Usability and Verifiability of Secure Features for Authenticating Identity

Teemu Partanen

University of Tampere
Faculty of Natural Sciences
Degree Programme in Computer Sciences
Software Development
Master's Thesis
Supervisor: Eleni Berki, Sunil Chaudhary
May 2018

UNIVERSITY OF TAMPERE
Faculty of Natural Sciences
Degree Programme in Computer Sciences
Software Development
TEEMU PARTANEN: Usability and Verifiability of Secure Features for Authenticating Identity
Master's Thesis, 49 pages
May 2018

Almost all financial transactions and personal data is nowadays online. A world with easy access to data and finances simplifies everyday life. Matters can be handled at ease where ever there is an internet connection. Contacting others can be done in ways unimaginable a decade or two ago. Instant messaging apps and video meetings bring the whole world close when working. If an end user finds something hard to handle they start sabotaging it with their personal behavior. They use less secure methods to keep their data secure because it is more convenient.

The world of software security is a balancing act between designing features secure enough and being able to verify the functionality of secure features against malicious attackers and making secure features usable. Usability improves the chances that the end user complies to use of every day security. Designing features secure enough to fight against malicious attackers has gained too large proportion of the effort. According to literature reviewed in this thesis usability of the secure features has not been seen as a priority.

This thesis examines usability and verifiability of secure features and methods. It is important to study the usability in this context, as better usability will allow secure features to appeal to a larger end user base, and adding the overall security. It will go through typical authentication methods and assesses their usability based on literature about usability and every day observations. It follows a high-level approach to secure features to be able to see what an end user encounters when using secure features. This is done to better evaluate the usability of the features. Especially when specifications are not fully available.

The thesis also introduces a formal testing process structure that can be used as a guideline in planning and executing tests for any software feature. Helpful toolsets to aid in creating functional test environments and support functions are presented. The thesis introduces different kinds of existing and future method that will make security and usability of the authentication
methods better.

Keywords and terms: authentication methods, security, usability, testing, test automation

Acknowledgement

The subject studied was a complex combination of subjects that were partially known, well known or not known at all by the author. During the times when the writing process was really hard my good supervisors showed the way to move forward with giving constructive feedback. A thank you is in order for Eleni for providing a base knowledge set needed to get this going. Sunil's feedback steered this thesis to the right direction just when needed. At the late stage I also received valuable pointers from Juri Valtanen. The challenging questions related to my work made me think the subject in hand more critically.

Table of Contents

1. Introduction

With this thesis, the author studies literature in the international and national community about the usability of secure features. The current state of usability of the secure features is illuminated along with biggest challenges of it. Furthermore, improvement suggestions for the situation are presented. In addition to that the author introduces essential processes and tools for testing secure features. This thesis studies usability and verification of secure features and methods.

What has been often notified is that the usability of secure features is not an interest for people who implement them [Dhillon et al. 2009] The goals of secure feature design are set on what kind of security level in product should achieve. Some common interfaces towards the end user are implemented without a thought of how usable they are. The focus is on the implementation of the parts of the feature that are invisible to the end user. What this results in is that people refuse to use the full protection offered by secure features. Improving usability will result in more secure end user devices. According to the studies referred in this thesis this is due to the fact that the end users tend not to use obstructive secure methods if they can avoid it.

The secure features and methods studied in this thesis are authentication methods. They are universal and they have also been affected a lot by a switch towards mobile handheld devices. The usability of the methods have stayed relatively unchanged even though the environment where they are used is totally different. Personal Identification (PIN) code for example was once entered to an Automatic Teller Machine (ATM) machine or at the privacy of one's own home with a computer. Now mobile carry everywhere devices are used in the middle of the rush hour traffic in the public transport. Tabletop devices simply cannot be used without being observed by others in the same space.

1.1 Research Questions and Methods

This thesis answers on following research questions

1. *What are the aspects of the good usability?*
   Secure features and authentication methods part examines literature about usability to form a view that is considered to be a good usability. This subject is further examined throughout the thesis. Usability may also contain subjective aspects that cannot be assessed here.

2. *How to approach to a new feature that is supposed to be tested?* How a new feature is approached from a testing perspective? What are the tools that are being used? What is test automation? These are covered in the vefification of secure features part.
3. *How to improve usability and security of secure features?* This is linked to answers to question 2. To answer one of the questions will also partially answer the another.

There is a need for thesis to clarify the process of getting to know secure feature in a process where the feature is seen for the first time and no specification is available, and verifying the functionality of it. This is one of the key points of the thesis and I need to enforce the tester mindset all the way through the thesis. Another thing is to take a critical look into one essential part of the quality assurance that is the usability of secure features.

Research methods chosen were to do a combination of research thesis and a hands-on user guide like approach for the reader. Literature review was done to research the subject further and to understand the phenomena. Furthermore, it was performed to get a clear view of the current state of the research done in the field of usability in secure features.

The hands-on part consists of designing, creating and up keeping the test process with tools needed for verifying secure features. That part of the thesis could be utilized for creating a testing environment for any given target feature. Verification of the features is also an important part of ensuring the usability.

Literature review examines studies and theses on security and usability point of views. The first main selector for selecting literature for this review has been an attempt to find literature about the key aspects of the thesis. Search words included usability, secure features and software testing. Then the sources were further selected based on the availability of the sources. Main sources of literature selected were acquired through the University of Tampere electronic library services. Materials regarding security e.g. Loberg [2004], Nalam [2015] were easy to find. Furthermore, usability as such is widely covered. The methods covered in the scope of this thesis, the authentication methods, were easy to find. During initial literature search it was noticed that the combination of two or more of these subjects was hard to find. The literature found from the combination of the issues suggested the same in their own studies. Main database for literature used in this thesis was ACM Digital Library. When searching ACM database with usable security search words 244 different hits were produced. Usability and authentication provided 14221 hits. Some key literature pieces were also received from thesis supervisors.

In the process of literature review itself sources were first evaluated for their usability for the scope of this thesis. Then not so relevant sources were quickly browsed through searching for possible information. The relevant sources were inspected more thoroughly. Key books in the usability and security field were thoroughly read through as they served two purposes. They were used for this thesis and also in other studies. Notes about the key issues were written down during the review process and for some parts revisited after reading.

Time used for the literature review cannot be assessed easily since the review was done in iterations. The first initial search round took a few days. More and more comprehensive checks of the literature were then taken during the whole thesis process as the subject matter became more familiar after each step.

1.2 Summary of Key Concepts

Authentication as a term in computer science describes a process where the end users establish their identity to a system. Methods used include use of a password or possession of a physical device, e.g. an OTP (One Time Password) generator. In case of a reverse authentication, the system authenticates itself to the end user, e.g. digital certificates in web services [Butterfield and Ngongi. 2016]. Authentication in the scope of this thesis refers to authentication of the personal identity. It refers to give the proof that the service user is the one that is supposed to use it.

Authentication methods are used to prove that a user is eligible to use the service. There are different types of authentication methods. They may require manual interaction with service's graphical user interface. Methods in the scope of this thesis are services that require this kind of interaction. They benefit from the best possible usability and are therefore relevant to the scope of this thesis. The authentication methods also provide an easy way to start learning testing of unknown methods.

Identity refers to proofing the identification of the user. Identifying is giving a proof that the user is allowed to access the service. Identity theft happens when someone steals personal information to use it to commit a crime, usually a fraud. Using a stolen identity, it is possible to open up bank accounts, get loans and even propose to be a different person when getting caught committing a crime [Loberg. 2004]. It is vital that our personal information is kept as safe as possible. Introducing more usable secure features can introduce them to a wider user base.

Security refers to protection against unauthorized access to information or protection against intentional but unauthorized removal or alteration of that information. Security acts against both unintentional as well as deliberate attempts to access sensitive information [Butterfield and Ngongi. 2016].

Usability refers to design user interfaces in attempt to make them intuitive and easy to use [Butterfield and Ngongi. 2016]. Usability in general applies to all products that are meant to be used by humans.

Verification is verifying accuracy of the information [Butterfield and Ngongi. 2016]. In this thesis, the information verified is the intended functionality of the examined secure features against their specifications.

1.3 Thesis Outline

This thesis studies usability and verifiability of authentication methods. Verifiability differs from verification that in verifiability it is unknown at the start what is the level of verification can be done to the feature. The thesis takes a high-level approach to the secure features to be able to see what the end users see when they are using the features. The high-level as opposite to low-level is the application layer that the end users see when doing everyday operations with operating system's user interface. The usability is chosen as a focus because secure features are used on every day basis by regular people.

Chapter 2 examines secure features and authentication methods. The theoretical phase of authentication process is at first described. Then different authentication methods are separately examined and usability of the secure features is evaluated using available methods.

Chapter 3 contains a literature review of usability and secure features. It also includes recommendations and future improvement aspects of improving both end user experience and security of authentication methods.

In Chapter 4 these authentication methods are studied from a formal testing process perspective and testing aspects are explained to the reader. The process is split to logical chunks. The testing aspect is opened up more by examining the test automation and the virtualization toolsets. Negative cases where the security is tried to be breached are addressed. Since trying out all possible outcomes is really labor intensive tools to automate the security breaches are introduced.

Chapter 5 of the thesis is for evaluating thesis content validity against other available research material. It evaluates the contribution of this thesis to its research field against other research in the field and gives recommendations for future studies for this area. Personal insights of the thesis process and the field of study area are also described.

2. Secure Features and Authentication Methods

In this chapter, different types of authentication techniques used for secure identification are studied closer. Authentication is used to secure personal data and finances. Authentication methods analyzed here have been selected to be ones that need the end user interaction with the secure service. Interactive functions of this type are interesting in the usability point of view.

2.1 Secure Features

Software security is an idea implemented to protect software against malicious attack and other hacker risks so that the software continues to function correctly under the potential risk.

Secure feature is usually implemented so that it only accepts the intended inputs and depending on the input it results in an outcome. It is protected against unintended inputs. E.g. a PIN code entering interface only accepts numbers and a limited amount of numbers to reduce the risk of unintended outcome.

2.2 Authentication and Authorization of Identity

Authentication to system can be split to three stages [Zylkarnain et al. 2013]. Identification is the part of the authentication process where an identity is provided to the system regardless of the result of the authentication. In next phase the system assesses the authentication by given password and if they are proven correct grants authorization to system. Some of the methods described here have all three stages in use. Some only ask for authentication and then either authorize the user to use the system or denials authorization. In the last stage the access to the system is either granted or denied.

Existing authentication methods involve three basic factors [FFIEC. 2018]:

- Something the end user knows, e.g. password or Personal Identification Number code.
- Something the end user has, e.g. One Time Password tokens, password lists etc.
- Something the end user is, e.g. biometric characters such as fingerprint

Depending on the level of authentication rounds authentication is either a standard authentication (one foctor) or two-stage authentication (two-factor). In the first one a single authentication input is required and in the latter there are two different authentication methods combined.

2.3 Authentication Methods

Authentication methods are covered here starting from mobile authentication methods and then for reference some other usable features are covered.

2.3.1 PIN code

A PIN (Personal Identification Number) code has been used as a verifier of the owner since introduction of credit card. ISO-9654—1 standard [ISO standards. 2017] defines a PIN code to be a 4 to 12-digit code that is used as a password to identify the owner of the credit card or a service.

In automatic teller machines (ATM) and internet banking services PIN code gives user the right to withdraw money or complete financial transactions. Code is usually four digits in length. In mobile phones PIN code gives access to use the SIM card. The length of SIM PIN code is restricted between four and eight digits. The additional security definition for SIM PIN code is that the end user can only try to enter wrong SIM PIN code three times and then the SIM card is locked.

PIN is widely known, understood and used method. The relative safety of it is more linked towards user behavior of not showing it to others. PIN code may refer to the SIM PIN code that is used to unlock the SIM card, or it may refer to the PIN code that is used to unlock the screen.

When addressing the usability of the PIN code inserting an Android OS mobile phone is used as an example. How can the PIN code be changed by the end user?

The end user needs to find a wheel symbol somewhere in the device menu. Alternative method is to locate the wheel symbol from the drop-down menu by dragging it down from the upper part of the screen. The Android interface is designed to give the end user many ways of entering the settings. The wheel is now located and can be pressed.

Press wheel -> Settings opens -> locate and press “Security”, it may or may not have a lock icon also -> Security menu opens -> Sim Card Lock Settings -> Change SIM PIN. When you get to this part of the menu, the existing SIM PIN code may be required when getting to the change menu.

Can that procedure be considered easy and usable? How easily reproducible that method is when it is needed next time six months from now? Questions about the usability of settings in Android have been raised.

How does one usually find the menu where this change is done if haven’t read this thesis in the last 30 minutes? Usually it is googled. “Changing SIM PIN code android” gives about four million answers right away. This is not a scientific

method. But judging from experience if there are four million hits instantly found for the problem, then it is shared by many people.

2.3.2 Password

Using password is a similar method to the PIN code for unlocking the device. A password is a set of characters that the end user inputs when doing authentication. The main difference is that it may include also alphabets and special characters.

The password gives way more variability in settings. Password can be set to certain lengths e.g. 4-16 characters. The passwords can be simple where all given characters are accepted. An example of an easy password is a word that means something. When a complexity requirement is added it usually means that the password needs to include alphabets combined with number(s) and special character(s).

Ease of use need to be considered when an end user is restricted to give a complex password. It is more demanding to insert a complex password than just to type few numbers like the PIN code.

Changing and setting the password can be as hard as changing the PIN code. In addition, depending on the settings there can be a need to set a complex password. The task of typing and changing constantly complex password can be hard to type using computer, let alone using a mobile device touch screen virtual keyboard.

2.3.3 Password Lists

A subcategory of authentication PIN code and password is password list that is used by banking sites as a 2nd factor authentication method. It is usually a physical document handed out to a specific user.

The end user is asked to authenticate him/herself with a named password from the list, this list is only available for that particular user. Some applications can also send the password request to the mobile phone requesting the end user to give it to online application.

2.3.4 Fingerprint

The fingerprint is a unique identifier used by mobile devices today. Fingerprints as such are secure [Jo et al. 2016].

They do create another possible security issue if the personal fingerprint identifiers are exported outside of the end user's mobile device for some malicious purpose.

Using fingerprint to log in to the devices requires trust towards the software provider. This kind of unique identifier could easily be used for tracking the life outside the proposed scope.

Although fingerprint is a good and usable authentication method for the most part it is not without issues. Users with wet hands usually have issues with fingerprint scanners. In Finland using fingerprint scanner after sauna does not usually work due to skin wrinkles. It is also not usable method for people with certain types of handicap or a profession that is hard on the skin of the fingertip.

Fingerprint scanners work usually quite poorly in these circumstances. If there is only one factor authentication with alternative password this might not be an issue. But mobile systems these days may require a two-factor authentication and an issue may result in not being able to access the device or even wiping the data inside the device. Severity of the unsuccessful authentication depends on the device settings and in corporate world they usually are not configurable by the end user.

2.3.5 Facial Recognition

Facial recognition authenticates the end user with unique facial features.

Technical implementation of the feature varies a lot. Traditional methods take a picture of the end user and compare pictures. Apple has announced an infrared camera module that actually measures the end users facial structure and uses it for recognition [Forbes. 2017]. In the method where picture is taken the system could be fooled by using picture of the owner. The traditional methods of facial recognition are based on the implementation of taking the correct picture and then when doing a facial recognition, a mathematical basic comparison is done to analyze the percentage difference between pictures. That presents a problem. Since only the two dimensions are taken into account there is a change that some systems can be fooled with showing the picture of the device owner.

The usability of the feature is acceptable since just a glance of the face should be enough to identify the end user. Putting on a hat, glasses or growing a beard is really problematic with traditional facial recognition in addition of security issues.

The new 3D scan introduced with iOS 11 intends to prevent these issues described when using the iPhone X [Forbes. 2017]. Apple Face ID is replacing TouchID in iPhoneX. Enablers of Face ID are the TrueDepth came and new powerful processor of the product. System operates by measuring and analyzing over 30 000 invisible dots and creates a depth map of users face instead of a picture. System is designed to open the device when the end user takes a look into it. This facial map is only stored into the device and not in any cloud based

solution. Face ID is supposed to solve issues with reliability and also with glasses and hats that have plagued the traditional systems. Theoretically if the system works it should recognize the end user's facial features even if a disguise is used. There is a limited real life experiences for this technology.

The facial recognition there is a trust issue between the end user and the software provider. If both fingerprint and the facial recognition are done adequately to give the user a unique identifier, that identifier could be used to identify them also in the other places. Keeping this data safe is really important.

2.3.6 Two-Factor Authentication

Two-Factor authentication is a two-stage authentication method. One implementation of two-factor is password list presented earlier. Theoretically it can be a combination of any of methods presented earlier.

There are some proprietary solutions for two-factor authentication. One of the most notable solution is implemented by Apple [How does Apple's new face id work. 2017]. Unlike the SMS based methods, a six-number code is sent to one of the end user's other Apple device that is linked to the Apple ID account. A pop up then opens requesting that is signing from this location acceptable? If the answer is yes, the code is shown and can be inserted to the interface pop up in the new device.

Although this is totally a proprietary solution by Apple, devices by some other manufacturers are included as a part of this. A new Microsoft Windows running machine with iCloud enabled can be used with two-factor.

Other common solution to implement two-factor is to use an SMS to send the authentication code to the end users mobile device. The security of SMS protocol is from the 80's and it is not safe [Brandon. 2017]. If one has a malicious mindset, an own intercepting cell phone mast can be easily created and send all traffic through it. A device named IMSI()-catcher [Ooi. 2015] can be deployed with a relatively low cost to identify as a real cell phone mast. But the device actually intercepts all mobile phone communication and then forwards it to the correct mast, and the end user sees no difference in mobile phone usage.

A sample device has been built around a regular laptop for 1500 USD, where the laptop price was the biggest cost of the build.

2.3.7 One Time Password (OTP) Tokens

One Time Password token that is used as a second factor of the authentication process is usually created with external hardware that shows a token that changes in regular intervals. Hardware implementations of this approach include RSA ID token generator [RSA. 2017] that is carried with the user

where log in is needed. After the initial login has occurred the current state of token is checked from the token generator and entered to secondary login.

The OTP password token login method is based on calculating the token from unique seeds that are either put to the hardware that is used by the end user. They are shared to the end user and end user can use software solutions such as MobileID [Deepnet security. 2017] downloadable from Apple App Store to generate the tokens from the seed in real time.

From the security point of view, OTP tokens increase the security. They are invulnerable to replay attacks since the OTP can never be guessed with repetitive attempts. The only ways to compromise this security approach is to first find out the original user id and password and the then implement a method of finding out the true OTP.

The usability of this approach varies from a little annoying to constant harm. It is a little annoyance to check and enter the token every once in a while, but entering constantly changing token many times a day is not usable.

2.3.8 Peripheral Device Recognition

Peripheral device recognition is one of the second factors in two factor authentications. It is implemented by physical existing hardware of the end user. Existing hardware might be a smart phone or an iPod etc. The end user then plugs the device into the computer when logging procedure is in progress. The usability benefits for the end user are the same than in the one-time password solution done by application that the end user doesn't need to carry around external piece of hardware just to get logged into some service.

2.4 Password Storage Options

To help the user with using many different passwords in many different sites the manufacturers of operating systems and browsers have different kind of general storages available for handling lists of different passwords at a safe place. The security of these solutions has been challenged. These kinds of solutions increase usability of the end user drastically but are they safe? We examine some publicly raised issues concerning them to raise awareness.

2.4.1 Browser Password Storage

The contemporary browsers solutions are examined closer here. These password storages are similar to keychains covered later but they are tied to the used browser.

Opera offers the Password Sync service where the end users passwords can be synchronized between all devices where which logged in and using Opera. The service operates by setting the user an account data to company's own

servers and it accesses data from there. In 2016 that server was breached according to announcement done by company itself [Opera. 2017]. Announcement says that 0,5% of all Opera users are using the service, passwords were stored encrypted but they are reset as a precaution.

Another real issue associated with a local password manager in recent years was when Google Chrome password manager could be entered to see the plain text passwords by others. This was enabled if the end user left the browser on and left from the proximity of the device for a while. Password storage can be found at: chrome://settings/passwords

2.4.2 Keychains

MacOS and Windows computers offer a keychain as an operating system service to the customers using the computer. The idea of the keychain is to offer a machine wide safe password storage for using all applications in the machine. E.g. when using the same network service with two different browsers the service recognizes that the end user is trying to log into the service and offers a saved password to be used to log in to the service. In MacOS the keychain usability from end user point of view is fluent.

How secure are these keychains which have an ability to store all of the delegate passwords? The safety of MacOS keychain has been proven to have vulnerabilities [Apple Insider. 2017]. What this presented vulnerability can do is that an unsigned app designed by anyone that has the software development skills can ask for plain text keywords from the keychain. In a proven concept, the social media passwords could be extracted out of the computer that way. That could also be done without user knowing it.

When keychains are analyzed for their usability alone they increase the usability of other software also. If the end user has saved e.g. the passwords used for Wi-Fi to one of the devices then when the end user enters into the same space the other device also connects to that Wi-Fi without any interaction from the end user.

3. Usability and Secure Features

This chapter covers usability and secure features as they are covered in the published literature.

3.1 Literature in the Usability Field

Norman [2013] has an interesting perspective to everyday objects. It makes a bold claim that if there is a concurring problem in using some normal item, the problem is with the design of the item and not the user.

Norman [2013] examines door handles closely. The book suggests that if the door is only meant to open towards outside it might as well have a metal plate instead of the handle. Then the interface would better indicate what it is meant to be used for.

Norman [2103] introduces six principles of evaluating the usability of a product: visibility, feedback, constraints, mapping, consistency and affordances. Visibility refers to that end users need to know instantly their options in using the product/feature. Feedback is a reaction to end users action. Constrains are the physical limitations of the product. Mapping refers to the relationship of control and effect. Consistency is the need that same action produces same effect every time a feature/product is used in a similar way. Affordance is the relationship between what something looks like and how it's used. The applications of these principles shape the thinking of the reader.

Although Norman [2013] does not involve security features it aids in understanding the design from the end user perspective. Reading the book alters the mindset of its reader to re-assess daily failures with everyday objects differently. It raised questions like “why is this done like this?”. This book is really relevant learning material for usability side of the thesis.

Human Computer Interface and its history has been covered in McKenzie [2013] The book starts from the early days of computing with text based user interfaces with really limited ways of interaction. Original computer user interface has evolved starting from text based command line interpreters to the graphical user interfaces. Graphical user interfaces apply what you see is what you get interpretation of the interface. The actions in the user interface are made to resemble the real life interactions, e.g. dragging a document to the trash bin. It also embraces the concept of thorough testing of the product. The book measures adoption rates and error rates of different input methods.

3.2 Studies Related to Secure Features and Usability

Cranor and Garfinkel [2005] describe usability aspects in an attempt to get the reader to think the concept of usability before implementing secure features.

It is focuses on human centric secure feature design than the specific technical details. However, the knowledge covered in the books found with literature search are from the time when there was no smartphone and instant messaging apps. Moreover, the books do not include how the features and properties of any authentication method can be tested and verified.

Sasse and Flechais [2005] examine usability of secure features. It examines the old security expert's mindset of people being the weakest link in security. While that might be true it is often offered as an excuse for not usable secure features. Evidence of secure features not being usable is presented. Book puts human in the center of implementing secure features. The aspects of human nature that affect to the use of the features are closely examined. The new mindset should be to comply to the human factors in the secure feature design.

Sasse and Flechais [2005] examines human nature by thinking of how the need for protecting one's own property has evolved through times. This socio-economical view helps readers to understand better the often-complicated nature of secure features. The secure features are a need based implementation but they answer to the need for the availability of the protection itself and not usually to the usability. Then it links the study content to today's world and electronic ways to communicate and do transactions.

Sasse and Flechais [2005] find out two main reasons for not wanted human behavior using secure features. Either humans do not know how to use secure features or they choose not to use them. Both the factors can be linked to poor end user experience of secure features. First is complexity of a feature. Second in my opinion is worse. There is a secure feature that end user knows how to use and could use it to improve the overall security. Instead he/she chooses not to use it and sabotages the security of the service or device.

This behavior is linked to the fact that using some secure features is hard and time consuming. It is easier not to use them. The study concludes that most people can't comply with the password standards set those days. Reasons for that mentioned are remembering a large number of passwords, the password complexity requirements and the demand to change them often. This paper is really useful for creating ground knowledge of the research subject.

In the paper by Dillon et al. [2016] study the reasons why security and usability aspects are often overlooked in defining the software features. Claim is made that they are not considered as strategically important parts of the features. The paper tries to identify value based changes to usability using a survey and analyzing the results. The paper continues in the lines set by Sasse and Flechais and they both combined give a coherent look of the state of the secure feature design.

In the comparative study based on national culture and gender [Berki et al. 2017] of cyber-security knowledge in higher education institutes is performed. Knowledge level of cyber-security plays important role in the use of security. Study examines five different countries: China, Finland, Greece, Nepal, and the UK. Main focus for secure feature part is to examine knowledge related to cloud service security.

Other relevant part of the paper [Berki et al. 2017] is to examine how different cultures affect the situation. It suggests that the educators of cloud security should consider these factors when designing solutions: incorporating real needs and taking into account cultural awareness in higher education degrees and corresponding industrial training schemes.

The comparative study examines the adoption rate of cloud services of people from different cultures. Study reveals that 20,45% of people who do not use cloud services don't use it because they do not trust services, and 18,94% of people participating this study don't use cloud services. If we consider that 20% is divided with five we get that about 4% of the participants in this five country study don't trust cloud services. Study combines the security field with cultural differences. It is important to broaden understanding that end users of secure services are not all alike. There are some built in assumptions that come from the culture that affect the way secure features are used.

3.3 Studies on Improving Usability and Security of Authentication Methods

To reduce physical carry around proprietary devices for OTP (One Time Password) creating they can be replaced with mobile phone apps. Apps like Deepnet MobileID [Deepnet security. 2017] depicted in Figure 1 let users use the device they already have to create authentication one time passwords instead of carrying proprietary token generators with them. A traditional OTP generator such as RSA SecurID [2017] depicted in figure 2 generates keys that are visible for all that have access to the device.

Mobile device itself most likely has some sort of authentication method of its own and further adds the security of the use of OTP.

In addition to usual OTP generators, the app provides patented the two-way authentication that authenticates both the end user to the service and the service to the end user. Seeds are fed to the device app and after that to access the Deepnet Security app is simply opened from the end users unlocked mobile phone. RSA SecurID token generator usually hangs in the keychain or together with a badge on the end users neck. RSA also offers software token solutions for mobile devices.

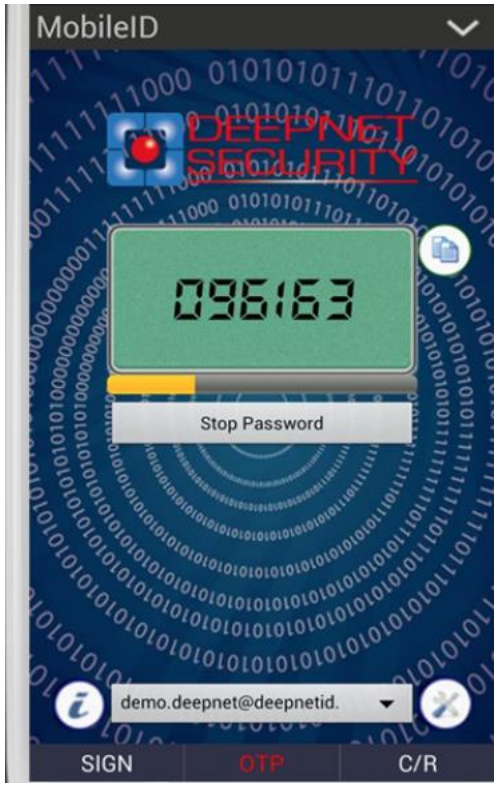


Figure 1. Deepnet Mobile ID app [Deepnet security. 2017]

Figure 2. RSA SecurID token generator [RSA. 2017]

New methods are developed for the two-factor authentication where the second factor is a picture taken with a mobile phone camera of an everyday feature the end user possesses. The taken picture is used to replace One Time Password. Pixie [Cimpanu. 2017] is one of these future systems that analyze the picture taken. In the method, a reference picture is taken by the end user and initial log in using text password is done. Then when in authentication end user retakes the picture and an app in the mobile device compares the pictures. Basic security of the picture taking is that object only known by the end user. Security of the method can be further added by taking the picture from a certain angle only known by the end user.

Harmonizing UI towards universal experience on similar authentication situations in the whole device range between manufacturers would assist the

familiarity of the features. The user experience improvements by familiarity encourages users to use secure authentication methods.

One of the problems associated with mobile device touch screens is the ease of shoulder surfing the passwords: A method used to acquire password or some similar other method of user authentication. PIN code entering, drawn patterns and password entering are especially vulnerable for this.

There are two useful paths in overcoming this issue both security and usability wise. First is the increasing the use of fingerprint reader that creates a unique authentication that can't be shoulder surfed. Other usable methods include the use of facial recognition. As examined in Chapter 2 all facial recognition methods may not be so secure. Apple's FaceID promises to get security aspects into acceptable level by measuring 30 000 facial point and creating a map of users face. To increase usability it attempts to adapt to users changing facial hair and hats etc.

There are also promising studies about creating usable secure authentication methods to touch screen that are effective against shoulder surfing. *Multi-Touch Authentication on Tabletops* [Kim et al. 2010] is one of these future looking studies. It focuses into the issue of tabletop interfaces where password entry discreetly is virtually impossible. It tries to solve an almost impossible problem. How to limit visibility of the authentication only to the person who is doing the authentication in an environment where there is really limited privacy. The example methods from the study are shown below. A group of three persons at the time were selected and one of them was chosen to enter the authentication code that was only known by him/her and the other two were trying to acquire what the password entered was. It was noted during the study that these two people that were trying to solve password grouped proactively and compared notes to solve the password. The study sees this as a feasible situation in real life. The Shield PIN method depicted in Figure 3 only shows the PIN inserting keyboard when a palm hand covers the green area and so forces the user to shield the pin keyboard.

The Slot PIN method in Figure 4 moves the inputs into rolling wheels in slots and it makes people around the end user harder to see what numbers are selected.

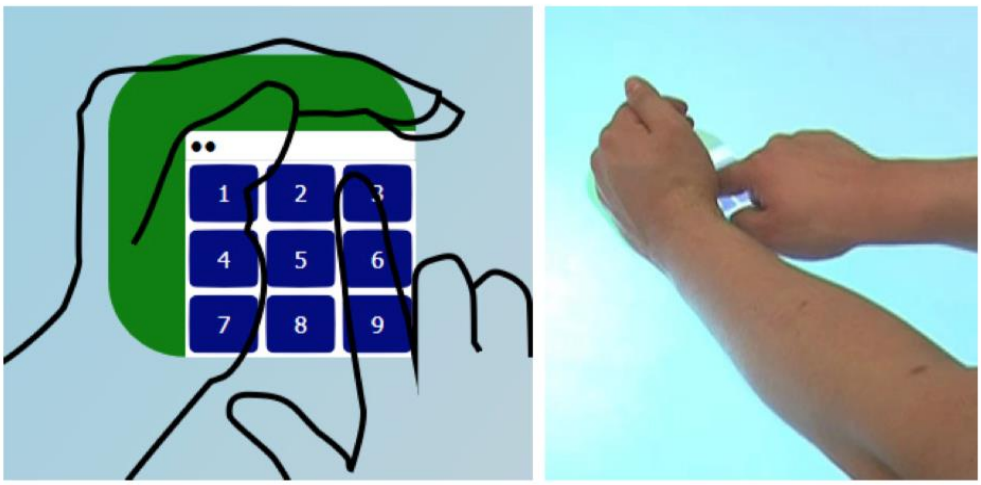


Figure 3. Shield PIN method [Kim et al. 2010]

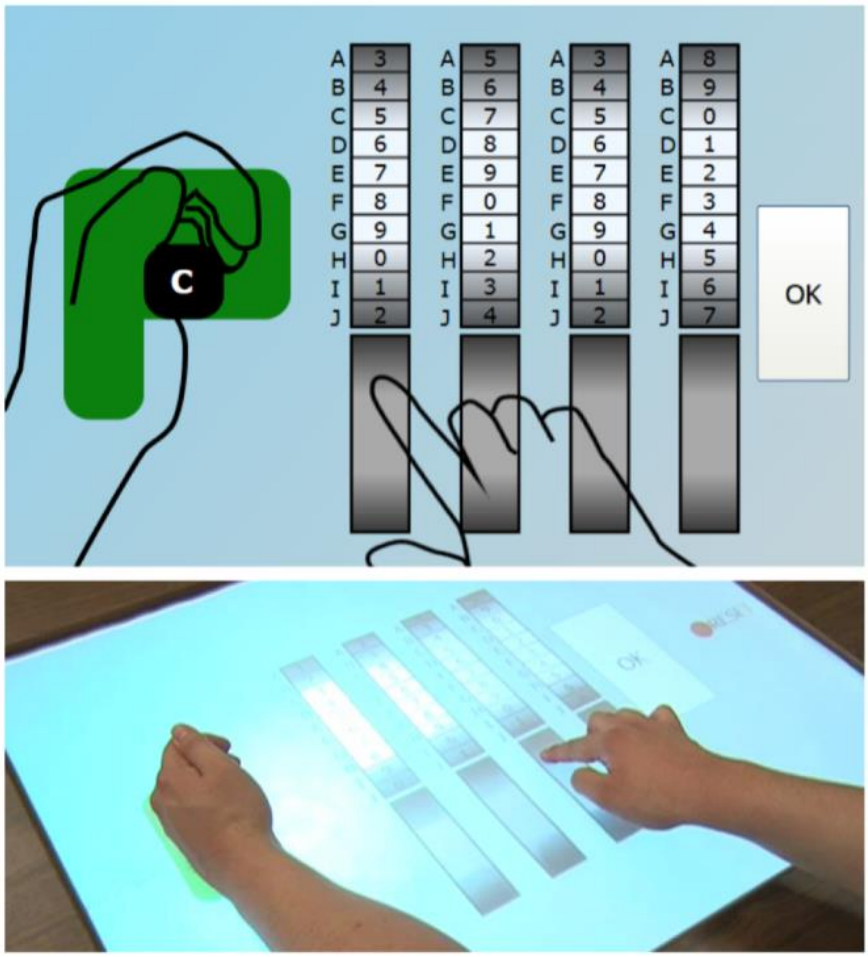


Figure 4. Slot PIN method [Kim et al. 2010]

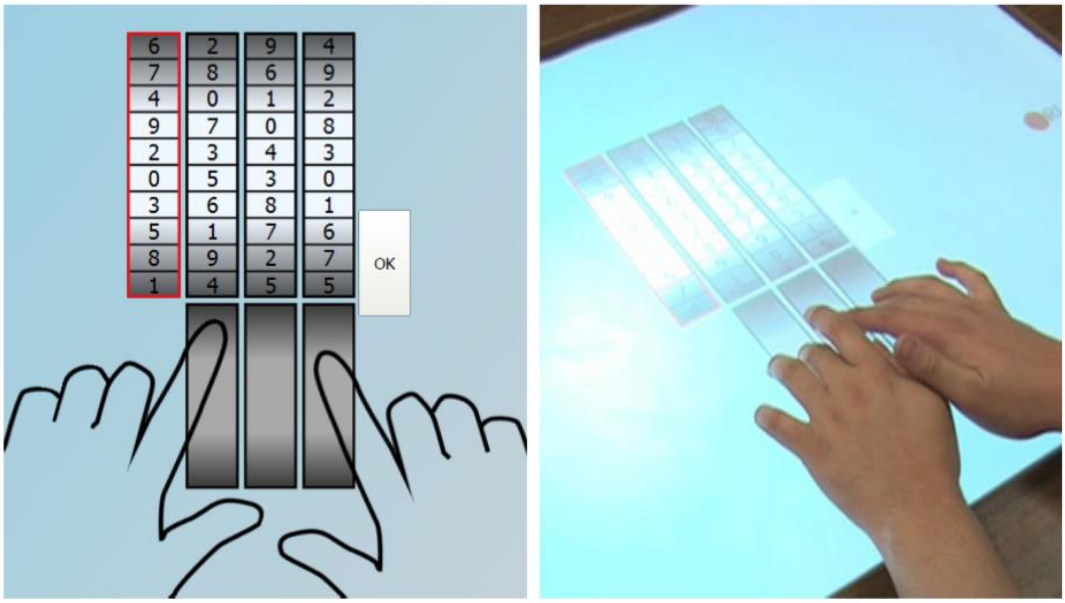


Figure 5. The Cue PIN method [Kim et al. 2010]

The cue PIN method in Figure 5 shows a hidden row alphabet against which the PIN input must be lined. The row alphabet is only visible when hand covers the green area totally.

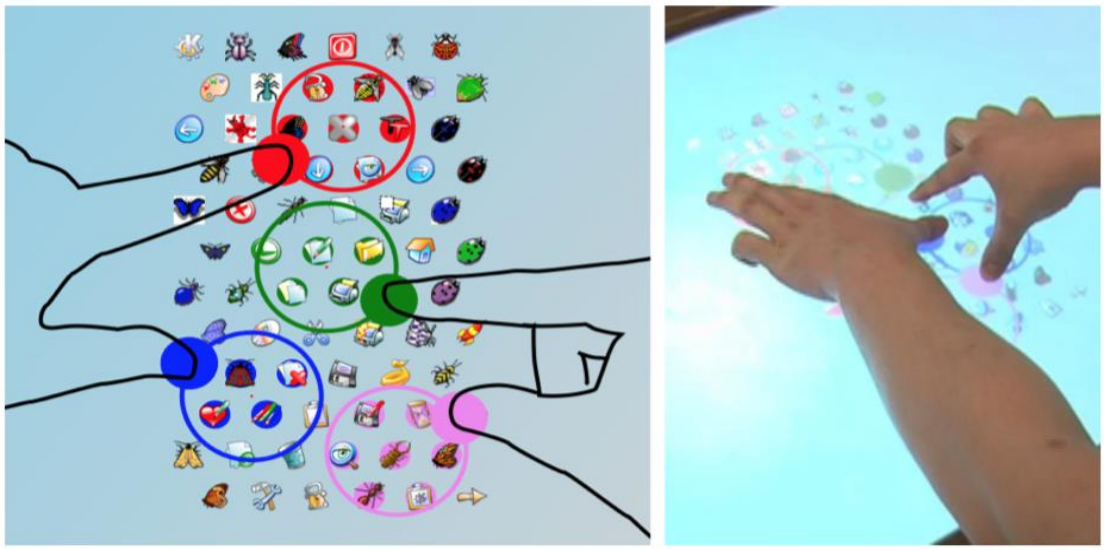


Figure 6. Color-rings method [Kim et al. 2010]

The Color-Rings method depicted in Figure 6 combines graphical and multi touch aspect to authentication. At first a user is asked to place four fingers into the display (ideally both thumbs and index fingers). Four rings of different colors are drawn around the fingers. The user must place four rings concurrently on top of the correct objects and confirm the location by releasing all fingers. To make the method secure only one of the rings is a correct one and others are decoys. Password guessing would require to find both the right ring and right objects.

Multi touch pressure grids in Figure 7 are used to select the correct pictures and the selection is only visible to the giver of authentication since the hands stay stationary.

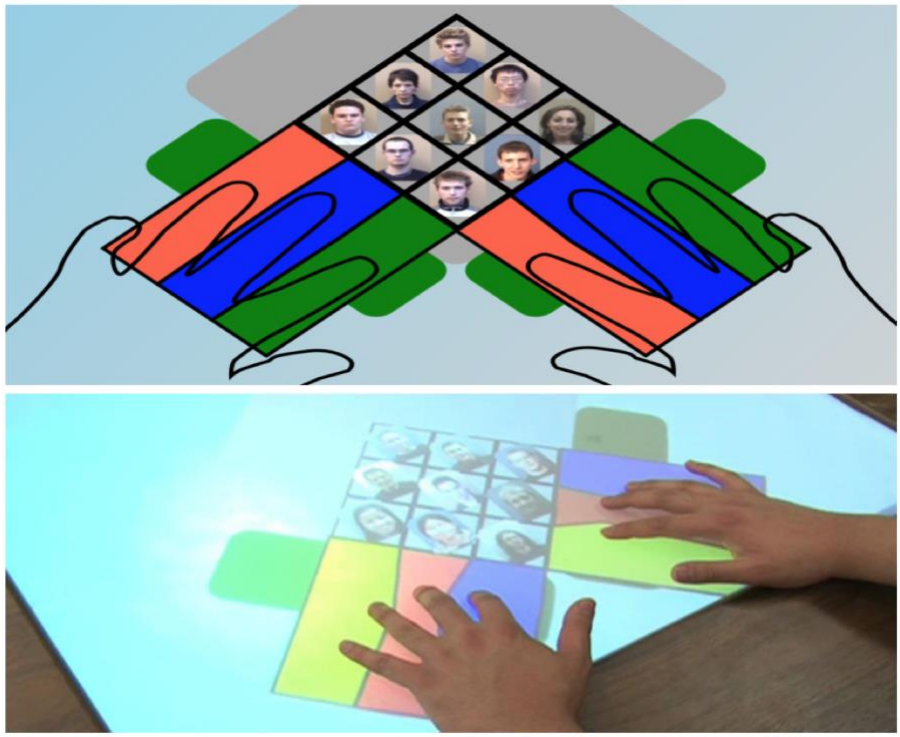


Figure 7. Pressure grids method [Kim et al. 2010]

These methods were all applied to the tabletop but e.g. shield PIN would be easy to replicate to mobile devices smaller screens. Also some kind of implementation for pressure as an authentication input method would be interesting. The Apple Corporation uses pressure sensitive touch screens in its contemporary mobile devices so the technology would already be in place for such a method.

Also, replacing the drawn patterns with the color-rings method requiring multi touch use would improve security of the mobile phone authentication

without sacrificing too much usability. It is easy to shoulder surf anyone's drawn pattern.

The downside in using these two methods in a touch screen handheld device is the need for using two hands. Then user would need to place the device somewhere out of hands to operate these log in methods.

3.4 Other Thesis Work Related to Security

The following three thesis cover security area. Din Toan Nguyen's MSc thesis [Nguyen. 2016] describes the importance of network security for security itself and also for the customer's needs to be able to trust the network provider. It is a highly technical thesis that focuses mostly in describing the secure API calls. At first the most relevant commercial solutions are described and then self-made implementation is introduced. It is a well written thesis with deep technological knowledge.

Thrushna Nalam's MSc Thesis [Nalam. 2015] is focused into RSS (Really Simple Syndication) that is an XML protocol for updating information. Thesis sees the decline of RSS use to be part of its security flaws that allow it to be used as a channel for SPAM. It shows deep technical knowledge about the subject. It includes the user experimenting part.

Sunil Chaudhary's MSc Thesis [Chaudhary. 2012] focuses in the identification of phishing sites and attempts. It tries to identify anomalies in the Uniform Resource Locators (URLs) and phishing websites. And to determine an efficient way to employ those anomalies for the automated phishing detection.

To accomplish this highly ambitious goal the author performs meta-analysis for current used phishing techniques. Then a wide set of actually found anomalies in phishing sites and their source code is examined with this new technique. During tests, it is notified that the key elements in discovering the existence of phishing site have been changed into different not-discoverable form. And those used markers are no longer effective in a fight against phishing attempts.

The thesis takes a look into URL and DOM (Document Object Model) structures of phishing websites. It is interesting because of past front end testing projects. It is interesting to see similar occurrences in DOM structures between malicious sites.

These three theses above have more technical approach to security field. They provide good reference how to write a security related thesis. The content itself differs from this thesis field.

3.5 Summary of Literature on Usability and Secure Features

All the books, papers and thesis read about subject of security cover well the technical points of the process and go deep into the secure features in an attempt to improve overall security. The security field has attracted good thinkers with a lot of knowledge of the subject. The literature was in part chosen to get representation of all covered fields of the thesis.

The field of end user experience of the secure features has little less coverage. Cranor and Garfinkel [2005] mention "*It is easy to make a secure computer. Just lock a computer into a safe and throw away the key*". The books that are written about usability of secure features such as Sasse and Flechais [2005] offer some help in the field of it. Relevant theories regarding human psychology towards using secure features were presented. They helped forming overall understanding of the subject. Furthermore, explained own personal selections of secure features.

Combining books that are strictly about usability and end user experience broadens the field of study to the right direction by providing more proven field of usability studies that can be applied to security features also.

The books describing the relation between usability and security such as [Sasse and Flechais. 2005] and [Granor and Garfinkel. 2005] described a coherent display of the situation surrounding secure features. I did not find any papers or books about secure features that are considered usable by end user.

4. Verifying Secure Features

Verification and also therefore the existence of overall security of features is studied in method level in this section. Often to be able to verify the existence of security intended features need to be tested in the happy mode. Try to be able to complete test scenarios where the end user just wants to use the device. On the other hand, when the actual protection is considered there is a need to learn the mindset of the malicious person. Good functionality increases usability.

4.1 International Software Testing Qualifications Board

International Software Testing Qualifications Board, ISTQB, certification [2017] provides clear structured reference material in applying methods to testing, test development and management.

ISTQB certifications are used to define the level of professionalism for experts working in the field of software testing. Levels include foundation certifications, the advanced level and the expert level. They are also split to agile, core and specialist levels.

In the foundation level, there are many different kind of expertise certificate options available. The advanced level is then further split into test designer and test manager levels and there is a separate path for different kinds of advanced professionals. Even if the certificate exam or the course(s) have not been taken the course materials are helpful in testing. Materials in the course section are divided into sections matching the different levels of certifications.

In certification syllabus, process view of software testing of any software is elaborated. It is intended to gather scattered testing practices and processes of software testers to form a globally uniform testing process. This kind of uniform experience then adds professionalism to software testing. And also, it allows software testers around the world to understand the phases of the processes everywhere.

ISTQB foundation is a non-profit organization that is formed in 2002. It is located in Belgium. The ISTQB Certified Tester has become the world leading certification to validate software testers. The work of the foundation is largely based into voluntary work by the software testing professionals. These volunteers then arrange certificate exams. In Finland, these exams are arranged by Finnish Software Testing Board [FiSTB. 2017]. In its pages FiSTB names its tasks as follows: arranging certification exams, organizing testing assembly seminars with cooperation with universities and guest lecturing.

4.2 Verifying the Existence of Functioning Security

A secure feature as such is not meant to be visible to an end user. This also presents a challenge to be able to verify the existence of the secure feature when the feature's specification is not available. An end user only notices security when it is working too well, e.g. preventing the intended use of software.

To be able to verify the existence of functioning security, there is a need to first understand what is tested. The PIN code is referred in examples as a mobile phone password entry. Most of us are familiar with that feature and it is useful as an example. As with any given verifying task there is a need to follow a certain type of testing process.

Verification flow for a new not known feature is given below:

- Getting to know the product. Initial acquiring what a feature is supposed to do
- Initial draft testing
- Exploratory testing sessions
- First actual test run
- Finalizing tests
- Planning for negative testing with a malicious mindset

How this process is handled in a more structured way is to follow a process for testing described in Figure 8. The process is from top to bottom flowing diagram with possibility to return from certain phases. These steps are linked to the process. The selected method of testing a new feature is formed by using the ISTQB testing certificate process [2017] as a guideline. And then it is shaped towards the reality of the testing process. There is a plethora of books describing software testing processes in detail and they usually follow roughly the same processes that are also the base of the ISTQB testing process.

This kind of process flow follows also the process guidelines that are studied in other testing related books like *Effective Methods for Software Testing* [Perry. 2006] which follows the circle of plan, do, check, and act. It presents the same testing cycle but in a cyclical form not as an up-down list. The testing part was done to answer research question "What is functional security?"

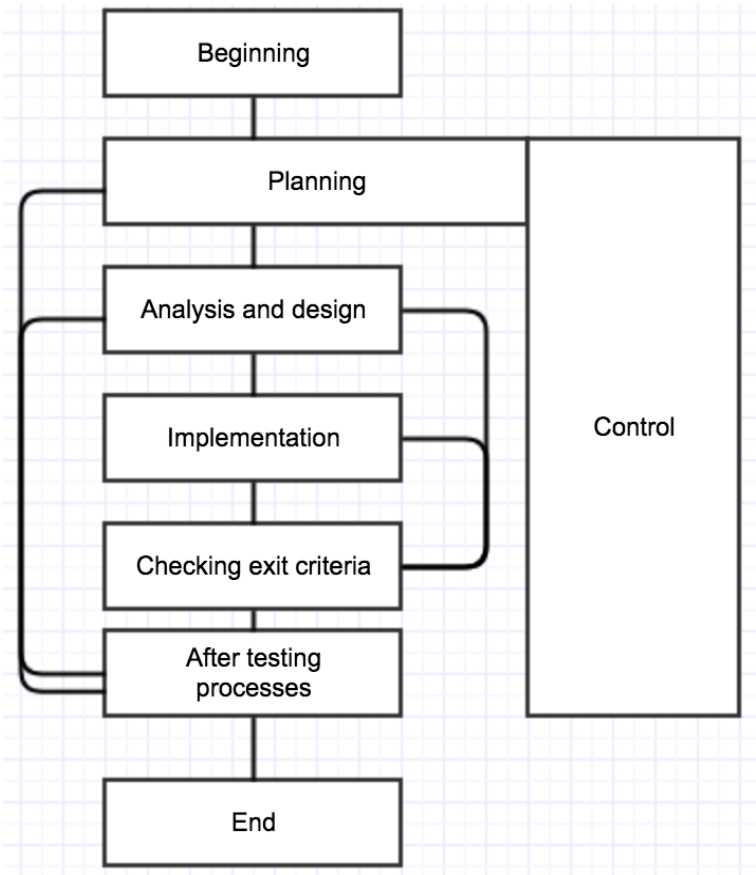


Figure 8: Testing Process Flow

4.2.1 Beginning

What is the feature? Is there any knowledge about the feature? Possible sources for information for answering these questions are specifications and requirements. Information from requirement engineering course proved that good requirements help build good features. Long experience in SW development projects supports these findings. My personal career in SW testing is 13 years long. It includes Manual SW testing, test automation, test design, usability assessment and SW package releasing. Projects have varied from mobile phone development, banking, insurance, to process automation. A test engineer often finds itself in a situation where there really is not any documentation how something should work. This is often the case with all features in agile SW development practices. Furthermore, it is even more common with security features which are not in the center of the interest for most stakeholders.

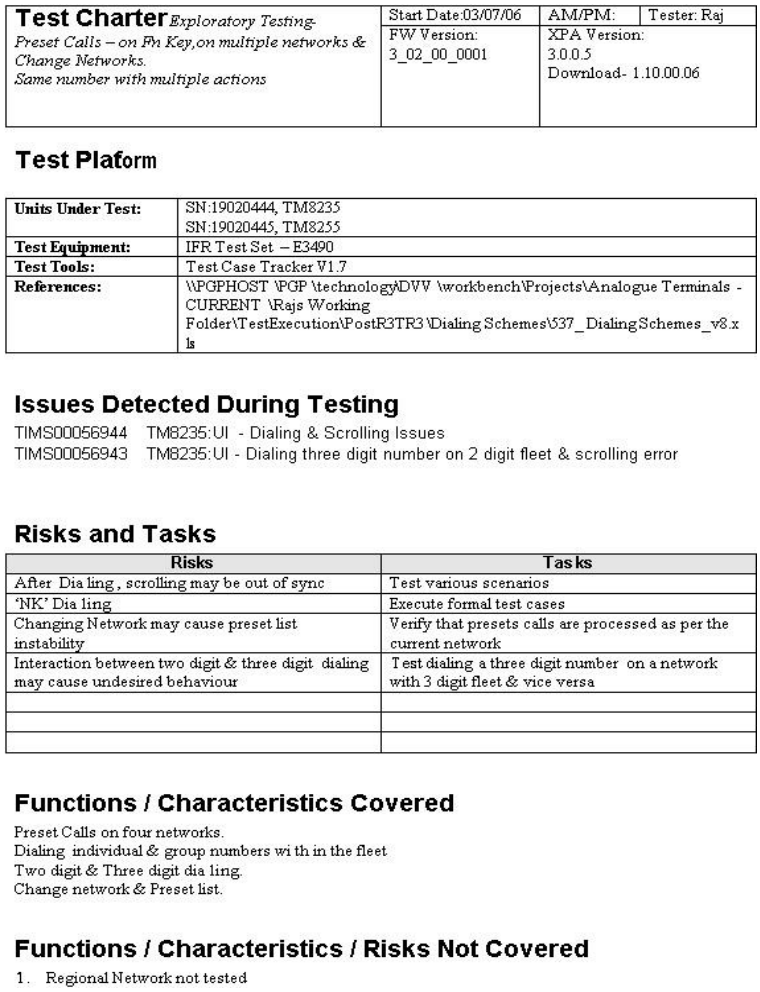
What is the feature supposed to do? How does the feature work? Are there limitations built in the feature? With security features this information often is not readily available and needs to be investigated by the test engineer. The test

engineer can often also affect the usability of features that are seen by the end user, e.g. the test engineer creates bugs, use cases and test cases to support understanding of verified features.

4.2.2 Planning

Exploratory testing is a way to get to know a feature that nothing is known about. The name exploratory testing may mislead. It is a controlled and preplanned testing effort. The test engineer needs to have a plan how to perform the session. The test charter should have a set time and should limit the field of testing effectively. An effective timeframe for executing an exploratory testing session is from 30 minutes to an hour. Then the tested feature is described. Notes are kept on all findings.

After the test round is over findings are examined and based on those finding test cases can be further fine-tuned. And possible bugs can be logged into test management tool.



Test Charter *Exploratory Testing-*
Preset Calls – on Fn Key,on multiple networks & Change Networks.
Same number with multiple actions

Start Date:03/07/06	AM/PM:	Tester: Raj
FW Version: 3_02_00_0001	XPA Version: 3.0.0.5 Download- 1.10.00.06	

Test Plaform

Units Under Test:	SN:19020444, TM8235 SN:19020445, TM8255
Test Equipment:	IFR Test Set – E3490
Test Tools:	Test Case Tracker V1.7
References:	\\PGPHOST \PGP \technology\DVV \workbench\Projects\Analogue Terminals - CURRENT \Rajs Working Folder\TestExecution\PostR3TR3 \Dialing Schemes\537_Dialing Schemes_v8.x ls

Issues Detected During Testing

TIMS00056944 TM8235:UI - Dialing & Scrolling Issues
TIMS00056943 TM8235:UI - Dialing three digit number on 2 digit fleet & scrolling error

Risks and Tasks

Risks	Tasks
After Dia ling , scrolling may be out of sync	Test various scenarios
'NK' Dia ling	Execute formal test cases
Changing Network may cause preset list instability	Verify that presets calls are processed as per the current network
Interaction between two digit & three digit dialing may cause undesired behaviour	Test dialing a three digit number on a network with 3 digit fleet & vice versa

Functions / Characteristics Covered

Preset Calls on four networks.
Dialing individual & group numbers wi th in the fleet
Two digit & Three digit dia ling.
Change network & Preset list.

Functions / Characteristics / Risks Not Covered

1. Regional Network not tested

Figure 9: Exploratory testing session charter document [Stickyminds. 2017]

In Figure 9 the upper part of the document describes the purpose and target of the exploratory testing session, starting from the issues detected during testing part and risks and tasks. This kind of charter document is formed combining the original charter with testing target and notes done during testing. Bug reports are written after testing.

4.2.3 Analysis and Design

The initial getting to know the feature part has been completed. The next logical step is to start forming initial draft test sets for the feature.

Writing a test case has a universal structure that needs to be followed (terms in Figure 10 are in brackets):

1. Prerequisites (Description) part describes what needs to be done before test execution starts. If a mobile device is considered, Device Under Test (DUT) needs to have SW installed, and is booted up to a certain state where performing the test is possible. In addition, a SIM card may be needed etc.
2. Test steps (Test Details) describe the actual steps that run one by one and what is expected to happen after each step
3. After the final step, the last part describes the expected result (In last step) i.e. what is supposed to have happened to the device when all steps of the test have been running. This is also called defining the exit criteria for a single test.

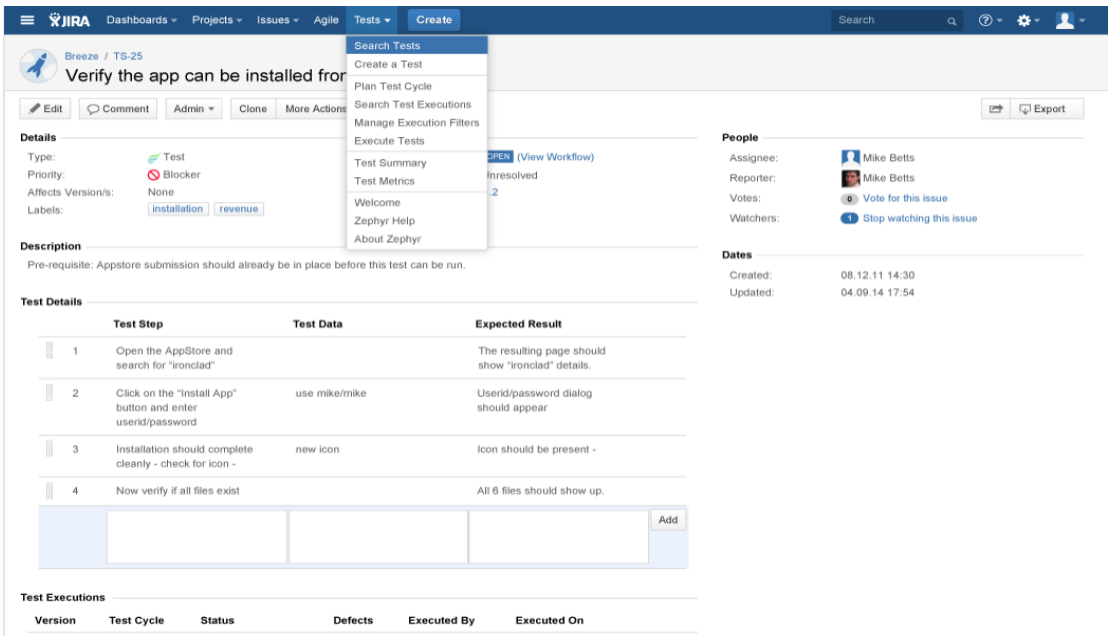


Figure 10: A test case in the testing tool Zephyr [getzephyr. 2017]

4.2.4 Implementation

These initially created tests are then used for the first actual test run. Tests are usually in a draft state then and findings are written down. There is a common exit criterion for the test run defined.

For the defining part, it might be that all tests need to be run no matter what the results are. When testing continues with reoccurring test rounds rules evolve and usually become tighter.

The first test run results are communicated to stakeholders. Communication is often overlooked in the testing process. But there is no point in testing in the first place if the results and findings are not communicated to everyone that benefits from them. A communication plan that is often made by the test manager describes in a clear way what is tested, what is the outcome and what are the bugs. The test manager communicates before the test round starts the data that is required from the test engineer for the test reporting.

4.2.5 After Testing Process

After the first run the findings are analyzed and tests are finalized to a non-draft form. After this stage, the tests are ready to be run by any qualified test engineer. So, it is important to look at the tests as not an own personal creation but how someone else might understand them.

Usually this stage will improve the prerequisites part so that anyone new to that particular test can look what needs to be done before that test can be run. Steps and expected outcome became more obvious.

4.2.6 Control

The control part in the testing flow is reserved for the test manager if there is a separate test manager. It is the responsibility of the test manager to oversee the operations. Usually communication and general test operations flow tools are operated by the test manager. The test manager assigns test assignments to the test engineers.

4.2.7 Planning of Negative Test Cases

The planning of negative test cases starts after verifying to agreed assurance that the feature actually works when used as intended. Negative test cases validate the feature against invalid inputs and user behavior.

A useful way of doing negative testing development is to freely do trials of the interface capabilities. In Figure 11 there is a Google Pixel XL mobile phone's PIN code entering interface. That current mobile phone is a really good example

since it uses unaltered Android OS version 8.0 Oreo interfaces everywhere (also called Vanilla Android). (Note before starting to investigate this feature, calling emergency number without a proper cause is illegal in many countries.

It clearly states that the end user needs to enter PIN code of the SIM card, as shown in Figure 11.

There is further info on the state of the SIM card - locked. Furthermore, a text says “emergency call”. The affordability of the interface looks clear so far.

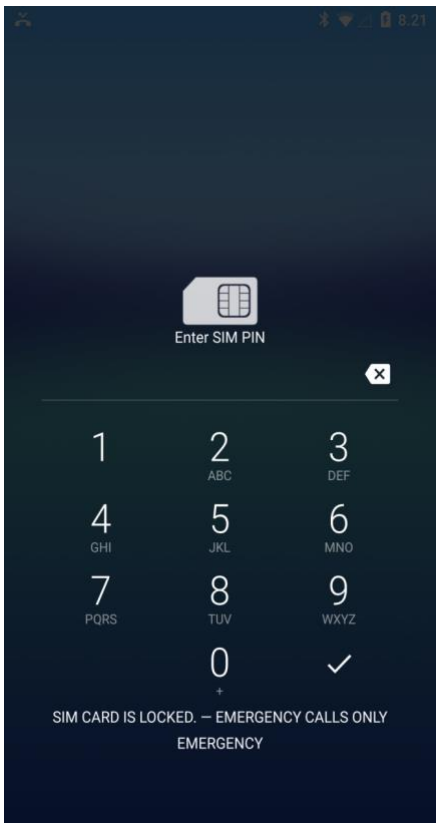


Figure 11: PIN code inserting interface for Google Pixel XL

In this stage, the interface has been used in the positive cases and a correct PIN code has been entered and therefore the SIM card is unlocked. But what if an incorrect code is entered? A pop-up appears like in Figure 12. It states that the user gave an incorrect code and have two attempts left. Left until what? Since there is no specification available this is needed to find out, when the SIM PIN code is typed incorrectly again, results are shown in Figure 13. There is a clear indication that there is only one more attempt left before there is a need to contact operator to unlock the SIM card.

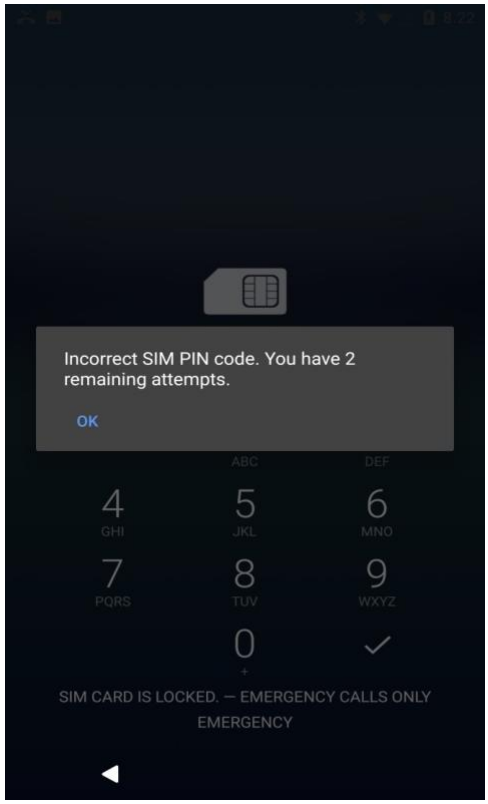


Figure 12: Incorrect PIN code inserted

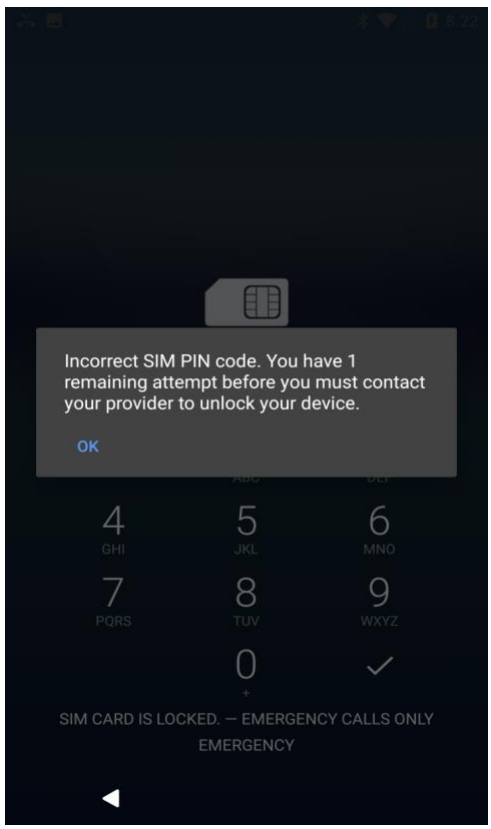


Figure 13: PIN code interface warning after two failed PIN code attempts.

What can be considered a good negative test case for this feature? If the PUK code of the device that is used to unlock the SIM card is known, a test case could be written where the code is entered incorrectly three times and inspected if the card really is locked. Then another (positive) case to follow is to unlock the card with the correct PUK code. Hence, the negative case is to try to unlock the card with incorrect PUK code.

From experience in the software quality assurance it is easy to say that the negative testing is often overlooked and done inadequately. Software implementers are content if the feature works as intended. This is true with all features but security features can break the phone if they are done incorrectly. Trying out negative test cases can be limited to a single try since it may render the device under test unusable.

Can something else be attempted with the interface than just entering the SIM code? When pressing the "emergency call" word below it opens a new menu with new options. It will either allow users to make a call and then there is a button up there saying "emergency information". What if the end user still only wants to make a call to friends instead of opening the SIM lock? Maybe this isn't even the end users own phone. An error message is shown in Figure 14. To make sure all incorrect phone numbers are not accepted requires more testing, but at least that the end user can not call whoever they choose to.

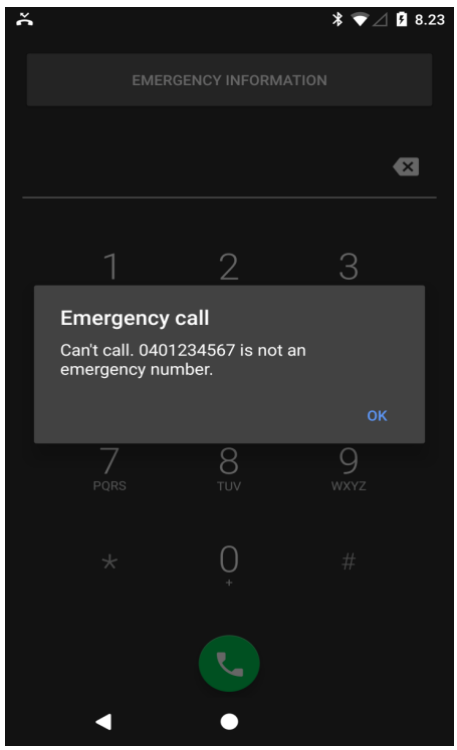


Figure 14. Trying to call incorrect emergency number.

When the button above titled “emergency information” is pressed, it results in either “no emergency info set” or the set of visible info. The interface suggests that the information can be set from somewhere, not from the interface itself though.

A summary of possible negative cases based on getting to know the feature is following:

1. Try to insert incorrect SIM PIN.
2. Try to insert incorrect SIM PIN until card is locked.
3. Try to unlock SIM with incorrect PUK code.
4. Try different incorrect emergency numbers.

When researching and writing this part of the thesis, the SIM PIN interface of Android 8.0 was challenged. It is possible to enter unrestricted amount of numbers to SIM PIN. If thirty ones are hit into the SIM PIN query, it only says “SIM PIN verification failed”. The interface gives no indication while writing that there is something wrong with the length of end user input. If the emergency info menu has been opened first and then got back to SIM PIN query entering menu to enter the PIN code the first thing noticed when device opens is an open emergency info menu.

Furthermore, if from the emergency dialer is returned directly back into the PIN code query there is a greyed half open drop down menu positioned above the PIN code query as shown in Figure 15. It is not operable but it should not be there in the first place.

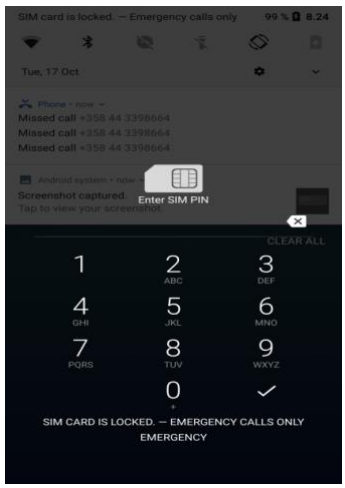


Figure 15: Broken UI after returning from emergency dialer

The greyed menu possesses a security threat. The user can read device. The user can read the last calls and other private info from the menu. None of these did not break the secure structure of the device's PIN, but are bringing security and privacy threats.

4.2.8 Intentional Malicious Trials

Negative aspects were already covered in the trials before. What if there is a desire to be more malicious. An attempt to use the emergency dialer to make personal phone calls was a move into that direction but it was an attempt to use the given interface in the wrong way.

The malicious approach searches vulnerabilities that are not visible to the end user. The software that is flashed or installed to the device may have been altered to do tasks that are not supposed to do. Taking this approach requires patience, going through software specifications that are available and searching for known vulnerabilities found by others.

The Verified Boot flow in the Android system [2017] is examined. It is a system designed into the Android OS to keep track that all booting components have been correctly signed. If signing is different the device should notify the end user that the device software has been altered.

A clever malware which gets root privileges can hide itself into layers of software where it is really hard to detect let alone remove. The solution by Google to this is to divide software into 4k parts and sign those parts with the SHA256 bit encrypted key.

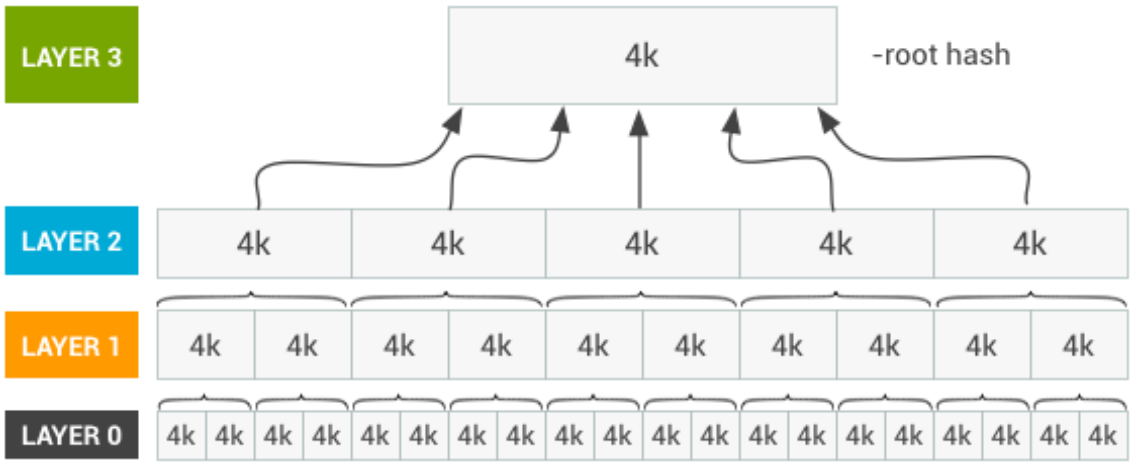


Figure 16. DM-verity hash table [Android verified boot. 2017].

As Figure 16 demonstrates 4k parts in layer 0 are gathered to upper level and the idea is that if the key in layer 3 (that is reasonably large chunk of code to be

inspected) are matching, all below keys are matching too. The upper level is the only layer that needs to be trusted. In addition, there is an OEM key for verifying the integrity of the boot image.

The actual boot flow follows the structure in Figure 17.

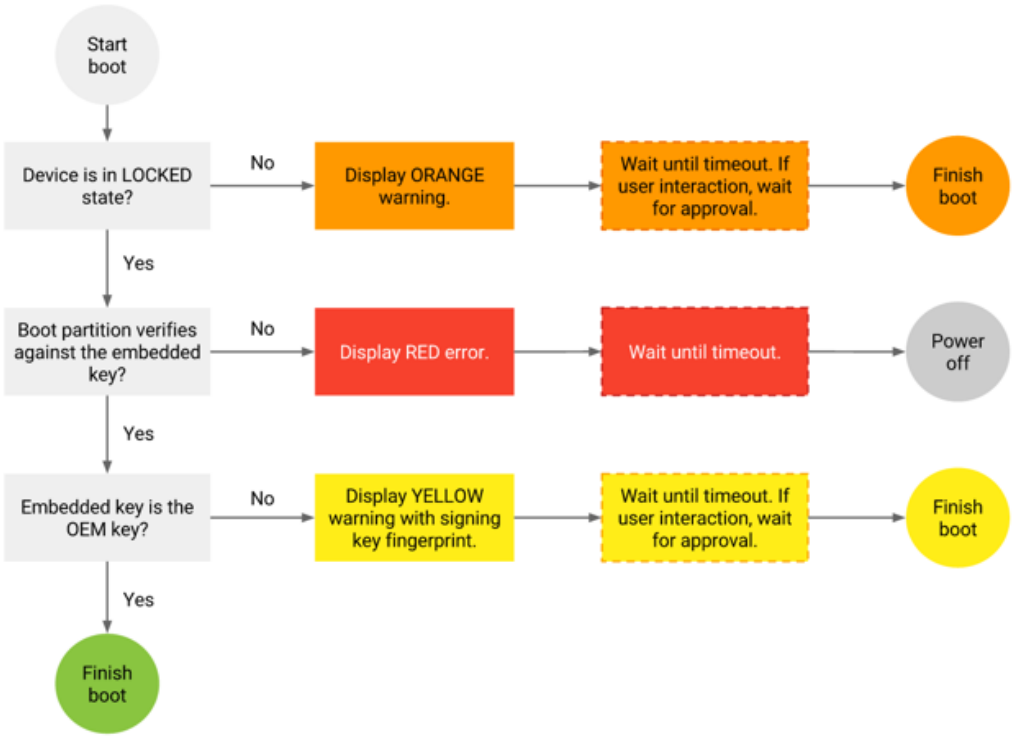


Figure 17: Google Verified Boot flow [Android verified boot. 2017]

At first it checks that whether the device is in a locked or an unlocked mode. Unlocked device can be flashed freely with new software and the locked device can't be flashed. The expected flow for the end user device with everything ok is to follow the left path to finish the boot. If that happens, the end user has no knowledge what is going on during the boot of the device. If the device discovers it is in unlocked state (that it almost never should be in a consumer device) the screen turns orange. If the end user chooses so the device can be booted still. For the device that is both locked and the boot partition fails the verification against the embedded key a red screen is displayed and only option for the device is to power off. If all key verification succeeds but the embedded OEM key verification fails the device can still be booted if end user chooses so.

How can the existence of this kind of secure structure be verified and are there possibilities to exploit this? At first what the test engineer must do is to try to get

all states to appear with correct device states regarding the wanted solution. Next there are some parts of Android OS that are also inspected by verified boot like the splash screen. The splash screen is the picture shown before initial boot and final OS. Nokia used to have hands shaking there in their own OS. Can that picture be altered and is it caught? Pictures can be made to contain hidden executable code. Is it possible to change something in the boot images that does something end user does not want and still be able to go through the boot flow in green state?

One of the big issues of Android OS is that even though devices such as Google Pixel XL receive monthly security updates the most of the other devices have not received them. For testing this means that if the software provider for the tested product has chosen not to use the Google official security updates or is lacking the latest the possibility to exploit those vulnerabilities need to made visible. Malicious approach is in a way made easy against most Android owners.

4.3 Assessing Automation Possibilities

Manual testing and testing trials are at this point of the testing process ongoing. The amount of manual testing is peaking, since the number of tests is high and there are no automated tests deployed to reduce the amount. It is good to look back into the tests already run and analyze the automation possibilities of them. The tools of the test automation are covered in the next pages.

4.3.1 Used Methods

Methods used to verify security features tend to be in the long run the same as verifying other software functionality. Usually in the definition phase and also for the large part in the execution phase there is more manual labor intensive testing involved.

Automation methods are normal use of test automation toolset for test phases that can be automated. Specialized automated toolsets for security only exists as well.

4.3.2 Manual Methods

In manual methods, a test scenario is created that can’t be reproduced automatically. The test subject is analyzed to see what is the wanted functionality and what are the ways to verify it. System design and specifications play a high role in this.

Zephyr is one of the tracking tools used for manual testing. When doing manual testing some kind of tracking of results is needed. These kinds of tools keep track of the test prerequisites and expected results as well.

4.4 Test Automation Common Tools

Test automation example is created with a widely-known set of common tools.

4.4.1 Jenkins

Jenkins [Wikipedia b. 2017] is an automation server that has been written using Java. It is a running application with a user-friendly web page interface. Jenkins is widely used because it can scale up to many different kinds of needs. Jenkins running machines can be divided into master server that handles big things and running slaves that do repetitive hard work.

The work Jenkins is doing is called jobs. If many jobs are lined up at the same time the tool will line them up to a certain executing order. Jenkins interface of running a job is shown in Figure 18.

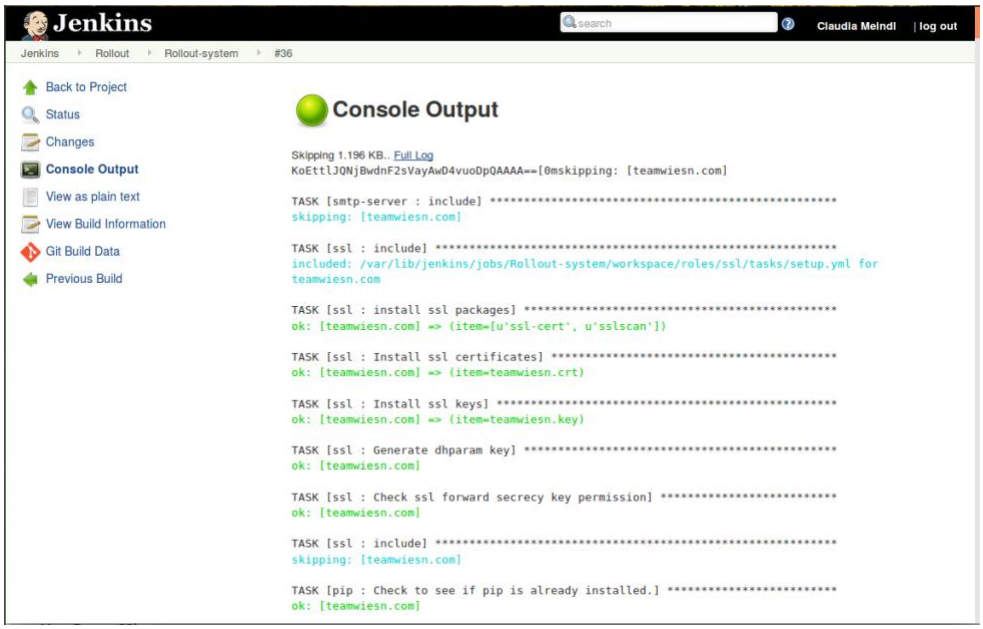


Figure 18: Jenkins UI running a job [Wikipedia b. 2017]

4.5 Virtualization Tools and Environments

Jenkins and other test tools have been traditionally used in a way that the server where data and running programs are stored is a physical set up controlled and located in the using organization's premises.

This has advantages on protection of the data those servers contain. However, they present a problem with scaling of the operation and maintenance of the services.

Virtualization of testing tools offers many different helping points to this issue. Virtualization like name suggests is creating virtual computers that run in a certain environment with certain pre-set and changeable resources. The flexible changeability of given resources and the easy way to relocate those services are the key. This is often referred as scalability.

Tools for virtualization include Ansible, Docker, Vagrant and VMWare. Let's take a closer look into Ansible tool [Ansible. 2017]. A working testing environment needs a working issue tracking software. We are choosing Jira [Atlassian. 2018] for that. Jira is the foundation of Zephyr add on that was discussed earlier. Then we create a Jenkins master. This all can be put up into the master node with Ansible. The Ansible structure is described in Figure 19. There is a host inventory and a playbook against which the system is first set up. From the management node secure SSH connections are made to virtual Ubuntu machines. In this case, the set-up is done in some unnamed cloud provider, in a smaller scale it could even be someone's own laptop if there are enough resources available. Scalability is visible in the picture. If there is a need one more virtual machine it just needs the resources to run it. Then again if there are excess machines they can be left out.

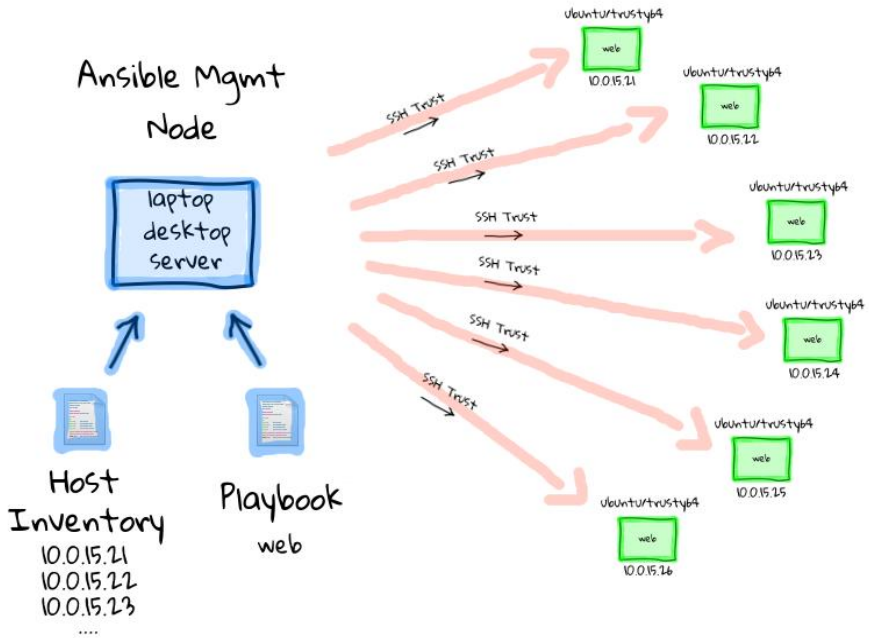


Figure 19: Ansible virtual structure [Devopscube. 2017].

In a simple example about docker use there are readily made free images available for setting up personal Jenkins somewhere. I personally have usually

one Jenkins running in my computer to test out the latest things. The easiest way to set up Jenkins to a personal own machine is to download Docker Toolbox to the computer and find the ready-made image.

Instructions for the system set up are following:

1. Download Docker Toolbox from [Docker Toolbox. 2017]. Versions for MacOS and Windows are available. (MacOs was chosen)
2. Start Docker Quick Terminal
3. Type "docker pull jenkins" to open terminal
4. Type "docker run -d -p 8080:8080 -p 50000:50000 -v $PWD/jenkins:/var/jenkins_home jenkins"
5. docker ps (shows docker identification id)
6. docker exec <identification id> cat /var/jenkins_home/secrets/initialAdminPassword -> Password prints out
7. open browser http://192.168.99.100:8080/
8. insert password to the password prompt
9. Jenkins UI opens

Now a local Jenkins is running inside the real end user machine in a virtual container. The idea is that depending on the configuration this Jenkins could be running at a place that it is wanted. It might be a local server or some cloud providers given storage space.

Virtualization is a mega trend in the world of software development now. Much like Agile methodologies have been. Virtualization allows small companies to start using issue tracking and test automation services in the early stages of the project. Unlike in the past where issuing a working environment for software development was an investment for actual server hardware. Now companies can start off with limited budget and scale up if and when needed.

Cloud service providers such as Amazon (AWS) and Microsoft (Azure Cloud) offer start up services with a small fee and scale up. Larger corporations can change their ageing server room infrastructure into cloud service providers one and pay to them to upkeep and update the server hardware. And only use enough resources that are needed.

4.6 Specialized Toolsets for Verifying Software Security

A subset of test automation toolset is tools that are specifically designed to handle testing that is needed in the security field. They are tools that do repetitive tasks that would practically be impossible to execute by the end user. These tools

make possible to improve testing drastically in many areas that are important to security feature testing.

4.6.1 Fuzzing

Fuzzing [Takanen et al. 2006] is to send incorrect data to a system in order to crash it, therefore revealing reliability problems. This approach started as a school project as described in the book and has expanded from checking UNIX variant's reliability issues to Windows OS and to mobile devices.

Fuzzer generates numerous amount of incorrect data inputs that is sent to different drivers to see what kind of output the input has. It is particularly useful in finding not sanitized driver inputs and missing overflow protections.

In the case of not sanitized inputs all inputs are taken as they come into the driver and assumption of the driver implementation is that all sent commands are formed correctly. By my personal experience this is really common with Microsoft Windows drivers. The fuzzer can easily break surprisingly many of the drivers just by sending incorrect inputs.

Missing overflow protection happens when the memory calls are sent to receiver that contains the proper data but in the end of that data there is executable code that is missed by receiver. Then while the receiver saves the correct input part it also saves the executable data in the memory above the correct data. Then that code can be executed for malicious purposes since it is no longer in control of any real authority of the device's intended memory handlers.

Open Web Application Security Project (OWASP) lists the following open source fuzzing tools for different uses [Owasp a. 2017]:

Mutational Fuzzers

- American fuzzy loop
- Radamsa – a flock of fuzzers

Fuzzing Frameworks

- Sulley Fuzzinf Framework
- rboofuzz

Domain-Specific Fuzzers

- Microsoft SDL MiniFuzz File Fuzzer
- Microsoft SDL Regex Fuzzer
- ABNF Fuzzer

Commercial products

- Codenomicon's product suite
- Peach Fuzzing Platform

- Spirent Avalanche NEXT
- Beyond Security's beSTORM product

4.6.2 Penetration testing

Essentially the penetration testing is smart fuzzing where as a fuzzing tool tries all kinds of wrong inputs including special characters and wrong length of gibberish into the testing target. A penetration test tool tries to be smarter. It sends data that has all the right qualities of being the right kind of input accepted by the testing target but it alters it in a malicious way to see if system inputs and outputs are sanitized. When doing penetration testing it is a bit closer to white box testing where the tester knows about the system design. Whereas in the black box testing like fuzzing where the structure of the system is an educated guess based on the feedback received from the inputs.

A classic example of sanity check is to insert SQL code after a name when inputting it into database using the supplied interface. E.g.

Insert first name: Teemu'); DROP TABLE students;—

If that input is accepted as is there is a real risk that there is no students table anymore in the used database. To be able to try out this kind of input the tester has the knowledge that students table exists and it is not a part of any bigger structure e.g. <structure>.students.

Tools are testing this kind and other inputs to see if some of the inputs results in a failure of database input sanitization. Similar kind of input sanitation failures can be applied in the smaller scale by trying out incorrect types of input to the fields of the input forms for example.

The Owasp foundation [Owasp b. 2017] toolset contains tool Owasp ZAP that can be installed into the computer and then simply order to attack to a certain website. Owasp ZAP tool interface is given in Figure 21. Simplest way to use the tool is just to insert a website address and press attack in the Quick Start section.

The attack keeps ongoing and can be stopped by pressing stop. Tools checks different layers of any given website and reports possible problems found. Example output is in Figure 22. Problems found are grouped by severity and further info on the problematic component location is available. Risk level is a product of Impact times likelihood and risk levels are low, medium and high. It is depicted in Figure 20.

Likelihood and Impact Levels	
0 to <3	LOW
3 to <6	MEDIUM
6 to 9	HIGH

Figure 20: Owasp ZAP impact levels [Owasp b. 2017].

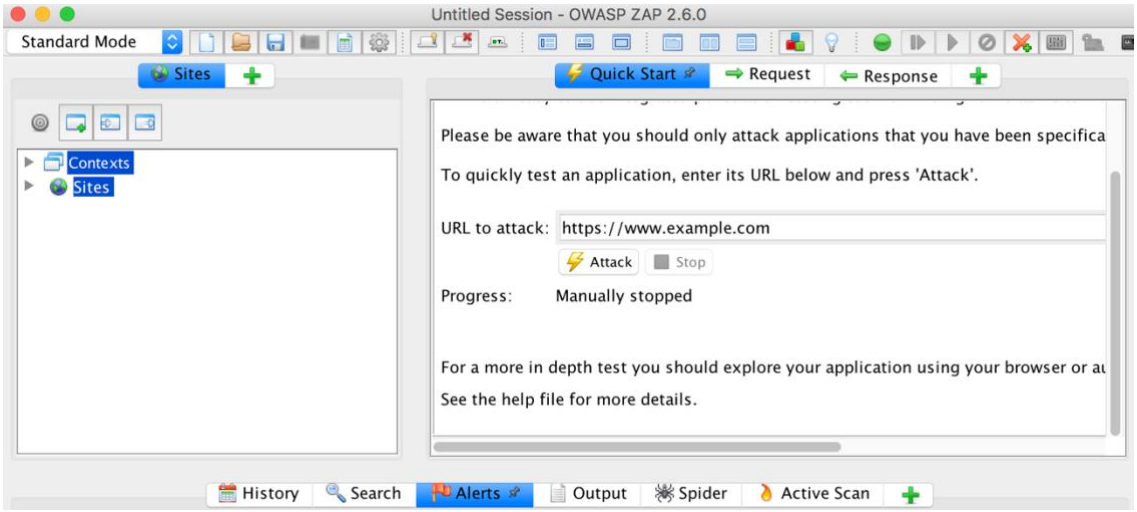


Figure 21: Owasp ZAP Quick Start interface [Owasp b. 2017].

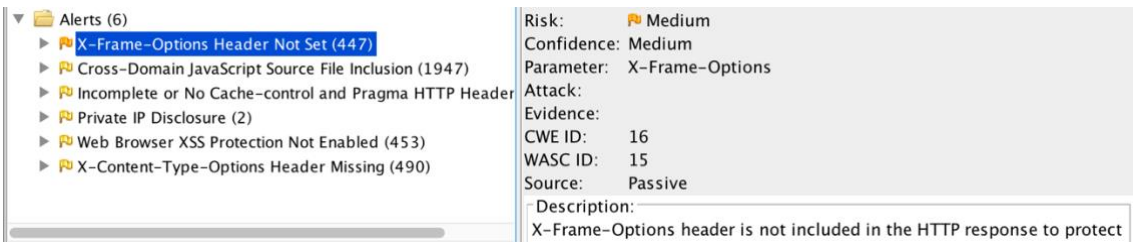


Figure 22: Owasp ZAP problems report [Owasp b. 2017].

Penetration testing tools include [Software testing help. 2017]:

- Metasploit
- Wireshark
- W3af
- Back Track
- Netsparker
- Nessus
- Burpsuite
- Cain & Abel

- Zen Attack Proxy
- Acunetix

4.6.3 Load Testing

Even though systems may operate well and as expected under normal operating conditions in especially network based operations they can be attacked to either Denial Operating Service (DOS) or to make them crash and maybe reveal more about their inner design. Crashed web pages sometimes reveal detailed information e.g. about the database structure.

Load testing automates certain preset amount of load towards the system and observes their behavior. The page load times, crashes and not correct return inputs.

Tools for load testing include:

- Gatling
- Apache Jmeter

4.7 Testing Approach in This Thesis

I personally noticed when taking a look into the ISTQB testing material during the writing of the thesis and then looking at the alternatives made me understand that I personally have drifted away from the process that I am trying to describe here on my every day work. Altering the thesis for these parts was useful for myself and it has itself helped me understand couple of the problems that I face in the real life.

Software presented here are mostly under open source license. The selection is done to improve possibilities for the reader to apply the taken testing approach without committing to big up keeping costs.

5. Conclusions

This study they must be subjected to critical examination against other studies in the area. Next suggestions for future studies are given. Then the thesis writing process from personal development point of view is explained.

5.1 Usability of Secure Features

As this thesis was written I examined quite a lot of usability studies, attended usability course arranged by UTA and read papers about disregarding usability in security features. I compared that to my work experience in the field. This was done to answer the research question "*What are the aspects of the good usability?"* and "*How to approach a new feature that is supposed to be tested?*"

My personal experience matches the findings of Sasse and Flechais [2005] and Dhillon and others [2016]. Secure features do not appear to play an important role in the end user experience design plans. There is a plethora of them offered; they can be used if chosen so. All secure features appear to follow a plan where they are a need based service rather than a usable service. Many end users choose to use the less troublesome ones. In fact, the only time when security of the devices interests someone is a couple of days period after some serious security breach makes the headlines.

The difficulty of using the secure features has started to play more important role nowadays with company-controlled handsets with strict security policies. It is a real-life example that if putting a correct fingerprint is unsuccessful (in an often unreliable fingerprint reader) after sauna and typing a correct complex password for a second factor authentication the device content will be wiped after five consecutive failed attempts.

5.2 Comparing the Results to Other Studies in the Field

Idrus and others [2013] review authentication methods. Theoretically they are examining the same methods as this thesis. They define the methods better than this thesis by separating terms authentication, identification and authorization. This thesis has a more straight-forward approach. While writing the thesis the usability was seen as the key element of the thesis where as Idrus and others see the methods itself as the most important study subject. This clarification of the stages was complementing the overall understanding and it was added to the beginning of authentication methods chapter.

Seranath et al [2016] study the security indicator in fallback protection. The paper resembles many papers already covered in the literature review since it emphasizes the security part of the equation. The fallback itself has not been a part of this thesis even though it is quite much relying on the usability factors.

Fallback is usually not included directly into mobile friendly interfaces but it is rather a web based service.

Braz [2011] reviews in her PhD thesis the same phenomena than this thesis. Braz covers computer systems mostly and briefly discusses mobile personal carry around devices, referred as Personal Digital Assistants. It is clear that Braz has made same literature based findings about the state of security being a need based implementation where usability, and the end user using the service have been left out of the design. The importance of the end user using the secure services is not a high interest. The thesis has been done six years ago and looking at the authentication methods in this thesis it seems clear that the situation didn't changed a lot. Introduction of biometric scanners such as iris scanning, fingerprint and facial scanning as authentication methods form the biggest change. Furthermore, a mobile identification services such as Google Authenticator [Wikipedia a. 2018] introduce new ways for the end user to identify. As seen from literature review many of the usability based studies in this field are over a decade old. Authentication methods as such are virtually unchanged during that period. This field needs constant reminders to change the attitudes to more compliant to usability design.

5.3 Future Studies

This thesis scratches the surface of the usability of secure features. As presented before in the thesis the field of usability in the context of secure features is not very well covered. Secure features are still considered a need-based service. If the usability of secure features is not further studied and emphasized to the software developers, it leaves more end users without usable security. As described in by Dhillon and others [2016] security developers and designers don't see the importance of usability. Most likely a study that could show the benefits and even the bad implications of lacking usability would drive this study subject area further. There are few interesting ways to research the study subject further.

A questionnaire can be sent to a group of mobile phone users about their preferred use of secure authentication methods. In addition to the preferred method the questionnaire could be also utilized to get detailed views on the usability of the features and the reason that led customers to choose the method.

There is also possibility to start doing test setups for different types of new techniques for authentication or the improved versions of old techniques. The techniques introduced in the end of the Chapter 4 could provide a starting point for this and similar setups could be used that were covered in the future insights

part of the thesis. University of Tampere has project work and project management courses that could be utilized for creating these new implementations. Then a group of hands on testers could be chosen for case study part.

5.4 Personal Thoughts

The understanding of the concept of human computer interaction (HCI), usability and even the computer/mobile systems has been formed in a time when I have been able to use them from the first try outs until the current implementations. Work and education has supported my overall understanding of these complex processes. The claim could be made that the significance of electronic systems usability has been born under my own eyes.

I got interested in computers at early age when devices such as Sinclair Spectrum and Commodore 64 were the must have devices. I started my computing career with a Salora Manager, a rebranded computer for Finnish market. When the transition into mobile technology started, I replaced my Commodore Amiga with a Motorola NMT handset. I was one of the early adapters of GSM technology with Nokia 2010 model in 1994 that even included the SMS sending feature as opposite of many devices of the time that only allowed you to receive the SMS messages. In 1997 I started to work for a growing mobile phone company, Nokia. Since then many different companies and products have been come and gone.

I love the latest additions of iOS and MacOS. I am not missing those 37 installation disks that I used to carry in my case for Word Perfect Office installation back in the early 90's when I was working in an IT-helpdesk.

When keeping a certain level of curiosity towards the latest development everyone can understand and appreciate the significance of this development to people's everyday life. Hence as this thesis and other studies show the development is far from over. It is important to use Kaizen [Lean production. 2017] style small step improvements in security area to open up possibilities and incentives for more ordinary end users to use better and more usable security.

In its own part the verification and security loop hole finding and fixing is really important. This thesis tries to introduce a test engineer approach into minds of its readers.

Also, the usability books referred in literature review have been eye openers, a sort of window, towards a world where there is a common logic in every-day objects.

The secure authentication methods are everyday object for all nowadays. They are used regularly on daily basis. They should work in a way that a conscious mind does not have to be utilized in operating them.

There is a lot more to do in this field of study. I know a lot of security developers from my every day profession and I know that they are very intelligent people. However, they tend to be overly logical about their work. They assume the end users are logical about their choices and use the devices and secure features to provide the extra protection that has been implemented. The end users have a different logic in their choices. They have a desire to make devices as easy to use as possible. Even if logically thinking they then may affect their personal safety. To achieve the ease of use the end user's personal data is left vulnerable for anyone that has access to their devices.

During the last stages of the thesis writing I realized that even though I was first involved with mobile security over a decade ago I did not use any password protection on my mobile devices until 1,5 years ago. I thought drawing patterns and writing Personal Identification Codes was too invasive to my device use. What I did, since my device no longer has borders, I moved the security into the data of the device by changing all of my key personal data into the format that I only understood to mitigate possible security risks if the device were stolen.

This all changed 1,5 years ago when mobile device manufacturers introduced fingerprint scanners. I instantly took it into use for my Samsung tablet, and later on into my iPhone and iPad, and lastly into my Apple Watch. I did not have any protection in my old Android driven smart watch. When realizing the issue now it seems really a dangerous way to use the device.

But the usability just was not there for me. If we consider the fact that I should by profession be able to understand the risks and also have a better insight in using them what would average mobile users do?

I see the studies during this thesis most helpful to my understanding and to my career aspirations.

References

Android verified boot. (https://source.android.com/security/verifiedboot/). Last accessed 3rd October 2017.

Ansible. (https://www.ansible.com). Last accessed 3rd October 2017.

Apple Insider. (http://appleinsider.com/articles/17/09/25/macos-high-sierra-vulnerability-may-let-unsigned-apps-steal-keychain-logins-in-plaintext). Last accessed 27th September 2017.

Atlassian. Jira. (www.atlassian.com/software/jira). Last accessed 1st May 2018

Eleni Berki, Chetan Sharma Chandel, Yan Zhao, and Sunil Chaudhary. 2017. *A Comparative Study of Cyber-Security Knowledge in Higher Education Institutes of Five Countries.* University of Tampere and Deerwalk Institute of Technology.

Kristina Braz. 2011. *Integrating a Usable Security Protocol for User Authentication into the Requirements and Design Process.* PhD Dissertation. University of Quebec.

Andrew Butterfield and Gerard Ekembe Ngongi. 2016. *A Dictionary of Computer Science*. Oxford University Press.

Sunil Chaudhary. 2012. *Recognition of Phishing Attacks Utilizing Anomalies in Phishing Websites.* M. Sc. Thesis. School of Information Sciences, University of Tampere.

Catalin Cimpanu. *Researchers Devise 2fa System that Relies on Taking Photos of Ordinary Objects*. 2017. (https://www.bleepingcomputer.com/news/security/researchers-devise-2fa-system-that-relies-on-taking-photos-of-ordinary-objects/). Last accessed 31st October 2017.

Lorrie Cranor and Simson Garfinkel. 2005. *Security and Usability: Designing Secure Systems that People Can Use.* O'Reilly Publishing.

Deepnet security. (http://www.deepnetsecurity.com/authenticators/one-time-password/mobileid) Last accessed 8th December 2017.

Devopscube. (https://devopscube.com/wp-content/uploads/2016/06/ansible-aws-dynamic-inventory.png). Last accessed 5th October 2017.

Docker Toolbox. (https://www.docker.com/products/docker-toolbox) Last accessed 14th November 2017.

Gurpreet Dhillon, Tiago Oliveira, Santa Susarapu, and Mario Caldeira. 2016. *Deciding between information security and usability: Developing value based objectives*. Computers in Human Behaviour. Volume 61. Pages 656-666

FFIEC. Authentication in the Internet Banking Environment. (https://www.ffiec.gov/pdf/authentication_guidance.pdf). Last accessed 10th January 2018.

FiSTB. (http://www.fistb.fi/fi/etusivu). Last accessed 1st December 2017.

David Kim, Paul Dunphy, Pam Briggs, Jonathan Hook, John Nicholson, James Nicholson, and Patrick Olivier. 2010. *Multi-Touch Authentication on Tabletops*. Proceedings in SIGHI conference pages 1093-1102.

Forbes. How does apples new face id technology work. (https://www.forbes.com/sites/quora/2017/09/13/how-does-apples-new-face-id-technology-work/#8b64abc2b7f4) Last accessed 5th October 2017.

Getzephyr. (www.getzephyr.com). Last accessed 27th September 2017.

ISTQB. (http://www.istqb.org/downloads.html). Last accessed 13th November 2017.

ISO 9564-1:2002. (https://www.iso.org/standard/29374.html). Last accessed 13th November 2017.

Young-Hoo Jo, Seon-Yun Jeon, Jong-Hyyuk Im, and Mun-Kuy Lee. 2016. *Security Analysis and Improvement of Fingerprint Authentication for Smartphones*. Electronics and Telecommunications Research Institute of Daejeon and Inha University.

Lean Production. Kaizen. (https://www.leanproduction.com/kaizen.html). Last accessed 29th November 2017.

Kristin Loberg. 2004. *Identity Theft: How to Protect Your Name, Your Credit and Your Vital Information-- and What to Do When Someone Hijacks Any of The*se. Silver Lake Publishing.

Scott MacKenzie. 2013. *Human-Computer Interaction: an empirical research perspective.* Morgan-Kaufmann Publishing.

Thrushna Nalam. 2015. *RSS v2.0: Spamming, User Experience and Formalization.* M. Sc. Thesis. School of Information Sciences, University of Tampere.

Dinh Toan Nguyen. 2016. *Web Security: Security Methodology for Integrated Website using RESTful Web Services.* M. Sc. Thesis. School of Information Sciences, University of Tampere.

Donald A. Norman. 2013. *The design of everyday things.* Basic Book Publishing.

Joseph Ooi. 2015. *IMSI Catchers and Mobile Security.* EAS 499 Senior Capstone Thesis School of Engineering and Applied Science University of Pennsylvania.

Opera. (https://blogs.opera.com/security/2016/08/opera-server-breach-incident/). Last accessed 5th October 2017.

Owasp a. Fuzzing. (https://www.owasp.org/index.php/Fuzzing). Last accessed 29th September 2017.

Owasp b. (https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project). Last accessed 3rd January 2018.

William A. Perry. 2006. *Effective Methods for Software Testing: Includes Complete Guidelines, Checklists, and Templates.* Wiley Publishing.

RSA. (https://www.rsa.com/en-us/products/rsa-securid-suite). Last accessed 8th December 2017.

Angela Sasse and Ivan Flechais. 2005. *Usable Security: Why do we need it? How Do We Get It?* O'Reilly Publishing.

Software Testing Help. 37 Most Powerful Penetration Testing Tools (security testing tools). (http://www.softwaretestinghelp.com/penetration-testing-tools/). Last accessed 29th September 2017.

Stickyminds. (https://www.stickyminds.com/sites/default/files/article/2012/XDD12629imagelistfilename1.jpg) Last accessed 27th September 2017.

Awanthika Seranath, Nalin Arachchilage, and B.B. Gupta. 2016. *Security Strength Indicator in Fallback Authentication: Nudging Users for Better Answers in Secret Questions*. International Journal for Infonomics.

Russell Brandom. SMS two factor authentication hack. 2017. (https://www.theverge.com/2017/9/18/16328172/sms-two-factor-authentication-hack-password-bitcoin). Last accessed 3rd October 2017.

Ari Takanen, Jared D. Demott, and Charles Miller. 2008. *Fuzzing for software security testing and quality assurance.* Artech House.

Zulkarnain Syed Idrus, Estelle Cherrier, Christophe Rosenberger, and Jean-Jacques Schwartzmann. 2013. *A Review on Authentication Methods.* Australian Journal of Basic and Applied Sciences, 2013, 7 (5), pp.95-107.

Wikipedia a. Google Authenticator. (https://en.wikipedia.org/wiki/Google_Authenticator). Last accessed 5th May 2018.

Wikipedia b. Jenkins. (https://en.wikipedia.org/wiki/Jenkins_(software)#/media/File:Ansible-playbook-output-jenkins.png). Last accessed 28th September 2017.