terminal, so you can be a victim without noticing it.

The beauty of the attack is it simplicity combined with the possible consequences. Especially since the place you most often are using a shell is your server! The attacker can compromise the server without using any software vulnerabilities, it is done through the only person you really trust to do things correct; yourself.

Now imagine what can be done if you are logged in as root, or was convinced to use "su" because the commands you wanted to copy required root access. Now the attacker has managed to run commands as root without you knowing it!

If you want to try this for yourself you can copy "ls/dev/null;clear; echo "Hello world" -l" and paste it to a shell and see what happen. I am nice, so it will not harm your system, only show a message you have seen before.

Defense

The defense is as simple as the attack. Always paste copied text in notepad or any other simple text editor to check that you only did copy what you intended to.

C.9 Simple penetration testing

First a disclaimer, only use these techniques on an application or website if you have the explicit permission to do use them. Even though this is simple and seems harmless, it might be illegal and can be considered as an attempt at breaking into the application. In the same way as you are not checking if random doors are locked by trying to open them, you should not check if random web-applications are open by testing to see which input they have problems with.

Another disclaimer, if you expect to see the world as green letters falling down your screen, or if you expect to fly through a 3D folder structure you will be disappointed. Most likely you will not even meet Halle Berry outside sunbathing when you go for a break.

C.9.1 Experiment and play around

Penetration testing is more about thinking, "how can I break this" and then try it out. It is about finding what the application except as input, and then send it something else. Here I will try to give you some ideas about how to start with this.

We will start with the basics, without using any fancy tools to automate the job or to scan for vulnerabilities. This way we can first focus on removing the low hanging fruits. You should first secure against the trivial and simple attacks, before continuing with the more advanced.

When an input field asks you to type a number, you send in a text. Only to see what happens. You can do the same with other fields, for instance by sending in a text as value for a checkbox or date field. When typing in text you should always use ' (apostrophe) and " (quotation mark) to see if it break any SQL queries. We all think that apostrophe and quote are not a problem anymore, everyone knows how to escape strings before using them in SQL. But we are wrong, injection is still number one in OWASP top 10.

If you get an error after adding ' or " you have most likely found a SQL vulnerability. This post will not cover how to exploit that, for now it will be enough to know that you are vulnerable and should fix it. I have even seen websites that crash when I try to log on because I did have a ' in my password, they have probably also followed the worst solution for handling passwords.

You could play around with input fields like this if you want to, you could also use some tools to make the job easier. You can do a lot with normal developer tools like Live HTTP headers in Firefox. There are also similar tools for other browser, use the browser and tools you feel comfortable with.

Here I will use Live HTTP headers, it can be used to capture requests, change them and replay the request. The requests will be changed after any client-side validation, so this can is an example on why client-side validation is for usability, and server side validation is for security.

<Technical> I did install The OWASP WebGoat Project so I could get examples without attacking real applications. I did install version 5.4 in Ubuntu, there was some parts that was not described, for instance that you had to set up users manually in conf/tomcat-users.xml. According to the WebGoat documentation the standard installation only accept request from localhost, this was NOT the case when I used the war file. I did set it up on one PC and was able to connect from another, be sure to not allow traffic to the PC you install it on from Internet, and always stop tomcat when you are not using WebGoat. The application is highly insecure and opens up the computer for attacks, so never install it where it can be reached from outside of your local network. </Technical>

In the first lecture Http Basics, I did use HTTP live headers to capture the request.

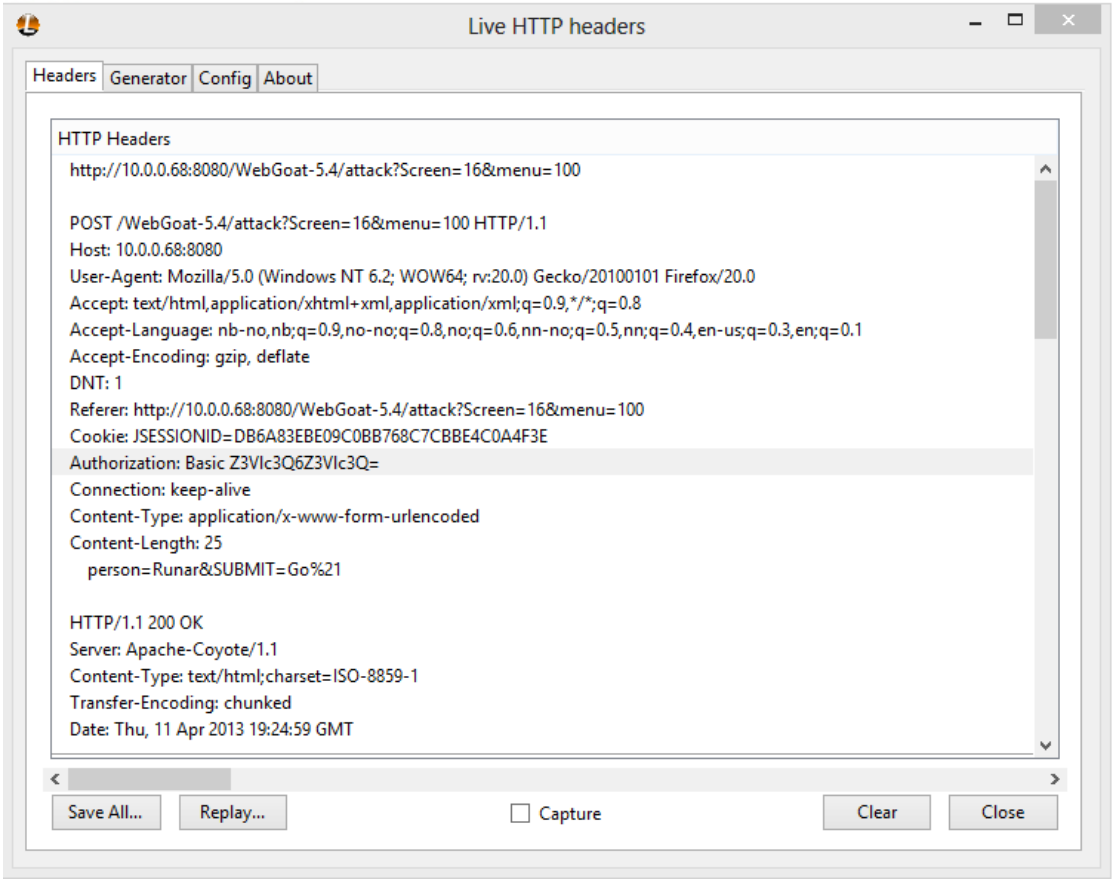

Figure 2: Capture

Here you can see all the details that are sent to the server. After capturing the request you are looking for you could remove the check in the checkbox Capture, and click the Replay... button. You will now get a new window where you can change everything and send them to the server again.

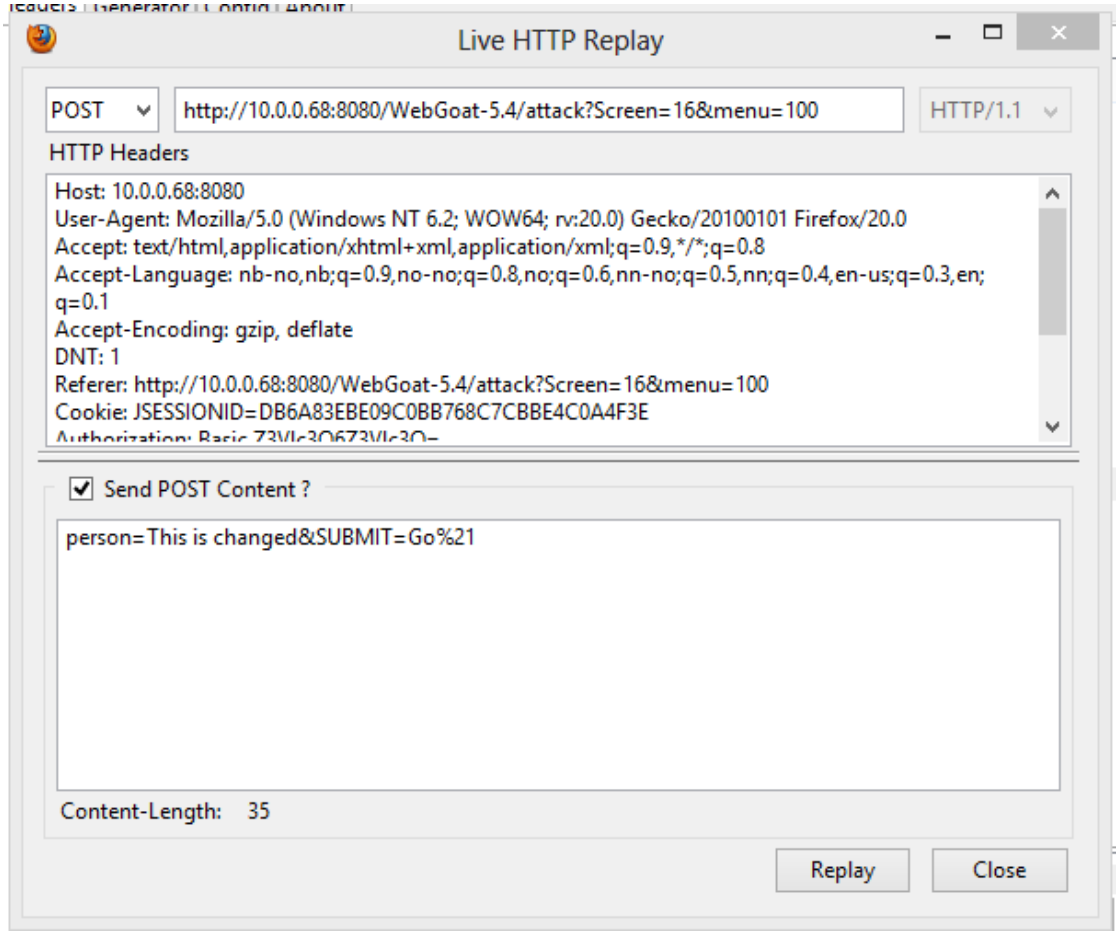

Figure 3: Replay

As you can see I did change the person= to "This is changed". If you use a proxy like WebScarab, you can change all this before it is sent to the server first time.

And as you can see from the result, the server did get the new text and returned it reversed as it is supposed to do in this example.

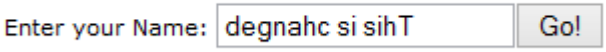

Figure 4: Result

Now you have gotten the basics on how to capture requests and change the data to what you want to send in. This way you can get around all client-side validation, and you can change to send a different data-type than expected.

You can also test your access control with this, when you are sending a request try to change ids to the ids for resources you should not have access to and see how the server respond.

Now you have gotten a basic introduction on how to capture and change requests before they are sent to the server, and have got a small insight in how it can be used in attacks. I have planned to soon follow up with more ways to use this to attack your application.

To get more training you could find and enter capture the flag competitions, and download and install The OWASP WebGoat Project. You could of course also test the applications you are creating, since that should be the main reason you are learning this.

Keep training and help make the web a safer place.

C.10 SQL injection

This is the first post about different common vulnerabilities on web sites. Description about other vulnerabilities will follow later. My intention was to have one post with some short details about each type of the most common vulnerabilities, but I did get a bit carried away when I started to write.

SQL injection is one of the most common ways to hack into applications, and is also one that is fairly easy to avoid. The attack is based on breaking out off a string and insert SQL statements. This works because many developers build dynamic queries that include input from the user without validating the input. One simple example is a bad code for login validation, this is only an example you should store the password as a normal hash:

public function doLogin($username, $password){ $result = mysql_query("SELECT firstname, lastname from users where username = '". $username."' and password = SHA1('".$password."'));

/* Code for getting the result and continue the login process. */

}

This can of course be done in any other language and database. One attack to get access with a known username, test, might be done by sending in the username:

test';– '

this will produce the query:

SELECT firstname, lastname from users where username = 'test';– '' and password = SHA1('')

As you can see only the username is checked and the rest is commented out from the query. If you do not know a username you can of course still get in, for instance by using the username: '

or 1=1 limit 1;– ' the limit 1 part is only in case there is a test for how many rows are returned, that part have to be changed based on the database used. In Oracle you can use **rownum = 1**. You can also use it to find different users by changing the rownum or where you start with the limit.

As you can see the username was ended after ' and the query was ended by ; the rest is commented out. So here the attacker was able to log in without knowing the password. This kind of attack can also be used to get any information from the database.

First a small sidestep to why you should not show detailed error messages to users. Considering having a query to get names and description for shops based on city name. If the query is vulnerable, the user can type in Test' and if the server is set to show the detailed error message the user might see something like.

You have an error in your SQL close to name like '%Test'%'; The query was SELECT id, name, description FROM shops where name like '%Test'%';

Now the user knows the columns that is included and that the query is vulnerable. Not showing detailed error messages is not considered security by obscurity, it is only to not the attackers. The attacker can now for instance try the input **' UNION SELECT id, username as name, password AS description FROM users;–** Now we can only hope you have stored the passwords correctly so the attacker do not get all passwords from your database.

The best way to avoid this is to use parameterized statements, often the recommendation is prepared statements. However it is not the prepared statement part that helps, since it still is possible to dynamically build the query with a prepared statement. But when using bind-parameters the database would know which parts are parameters and know how to handle them. So the correct will be to build the query as
SELECT firstname, lastname from users where username = :username and password = SHA1(:password)
or use **?** instead of the named parameters, and then bind the parameters in the correct way for your language.

If it is not at all possible to get parameterized statements in your environment, you should always escape all input that is not created by your code. See OWASP for more about this.

OWASP do also recommend using stored procedures, however I think it makes the code difficult to read when you have business logic in your database, and code that is difficult to read will soon start to smell and be a place where you will have bugs. See Security vs. quality for a bit more about this.

You should always secure the input to your queries, not only if you get it directly from a user, but also if you get if from a file, a web-service or any other source. The source do not know how you need to have the data secured.

It is a shame that it still is possible to find tutorials and books that builds dynamically queries where input is concatenated into the query. Don't make your application vulnerable to SQL injection, as shown here it do not require extra work.

C.11 Security through obscurity and hiding implementation details

I did read a short article by Brian Honan in (IN)SECURE Magazine issue 37 where he recommended to always ask "why" in the same way a 4 year old will. I might have stopped to ask why, but I have not stopped to think "why". I encourage you to also keep the curiosity and ask "why", it will help you in finding how things work and the real reason behind decisions and recommendations. I am trying to include some answers to why in my posts, and from now on I will be more aware of it while writing.

Security by obscurity is believing that something is more secure because the implementation is a secret, while hiding implementation details is, well, not showing how something is implemented. These two concepts might at first glance seem as two names for the same thing, but they are not.

Security through obscurity is to create your own encryption solution because others know how the standard solutions work, think that your JavaScript is incomprehensible because it is minified, or that no one will understand your configuration is in the file notConfiguration.xml. You can consider it as hiding your front door key under the welcome mat, everyone that knows where to look will find it.

Figure 5: Key under mat

You should expect that the bad guys know everything about your implementation, but you should not feed them with this information. Remember that this is a competition, so do not lose by giving away all your moves before the fight have started, instead have them fight to get information.

You will get more security by using best practice and solutions that have been tested by experts for a long time, than by creating your own solution that only is proven to not be broken by you

or any of your co-workers.

If you create your own encryption solution and have to keep the code a secret then you have security be obscurity, if you do not show your table- and column names of your database you are hiding the implementation details. On the other side calling a table "table1" instead of "users" is back to security through obscurity, and will only add to the complexity of the program and lead to bugs.

You should not include implementation details in error messages sent to user or to the client so the user can find them. Instead you could log the error including a number, string or GUID that also is given to the user, so you can use it to track the logged error. If you have SQL injection vulnerabilities and show detailed errors to the user, the user will not only see that the application is vulnerable, but will also be able to use the error message to easier exploit the vulnerability.

Another aspect is that a directed attack is often starting by using public know information, this include looking through your web-sites, searching for your product and company using Google, or any other search engine, and looking up any other public available information. If it is done by someone local dumpster diving might also be involved. All information can be valuable for the attacker, but to use with software attack and for social engineering.

The company policy often does, and should always, contain who can publish information about the company on the web-pages, and which information they can post. It should also include the information the applications should be able to give away, this should be based on a risk assessment, and not by guesses from the development team. Security is owed by top managed, and they should set the constraints.

Keep practicing and make it difficult for the bad guys to win!

Figure 6: Workout

C.12 Webification

Webification is defined as the act of process change to take information and convert it into viewable information on the internet. I prefer to take a broader definition and use it for all processes that are moved to the internet. It can be everything from applications that used to be on the relative security from operating on a local network or on a local machine to devices and machines that previous have not been thought of as something that will be connected to a network.

The consequence of this is that developer organizations creating software and firmware for this have to adapt to a new reality with a much larger attack surface than they are used to. Now everyone with an Internet connection can have a go at hacking the code they are writing.

Instead of trying to stop this we have to adapt, welcome any newcomers and help them see the pitfalls. There is no reason for why they should relearn the best-practices we have learned programming for the web. Internet is a much more hostile place for the code now than it was years ago when the web was sparkling new.

We must also admit that it is cool with this rapid pace of moving more software to the cloud. Right now I can to music on Spotify, copy the draft for this from Google drive and post it to Blogger after doing the final editing, all using Wi-Fi from a hotel room in Sweden. You are probably rich from your software inventions if you did see this possibility coming 10-15 years ago.

C.12.1 The Internet of things

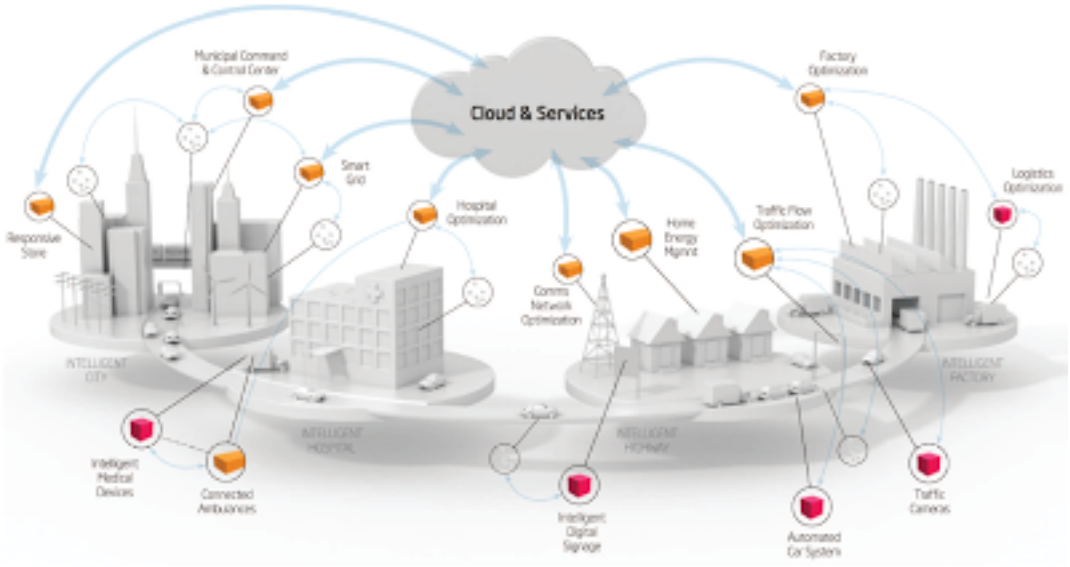

Figure 7: The Internet of things

Some years ago there, around year 2000, it was a big hype about the Internet of things and how everything will be connected to the Internet. It did maybe not get the initial traction as promised, but now we are starting to get there, without all the fuss about it. Mobile phones have been Internet enabled for years and many are now closer to computers that also can be used for making phone calls, blue-ray players and TVs are connected to Internet and are being labeled "smart". Even cars are starting to have so advanced control system that they can be hacked.

C.12.2 A short technical part

I did plan to not get technical in this post, but have to add a bit or two.

Because of limitations in IPv4 most private users have one IP address and uses NAT to connect several devices to Internet. A consequence of this most things are not directly open for incoming traffic, and avoid several types of attacks. Sooner or later IPv6 will be more normal, and instead of having one address each user will have thousands of addresses. When this happen we will have to start getting everyone to use firewalls, or accept that our software can and will be attacked. Not all software will be targeted for direct attack, but all will sooner or later come under automated scripted attacks.

C.12.3 Moving software to the Internet

Statistics from several sources shows that the overall security in web-applications has risen the last years. We must make sure that this trend is continuing, and accelerating.

One concern is all the reports of apps for phones that have the most basic security vulnerabilities, like not using https and sending sensitive information as part of the GET string.

Figure 8: Credit Card information in GET string

This example is from an app from Scandinavian Airlines, where credit card information is sent as part of the GET string. This is taken from this security report. The app was quickly updated fixing this.

This is only one example and should not in any way be seen as a way of saying that they have worse security than other apps. As we know doing security reviews, both internal and from external parties, is necessary for improving the security. Having the findings public is in my opinion a positive action and help showing that security is a priority, even when simple mistakes like this is done.

Another research finds that 1 in 13 Android app have "inadequate use of SSL/TLS". This do not only affect Android, apps on iPhone do also have the same issues. I can go on and on with more examples, but I think you get the drift.

One reason for this might be that developers think it is more difficult to listen in on traffic from phones, or it might be because of the popularity of apps where many businesses see it as important to have an app and the simplicity with creating apps. Not to be able to offer more to users, but because it is "cool" to have an app. However it is not any more difficult to listen in on Wi-Fi traffic from a phone than from a PC. The reason can also be that it is too easy to create an app, like for web-application it is possible to create apps without extended knowledge of programming, and no knowledge of security is required.

C.12.4 What can we do?

The difficult is what takes a little time the impossible is what takes a little longer. - Fridtjof Nansen.

The challenges with securing new players on the internet are not only limited to phone applications, it includes everything that is connected to Internet. It can include your new fancy refrigerator or car. Imagine getting up in the morning and finding that your milk is sour because your refrigerator was hacked and turned off, or having a car that use Wi-Fi instead of wiring to save weight, and an attacker in the car next to you manages to disable your brakes.

This might sound like science fiction, but it is not. Cars do use Wi-Fi, refrigerators, water treatment plants and other SCADA systems are connected to the Internet and are often not well secured or are set up with a default password.

More important than finding the reason is to spread information about why and how to secure software from network attacks. The required level of security will differ from system to system, you do not expect your refrigerator to have the same security level as the system sending your medical prescriptions from your doctor to the pharmacy.

In short all software connected to Internet should at least have the same level and type of protection as normal web-applications. You do expect that script kiddies can hack into the camera on your web enabled TV so they can watch you without you knowing. The extended version is that a risk assessment is required to identify additional attack vectors and the desired security level.

To create more secure applications you have to learn about security and apply what you learn. When coding you should always thing; how can I break this, and in order to see how you can break it you have to know something about how to break code.

Buckle up and get ready to fight off attackers!

C.13 Cross site scripting (XSS)

Cross site scripting, or XSS, is a nasty attack that it is not straightforward to defend against. Using XSS an attacker can attack other users of the web-application through vulnerabilities in the application. All sites that include some user generated content must defend against XSS, even something as simple as showing a search parameter from the user can be used in an attack.

C.13.1 Difficult to defend against

It is difficult to defend against XSS, or actually it is not a problem if you do not let users input any information that other users can see. But this sets us back to the mid-90s static web-sites, and is probably not what you want to create. Since you are creating modern web-applications with user interactions you will have to defend against XSS to secure your users. It can be difficult

to get 100% coverage and be totally secure against XSS.

In the same way as you do your best to avoid bugs you should also do your best to avoid XSS vulnerabilities and in many cases bugs can be exploited as XSS vulnerabilities.

C.13.2 Attack vectors

The simple example of XSS is to use input of the type: "alert('vulnerable')" or <script>alert('vulnerable')</script>. It is easy when seeing this to think: "Why is this a problem? Letting users create an alert in their own browser is not an attack". That is correct; showing an alert the user created is not an attack, it only shows that the application is vulnerable. If an attacker first has found a XSS vulnerability he will have many ways to exploit it, based on what he want to gain from the attack.

The attack can be innocent, for instance where one user do a search including some JavaScript to show a message and sends the search as a link to another user, so the other user sees the alert from the first.

It can also be much more serious where they payload collects data and cookies from the application and sends to the attacker, or includes a hidden iframe from another site that tries to break into or infect the computer using vulnerable add-ons.

C.13.3 Defending from XSS

As previously stated it is not trivial to defend against XSS when users can provide content, so we will have to know several defenses. First we should start with setting some headers, to tell the browser that we know what we are doing so it should not guess what we try to do.

Headers

X-Content-Type-options: nosniff This header prevents Internet Explorer and Google Chrome from MIME-sniffing, so that they do not question and try to correct MIME type set from the server.

X-XSS-Protection: 1; mode=block This enables the XSS filter most browsers have built in if it is disabled. Default value for most browsers is to have it on, we want it on even if users or rouge applications on the user's computer have turned it off.

X-Frame-Options: deny/sameorigin/allow-from: URL Use any of this three options based on your requirements, as long as you do not expect to be in an unknown iframe.

X-Content-Security-Policy: This can take an extended policy that you can have as precise as you want. You can for instance disable inline JavaScript, a side-effect of this is that you are forced to have cleaner HTML with all JavaScript code in the head section or linked into the head section. You can find the complete description at W3C.

Coding

It is quite straightforward to avoid XSS attacks if you only expect unformatted text and only showing it inside the HTML markup. Then the only thing you have to do is HTML-encode all text before used. If you are going to use it as part of a link you will have to use URL encoding on it, and so on.

If you are allowing any HTML to format the code, insert images, etc. you have more work to do. You will have to validate the data through a whitelist to only allow the tags and attributes you have defined.

Blacklist vs. whitelist

Some developers try to create a blacklist to say what is not allowed. Do not do this, there are too many different scenarios you have to cover. Even if you manage to include all know ways to include script you will probably not have covered all that is found later. You should always use a whitelist, and add more to the list when a valid request for it is made and it do not compromise the security.

Using whitelist

When whitelisting the data to show you have a list with what is allowed. When dealing with HTML this means that you will use the list to set which tags are allowed and if they can have attributes. If input that is not allowed is detected the best is discharge the whole text or html-escape that part. If you are trying to remove the invalid parts you will have to check the text again after it is removed, in case an attacker have used input like "«SCRIPT>alert("XSS");//«/SCRIPT>" or "<scr<script>ipt>alert(1)</scr</script>ipt>".

Encoding

Encoding is not only a way to defend against XSS attacks, it is also a way to try to get around defenses. The input can be encoded in ways the browser automatically decodes. One possible attack is to use UTF-7 encoded input when the rest of the site is UTF-8, if correct charset is not set in the header and if no-sniff is not set the browser might try to correct this and convert the charset and changing input the code did not see as an illegal text, into an attack. The string "+ADw-script+AD4-alert(1)+ADw-/script+AD4-" in UTF-7 is the same as <script>alert(1)</script> in UTF-8.

Use library

Do not reinvent the wheel and let the experts help, see you can find a library you can use to validate the input texts. OWASP recommends:

- For Java `https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project`

- PHP http://htmlpurifier.org/
- JavaScript/Node.JS https://github.com/ecto/bleach
- Python https://pypi.python.org/pypi/bleach

What and where to validate and escape

The best place to escape the texts is right before output, because then you will not have to remember if you already have escaped it. This will also make it easier

C.13.4 Summary

Do not trust user generated input and do not trust input from external sources. Escape all output using a whitelist. When dealing with XSS there is no such thing as being too paranoid.

C.13.5 Resources

https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet

Bibliography

[1] No silver bullet. http://www.cs.nott.ac.uk/~cah/G51ISS/Documents/NoSilverBullet.html.

[2] 2000. First computer bug. http://www.jamesshuggins.com/h/tek1/first_computer_bug.htm.

[3] Timeline of computer viruses and worms. http://en.wikipedia.org/wiki/Timeline_of_computer_viruses_and_worms.

[4] Internet versus world wide web. http://computer.howstuffworks.com/internet/basics/internet-versus-world-wide-web.htm.

[5] William Bradley Glison, L. Milton Glison, R. W. 2007. Secure web application development and global regulation. *Second International Conference on Availability, Reliability and Security*.

[6] Owasp risk rating methodology. https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology.

[7] Usher, R. W. 2002. Improving software security during development. *SANS Institute Reading Room*.

[8] Janzen, D. S. 2005. Software architecture improvement through test-driven development. *Proceeding OOPSLA '05 Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*.

[9] Whitehat website security statistics report. https://www.whitehatsec.com/assets/WPstatsReport_052013.pdf.

[10] 2013. Rypskar on security. www.rypskaronsecurity.com.

[11] Owasp. https://www.owasp.org/index.php/Main_Page.

[12] 2012. Mozilla secure coding guidelines. https://wiki.mozilla.org/WebAppSec/Secure_Coding_Guidelines.

[13] 2011. Microsoft security development lifecycle. http://www.microsoft.com/security/sdl/default.aspx. [Online; accessed 2011-12-06].

[14] Sans institute. http://www.sans.org/.

[15] 2013. Owasp top 10. https://www.owasp.org/index.php/Top_10_2013-Top_10.

[16] 2013. Rypskar on security - reasons to focus on security. http://www.rypskaronsecurity.com/2013/03/reasons-to-focus-on-security.html.

[17] 2013. Rypskar on security - web application security. http://www.rypskaronsecurity.com/2013/03/web-application-security.html.

[18] 2013. Rypskar on security - webification. http://www.rypskaronsecurity.com/2013/08/webification.html.

[19] Hakan Erdogmus, Maurizio Morisio, M. T. 2005. On the effectiveness of the test-first approach to programming. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 226–237.

[20] Cobit 5. http://www.isaca.org/cobit/pages/default.aspx?cid=1001118&Appeal=Google&gclid=CMWq9pHtk7oCFa93cAod7VYAsA.

[21] Val it. http://www.isaca.org/Knowledge-Center/Val-IT-IT-Value-Delivery-/Pages/Val-IT1.aspx.

[22] Risk it. http://www.isaca.org/Knowledge-Center/Risk-IT-IT-Risk-Management/Pages/Risk-IT1.aspx?utm_source=multiple&utm_medium=multiple&utm_content=friendly&utm_campaign=riskit.

[23] Cenzic security education series. https://info.cenzic.com/getting-better-watch-lessons.html#learn?mkt_tok=3RkMMJWWfF9wsRonvqXIZKXonjHpfsX57e4sW6GylMI%2F0ER3fOvrPUfGjI4ATMVkI%2BSLDwEYGJlv6SgFS7DAMax5z7gKWBA%3D.

[24] Introduction to secure software development life cycle. http://resources.infosecinstitute.com/intro-secure-software-development-life-cycle/.

[25] Glass, R. 2002. *Facts and Fallacies of Software Engineering*. Addison-Wesley Professional.