

Security and usability in the mobile context

Robert Eriksson

June 7, 2007

Master's Thesis in Computing Science, 20 credits
Supervisor at CS-UmU: Thomas Nilsson
Examiner: Per Lindström

UMEÅ UNIVERSITY
DEPARTMENT OF COMPUTING SCIENCE
SE-901 87 UMEÅ
SWEDEN

Abstract

This paper describes the design and development process of an application that enables mobile access to features in a administrative portal developed by WM-data in Umeå, Sweden. For this purpose a Java Micro Edition application was developed that allowed management of Windows Workflow Foundation (WWF) workflows, as well as password retrieval. As WM-data has a diverse group of customers, it is important that the application is both secure and usable. Therefore the thesis starts by covering these two concepts in detail, discussing attributes of a secure application such as confidentiality, integrity and availability as well as threats that face mobile applications and what usability criteria must be taken into consideration.

That theoretical framework is later used to discuss the design choices made and to describe the development process and the four alternative solutions considered. The result is an application providing transport layer security to protect the transmission of the sensitive password retrieval and workflow management data, thus allowing these tasks to be performed in a secure way.

Contents

1	Introduction	1
1.1	Background	1
1.2	Problem and goals	2
1.3	Restrictions	3
1.4	Description of WM-data and AD-Tools	3
1.5	Thesis outline	4
2	Security In The Mobile Context	5
2.1	What is security	5
2.2	The Parkerian hexad	6
2.3	Security threats to mobile devices	8
3	Usability In The Mobile Context	13
3.1	Usability put into context	13
3.2	Usability goals	14
4	Design Considerations	17
4.1	Security issues	17

4.2	Considerations at the mobile device	21
5	The Development Process	25
5.1	Alternative 1: Windows Communication Foundation	25
5.2	Alternative 2: Public Key Infrastructure	27
5.3	Alternative 3: Encrypted identity token	28
5.4	Alternative 4: Unencrypted identity token over TLS	30
6	AD Mobile Tools	31
6.1	System overview	31
6.2	Restrictions and limitations	32
7	Conclusions	35
7.1	Conclusions	35
7.2	Thoughts on the project	35
7.3	Future work	36
8	Acknowledgements	37
A	Relevant Platforms and Frameworks	41
A.1	Active Directory	41
A.2	Transport layer versus message layer security	41
A.3	Transport Layer Security	42
A.4	Public Key Infrastructure	43
A.5	Java Platform, Micro Edition	43
A.6	.NET Framework 3.0	44

A.7 Web Services Interoperability Technologies	45
B AD Mobile Tools Flow Chart	47

List of Figures

1.1	The costs of security related incidents in the USA 2005, divided on security categories[4]	1
1.2	An example screenshot of the usage of AD-Tools in a web browser . . .	4
2.1	A categorical division of the security threats that apply to mobile applications and some of their solutions[4]	8
3.1	Nielsen's attributes of system acceptability, with the usability criteria put into context	13
4.1	System overview	17
4.2	Security issues affecting the system	18
4.3	Security threats affecting the different parts of the system	20
5.1	Conceptual drawing of the WCF security solution	25
5.2	Conceptual drawing of the PKI security solution	27
5.3	Conceptual drawing of the purely encryption based security solution . .	29
5.4	Conceptual drawing of the unencrypted solution	30
6.1	AD Mobile Tools	31
6.2	Workflow approval, information and password retrieval in AD Mobile Tools	32

A.1	Transport layer security	42
A.2	Message layer security	42
A.3	.NET Framework 3.0 Architecture	44
B.1	AD Mobile Tools Flow Chart	47

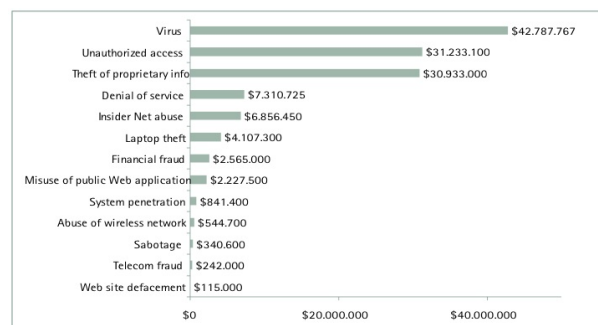
Chapter 1

Introduction

WM-data wishes to enable mobile access to their application AD-Tools (refer to section 1.4). In this chapter this problem is described and the goals of my thesis are stated. A background to this current and important topic is given before WM-data is introduced and the environmental base of the project is shortly described.

1.1 Background

As the area of mobile applications have skyrocketed the last few years, more and more businesses discover the advantages of exporting their systems functionality to mobile platforms. The possibility to interact with and control business critical computer systems from mobile devices practically anywhere in the world has created a whole new set of possibilities for organizations, as well as new dangers.



Source: Computer Security Institute, CSI/FBI 2005 Computer Crime and Security Survey

Figure 1.1: The costs of security related incidents in the USA 2005, divided on security categories[4]

Figure 1.1 shows the estimated cost of security breaches in American companies during

2005. Clearly, security incidents are associated with high costs and organizations, small as well as large, should consider taking actions to minimize the security threats to their operations if they wish to remain profitable, which ultimately must be the end goal of every security related action [1, p.7].

To be able to fully take advantage of the possibilities of mobile applications, security as well as usability must be carefully considered. These two factors are often perceived as barriers to each other [8, p.2] and managers must often sacrifice one or the other based on their perceived importance. Optimal usability often results in a more insecure application as more system critical functionality is exported to the mobile platform. To find the optimal balance between security and usability, managers must weight hard measured factors such as user acceptance and learnability against performance gains, estimated profit and the possibility of security threats. Naturally, this is a complex task that many senior project leaders struggle with.

The aim of this thesis project is to investigate different options available for exporting existing functionality from a user management system to mobile platforms and implement the one with the best fit. While doing this, the above mentioned concerns regarding security and usability will be taken into account and reflected about from a firm theoretical framework. This will allow the reader to gain a understanding of the problems associated with the mobile application development process.

Many similar applications exist, among which the m-commerce¹, application category is a big part. Mobile poker clients, bank account handling applications and mobile cashier payment are other examples of applications where sensitive data must be securely transferred over a insecure connection.

1.2 Problem and goals

Previously, users of the Swedish company WM-data's intelligent portal system AD-Tools that had lost their password had to request a new one from the help desk, which meant tying up help desk resources for a task that they should be able to solve themselves. Furthermore, handling of simple workflow tasks such as approving of user access to groups had to be done from the AD-Tools application, which oftentimes could be out of reach for the busy managers designated to handle them.

The solution was to extend some of the functionality of WM-data's AD-Tools to mobile platforms. Not all parts of AD-Tools were suited or needed to extend onto mobile devices though. After studying the customer requirements two parts of the system that were appropriate to extend onto a mobile platform were identified:

- Password retrieval
- Workflow approval

¹M-commerce (mobile commerce) is the buying and selling of goods and services through wireless hand held devices such as cellular telephone and personal digital assistants (PDAs).

Password retrieval would allow the users that had lost their password to request a new one using only their mobile device, making it easier and faster to get back to work again, while at the same time freeing up help desk resources. Workflow approval would enable privileged users with the correct security level to approve or deny Active Directory requests directly from their cellular phone - useful for example for granting a user access to resources while on the road.

1.3 Restrictions

The major restrictions that affected this project was time, mobile security threats and the mobile interface. Time is always a constraint, and that's why the goal was restricted to implementing a smaller subset of AD-Tools functionality. As this was a side project to the development of AD-Tools, it was agreed that it was better to aim to implement fewer parts and get them to work than to aim to high and be left with an unfinished project. Also, the physical limits of the mobile interface put restraints on what kind of functionality could be made useful on the mobile devices. Taking usability into account is especially important when operating in a such a constrained environment as mobile platforms, as an application that no one uses can be extremely advanced without adding any value to the organization. The usability criteria meant that the user couldn't be required to authorize himself every time he used the application, and instead had to rely on using some unique token identification method to identify the mobile client. Furthermore the amount of information that could be displayed on the screen were a restriction that limited the amount of operations that could be performed on the workflows. Therefore the workflow operations were restrained to simple acceptance or denial functionality and the more advanced functionality that would require displaying more in depth information were left out.

1.4 Description of WM-data and AD-Tools

WM-data is part of LogicaCMG since the autumn of 2006 and is one of Swedens leading consultant companies within information technology solutions, with focus on integrating IT and business. WM-data exists throughout the whole Nordic region and also have offices in Estonia and Poland, for a total of about 9000 employees.

WM-data in Umeå is also developing their own applications besides their consultant services, among which AD-Tools is a web application built on the ASP NET platform. AD-Tools acts as an intelligent portal that connects systems, user information, roles and business rules in a way that makes administration easy. It gathers the functionality from different Microsoft components such as Active Directory, Identity Integration Server, Windows Workflow Foundation, BizTalk and Authorization Manager and ties them into a single identity portal that can be customized for different needs. AD-Tools is deployed in a number of organizations such as General Electric, Sweco, Landstinget Västernorrland and Krisberedskapsmyndigheten.

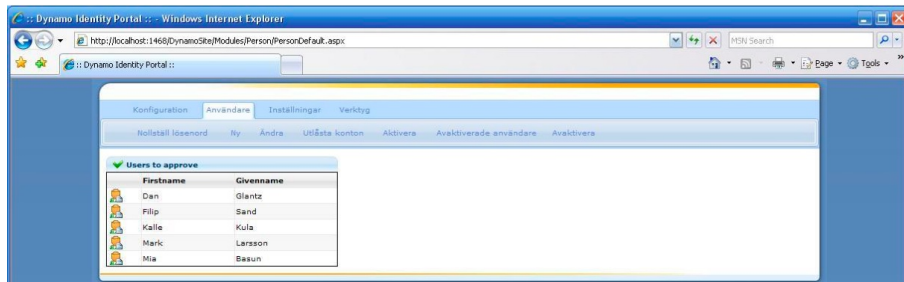


Figure 1.2: An example screenshot of the usage of AD-Tools in a web browser

1.5 Thesis outline

Chapter 2 gives the reader a theoretical framework on the security concepts necessary to consider when creating a secure mobile application. This chapter takes the reader from the most basic definitions of security through to more advanced ways of viewing the different aspects of security and the different kinds of threats that apply to mobile applications.

Chapter 3 provides a theoretical background on the usability criteria that needs to be considered when implementing a mobile application and also puts usability into a overall acceptability context.

In *Appendix A*, relevant platforms and frameworks mentioned throughout the paper is briefly explained for the benefit of readers unfamiliar with the topics.

The theoretical concepts from Chapter 2 and 3 lay the groundwork for the discussion of the design considerations in *Chapter 4*. Here an abstract view of the solution is provided and the security and usability aspects of the components are mentioned. Design choices made during development is argued for based on the security framework developed in Chapter 2 and the different kinds of threats affecting each part of the solution is discussed. The choice of platform for the mobile client is argued for and the usability goals stated in Chapter 3 are used to describe how the mobile interface was designed.

In *Chapter 5* the development process is described. Here the reader is led through the four different solutions considered and the pro's and con's of each are discussed.

Chapter 6 shows and discusses the final solution, and Chapter 7 ties everything together by presenting the conclusions of the thesis as well as some personal reflections.

Chapter 2

Security In The Mobile Context

In this chapter, a theoretical framework is laid for the later discussion of the security related design choices. The chapter starts with a discussion of different definitions of security and proceeds by presenting a model for describing the different aspects of security. Every aspect is exemplified and explained to give the reader a good understanding of the concepts before moving on to the discussions in Chapter 4. After the model, different kinds of security threats that affect mobile applications are mentioned. This provides another way of looking at security and also prepares the reader for the threat analysis in Chapter 4.

2.1 What is security

Anderson describes the practice of security engineering as being about building systems that “remain dependable in the face of malice, error, or mischance”. He also states that as a discipline, it focuses on “the tools, processes, and methods needed to design, implement, and test complete systems, and to adapt existing systems as their environment evolves” [15, p.3].

Olsson has a similar but more practical view of security. He suggests that security is mostly concerned with protection and safety, and gives examples of how security can be applied in different contexts such as physical protection of a house compared to providing safe transfer of data over a wireless network. He also argues that dependability is an important security aspect besides information security, as businesses and hospitals operations for example could be totally dependent on being able to use their networks [9, p.10].

Another, more abstract view of security is provided by Carlsson. His theory is that security is ultimately a matter of survival and concerned with solving problems that

could arise now or in the future. He compares this to the animal kingdom, where the strategy for survival could be described as eat, survive and reproduce. He argues that this concept can be applied to the human use of security in the current information society, and uses Darwinism as one of his arguments to why the human background as a purely biological creature compared to today's socially shaped individuals can explain the security issues that arise in the modern society. He also uses the concepts of Machiavellian intelligence, arms race, the tragedy of the commons and the red queen hypothesis as examples of models that explain why this is the case [2, p.19-31].

In summary security on the practical level be said to revolve around creating systems that provides protection and safety against malicious attackers in various forms by the use of security tools and practices. On a more abstract level, security can be viewed as one of our modern ways of guaranteeing our survival and a "natural" part of us and our society that has been inherited from our ancient ancestors.

2.2 The Parkerian hexad

This model was created by Donn B. Parker and extends the classic information assurance model CIA (Confidentiality, Integrity and Availability) with three additional attributes. The model is useful as it is *atomic* and *non-overlapping*, i.e. non of the attributes can be further broken down and every attribute refer to a unique aspect of information. Any information security breach can be described using one or more of these fundamental attributes of information [13].

2.2.1 Confidentiality

The confidentiality attribute refers to the risk of disclosure of information to an unauthorized entity [13], or the obligation to protect some other person's secrets if you know them [15, p.10]. Examples of confidentiality violations are when unauthorized people look over your shoulder as you type sensitive data into your computer, when an employee gives out confidential information over the phone, when data are sniffed on a corporate network by an intruder or when a laptop with sensitive and unprotected employment information is stolen from a organization. All these scenarios could possible allow unauthorized personnel to access un allowed information, and are therefor considered confidentiality risks [14, p.4]. One solution to keep information confidential is *encryption* that will be discussed more later.

2.2.2 Integrity

Within the context of information security, integrity means that data should be *correct* and *consistent*. That means that it can not be created, changed or deleted by unauthorized entities, and that data should be consistent throughout the whole area of its usage [13]. For example: an integrity loss occurs when a database becomes unsynchronized

after a sudden shut down and multiple instances of the same object exists. Integrity loss can also occur when employees accidentally deletes important files, a virus is released onto a computer or when a student is able to change his grade online instead of only viewing it. In the mobile context, integrity is mostly concerned with assuring that transmitted data has not been altered in the process. Tools for addressing this issue are *digital signatures* and *hash algorithms* [14, p.5-6].

2.2.3 Availability

Availability means that the accessed information should be available to authorized users when requested. This attribute is also concerned with the access rate of the information, and availability breaches in larger organizations were common a few years ago in the form of *Denial of Service* attacks [15, p.372] [13]. These attacks proved that keeping your information protected with regard to its confidentiality and integrity means little if it is unavailable to you. The solution to this kind of problem is achieved by using good network architectures and hardware without any single points of failure [14, p.6].

2.2.4 Control

Parker pointed out that there exists a distinction between confidentiality and control. If a corporate laptop is stolen, it results in a loss of control for the owner. But it doesn't necessary imply a loss of confidentiality as the thief may not be able to gain access to the stolen data due to techniques such as encryption, passwords etc. Therefore there is a need to distinguish between the information security attributes confidentiality and control, where control concerns only the physical control of the information without any need for disclosure [13]. Another example of a control breach is the presence of a rootkit¹ on a machine, where both the owner and the unauthorized entity has access to the machine.

2.2.5 Authenticity

Authenticity refers to correct labeling or attribution of information; the information should be both genuine and original [13]. The following example clarifies the difference from integrity and confidentiality. Imagine that a person fabricates an email address and decides to trick a friend by sending him an email from his boss telling him he is fired. This entails no breach of confidentiality as the person uses his own e-mail account the send the message. It also doesn't violate any integrity attributes as the e-mail message are sent exactly as intended by the person. What is being breached is authenticity: the e-mail is attributed to someone else than the real sender. Another example of a authenticity breach would be saving information in the wrong database field; e.g., storing a persons address where his phone number should be. Anderson gives another example: In a country whose banking laws state that checks are no longer valid

¹A rootkit is a set of software tools intended to conceal running processes, files or system data from the operating system.

after six months, a seven-month-old uncashed check would have integrity (assuming it has not been altered) but not authenticity as it no longer is valid [15, p.11].

2.2.6 Utility

The final attribute is utility: this means that the data should be useful [13]. If after securing your data by encrypting it you lose the key, the data could very well be confidential, controlled, integral, authentic and available - but without being useful to you. This concept is also dependent on the user of the data, as software code for example is of little use to an accountant but can have a great deal of utility to a programmer.

2.3 Security threats to mobile devices

A threat is a “potential violation of security” [14, p.6]. Regardless of whether the violation occurs or not, it still means that actions must be taken to prevent the violation.

Network Security	
Denial of Service Eavesdropping	VPN Wi-Fi Security PKI Firewall
User Security	
Identity Theft Hijacking Fraud Unauthorized Access	Fingerprint SmartCard Single Sign-On Pre-Boot Authentication
System Security	
Data Modification Data Theft Data Destruction Eavesdropping Espionage	AntiVirus AntiSpam Personal Firewall Data Encryption Trusted Platform Module Disabling of PnP Devices
Physical Security	
Device Theft Hardware Tampering	Theft Protection System Intrusion Detection

Figure 2.1: A categorical division of the security threats that apply to mobile applications and some of their solutions[4]

Figure 2.1 shows a model depicting various security threats to mobile devices, divided

in 4 different categories. Each category contains different kinds of threats and examples of solutions to them.

2.3.1 Network security

Network security refers to the process of protecting the traffic passing between the main system and the user with the mobile device [14, p.773-799]. This process is subject to two kinds of security threats - denial of service and eavesdropping [4].

Denial of service or DOS is when the attacker makes the service of the system unavailable to the mobile client, for example by overloading the server's or intermediate channels bandwidth or traffic limit. Unplugging the power cord of the server where the service is hosted or interfering the data sent on the wireless channel between server and client would also count as an denial of service attack [15, p.372] [2, 68-69].

Eavesdropping is the act of surreptitiously overhearing a private conversation. In the information security context, this means intercepting data sent between two communicating parties. A man-in-the-middle attack² can be an example of eavesdropping as can traffic analysis or traditional eavesdropping over a person's shoulder when he enters his password [15, p.45]

Protecting against network security attacks usually involves the following steps:

- Blocking bad traffic
- Encrypting own traffic

To hinder bad traffic from entering and flooding the communication channels firewalls are often used. This can lessen the success rate of DOS attacks to some degree and also provides an outer line of defense for system critical components. Encryption of the data sent between client and server makes eavesdropping harder and can be achieved through different technical solutions such as Virtual Private Networks (VPN) or a Public Key Infrastructure (PKI). Common for these kind of solutions are that they rely on public keys and strong mutual authentication to establish confidentiality (blocking packet sniffing and snooping), message integrity (blocking message alteration) and user authentication (blocking identity spoofing) [4].

2.3.2 User security

The user security concept is concerned with securing the system at the user level [14, p.845-868]. Previously, passwords were heavily relied on to identify users which resulted in writing them down on notes or storing them in text files. Today there exists a larger

²A man in the middle attack is one in which the attacker intercepts messages in a public key exchange and then retransmits them, substituting his own public key for the requested one, so that the two original parties still appear to be communicating with each other.

number of possibilities for securing the user end of the system, and the ways the user can identify himself are divided into the following categories:

- Something you know

- Something you have

- Something you are

Of course many of these can be combined to create a even safer form of authentication. *Something you know* is oftentimes a secret code, for example a domain password or the 4-digit code used to access your credit card [15, p.35-50]. This system by itself is not very secure and is not directly connected to a person but is still the easiest and most used identification method today [4]. Naturally something you know can easily also be learned by someone else and thereby jeopardizing the systems safety.

Something you have can be a key or a entry card to a building for example [15, p.277-304]. This kind of security mechanism isn't connected to a person either but is slightly more secure as a physical thing is somewhat harder to duplicate than a password [4].

Finally, *something you are* is the most secure way to identify users and often biometrics [15, p.261-276] are used to scan fingerprints or eye pattern for identification use. This system is physically connected to a person which practically prevents it from being lost or stolen [14, p.328-330].

In general, the more secure the system, the more expensive it gets. As with everything, security finally turns into a financial question and the cost has to be weighted against the perceived gain. Installing a fingerprint reader together with a entry card system to protect the company's restroom might be too much of an investment compared to the gain and would most likely result in the restroom never being used.

One of the threats to user security is identity theft [15, p.35-36]. As have been seen, this is a major risk when relying on the "something you know" kind of identification as it generally is the easiest kind of identification mechanism to get hold of for an attacker. The solution is to use or add more complex ways of identification to allow users to be uniquely identified. Hijacking [15, p.310] and fraud [15, p.394] are closely related to identity theft and involves impersonating legitimate users and using their identity to perform tasks.

Unauthorized access has a subtle difference from identity theft in that the perpetrator is not required to know/have/be whatever the identification demand requires. An example of a unauthorized access scenario is if a attacker gets hold of a workstation where a authorized user has forgotten to log out of the system. The intruder have now gained access to the system without knowing the user's password.

2.3.3 System security

System security is a complex topic and involves securing both the server and client side internals. Here encryption can secure the storage instead of the transfer of sensitive data and help prevent security threats such as espionage and data theft on lost or stolen mobile devices [14, p.805-840]. Using a Trusted Platform Module (TPM) is one example of how data can be securely stored on a micro chip in a phone or laptop. TPM:s generally offer support for the use of cryptographic keys and contains a unique RSA key that identifies the module. Data can also be sealed which enables data to only be decrypted in exactly the same state, i.e. by the same software and on the same device. Features such as this helps prevent data modification as well as eavesdropping, espionage and data theft.

Another similar problem is the growth of viruses, trojans and other malicious applications on mobile devices that previously only existed on non-mobile platforms [15, p.379-382] [2, 63-76] [14, p.613-626]. As the support for major standardized platforms such as Java ME and Windows CE continues to grow on mobile devices so does the number of vicious applications. This creates the need for precautions such as anti virus software to operate on the mobile device to make sure that 3rd party application doesn't access or alter important data.

2.3.4 Physical security

Physical security is concerned with the material security of the devices used. Mobile phones and other portable devices are especially vulnerable to physical theft due to their size and infer a security threat to organizations as more and more business critical information is handled in mobile devices. The solution to material theft often involves traditional physical ways of protecting information such as the use of different kinds of locks and material obstacles. Hardware tampering on the other hand occurs when the physical device is altered in some way to benefit the attacker, for example when a bug is implanted to tap a phone [4].

Chapter 3

Usability In The Mobile Context

This chapter discusses the important aspects of usability to consider when designing a mobile application. As a mobile device differs from the standard environments where most applications are usually deployed, extra care must be taken to ensure that bad usability isn't a reason why a otherwise well built mobile application is never used. This is done by taking the different aspects of usability discussed below into consideration while at the same time viewing usability as a piece of the larger puzzle.

3.1 Usability put into context

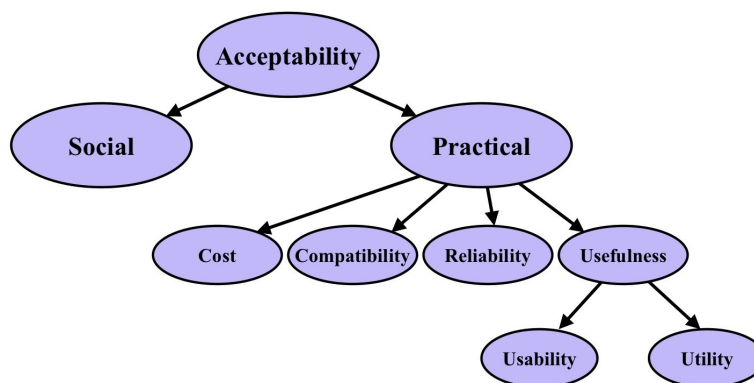


Figure 3.1: Nielsen's attributes of system acceptability, with the usability criteria put into context

Figure 3.1 depicts Nielsen's view of usability in a greater context. Here Nielsen clearly

differentiates between the concept of usability and utility, whereas Preece et. al have included utility as a central part of their definition of usability. Nielsen did stress the importance of balancing usability against utility though, and Figure 3.1 shows the other aspects of acceptability that he finds important to consider. Even though Nielsen's and Preece's definitions of usability differs somewhat, it is interesting to view usability when put into a larger context as it allows us a feeling of what other aspects to consider when designing an application that will be accepted by its users. The figure shows that costs, compatibility and reliability are other aspects of a user's experience that affect the usability. A perfect usable application may be too costly to implement, incompatible with important 3rd party software or too unreliable to be released. Also, besides the practical aspects, the need exists to examine if the software will be socially acceptable to the users. This can be a problem if cultural differences are not taken into account, as well as if the end-users social values are misjudged.

3.2 Usability goals

Preece et. al defines usability as a mean to ensure that “interactive products are easy to learn, effective to use, and enjoyable from the user's perspective” [12, p.14]. They proceed by breaking down the usability concept into the following “usability goals” described below:

3.2.1 Effectiveness

For a product to be defined as effective, its users must accomplish their tasks accurately and completely. Preece et. al. argues that this goal refers to “how good a system is at doing what it is supposed to do” [12, p.14]. Examples of good effectiveness are if users of a web site can find correct and complete information about the subject they need and if a user of a mobile weather service can view a weather forecast on his cell phone.

3.2.2 Efficiency

Efficiency refers to the way a system supports its users in carrying out their tasks. The principle here is that a minimal number of steps should be used to carry out a task to enable users to quickly get their work done [12, p.14]. Efficiency can be measured both in terms of number of steps used or the time spent on a task. A interesting thing to note is that what is important is the user's *experienced* efficiency, i.e. how fast the user thinks that the product operates compared to the actual efficiency. An example of efficient systems is many of the currently available e-commerce platforms. They allow the customers to save their personal details when they make a purchase, thus allowing them to complete their following orders faster with only a few clicks.

3.2.3 Safety

Safety is a wide concept and involves protecting users from “dangerous conditions and undesirable situations” [12, p.14-16]. This could refer to physical external conditions like when an X-ray machine is being operated, or simply helping people to avoid carrying out unwanted actions accidentally. An example of the latter is not placing the save and quit commands next to each other in the menu of an application. The purpose of a safe system is to make its users confident and allow them try out different features without risk of losing work or information. Other examples of safety concepts are undo mechanisms and confirmation dialogs when the user has requested to perform a operation that possible could result in negative consequences. Other safety issues refers to user privacy and information security, for example what happens when unauthorized individuals get access to the system or product?

3.2.4 Utility

Utility refers to the amount of functionality a system provides. If a system provides the right kind of functionality so that users can perform their tasks it is said to have high utility [12, p.16]. An example of a system with a high degree of utility is Matlab that provides its users with huge amounts of mathematical tools. Utility must be carefully balanced against efficiency otherwise usability is affected. This is because a system or product cannot be overwhelmed with rarely used features without making it harder to use for the standard user.

3.2.5 Learnability

A systems learnability refers to how easy it is to learn to use the technology. Users expect products to be easy to operate and do not like to spend a long time learning to use a system [12, p.16-17]. You can have developed the most functional and innovative product in the world, but if your users can not motivate themselves to take the time to learn how to work with it, no one is going to use it. A bicycle is a good example of a product that is easy to learn to operate.

3.2.6 Memorability

Making a system memorable means to make it easy for the users to remember how to use it. This is especially important for systems that are infrequently used, as the goal is to avoid having the users relearn the task each time. Examples of tools to make a system memorable is good use of icons, command names and menu options, and also structuring options into relevant categories (e.g. dividing Microsoft Office’s functionality into Word, Excel, Powerpoint and Access applications) [12, p.17]. A bicycle is once again a good example of a memorable product as it is very hard to forget how to ride a bicycle once you have learned to use one.

Chapter 4

Design Considerations

This chapter outlines the design choices made before and during the development process. The theoretical framework laid in Chapter 2 and Chapter 3 is used to argue for the choices made and to discuss the design issues related to developing a secure and usable mobile application.



Figure 4.1: System overview

Figure 4.1 shows a conceptual drawing of the basic components in the solution. AD-Tools handles among other things the communication with Microsoft Active Directory and needs to implement some kind of service that provides a mobile client with the ability to use some of the AD-Tools functionality. This service must be provided over a data transport channel in a secure way, which presents a few design issues to consider.

4.1 Security issues

Here the security issues facing developers when implementing this kind of solution are analyzed and discussed. The kind of security features that should be developed are also considered.

4.1.1 Security concerns regarding the Parkerian hexad

Figure 4.2 shows the security issues affecting the system divided on the attributes in the Parkerian Hexad discussed in section 2.2.

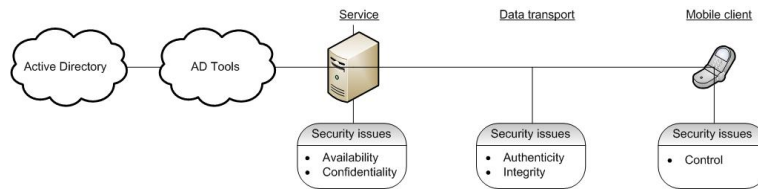


Figure 4.2: Security issues affecting the system

The *service part* of the system is affected by both the availability and confidentiality criteria. The *availability* attribute affects the service as the client needs to be able to access the service and the AD-Tools data at any given moment, and thus actions must be taken to ensure that the service is accessible. This is a hard thing to do though, as a number of factors can affect the ability to communicate between the service and the mobile device. The mobile device is often dependent on the mobile communication infrastructure available for its Internet connection to function, or otherwise connects through some kind of bluetooth connection or similar link that are uncontrollable. Other sources of disturbance includes, but are not restricted to, electromagnetic interruptions, physical obstacles and material defects in the physical path connecting the device to the Internet. Obviously, these are factors that are outside of our control, and instead focus must be on the ones that can be affected, which is the availability of the server on which the service is hosted and the reliability of the mobile client. As there is not a lot to do to improve the security and operation of WM-data's machine park, the only way the availability of the solution can be affected is to make sure that the mobile client runs smoothly and is able to operate on different kinds of mobile devices. To enable this support the application has been tested on as many kinds of devices that was possible and under some different circumstances to make sure that the client at least does not suffer from availability issues.

To take the *confidentiality* attribute into regard means that it must be made sure that the sensitive data the system handles is not disclosed to unauthorized entities. One way to protect the data is through the use of cryptography on the service and client sides. Here the messages are encrypted and transferred over the insecure medium and the strength of the encryption is relied on to make sure the data is inaccessible to potential intruders[15, p.73-113]. Another solution that incorporates cryptography to prevent eavesdropping, data modification and message forgery is TLS/SSL[15, p.398-399]. Transport Layer Security can either be based on server authentication, where the server is authenticated to the clients by the use of a certificate while the clients remains unauthenticated, or mutual authentication, where a public key infrastructure (PKI) is used to ensure the identity of both server and clients. The benefits of TLS includes authentication of server and possible clients and protection against Man-in-the-middle attacks, but requires the clients to trust the server based solely on the server name and some additional data which sometimes can be a problem when the end users themselves need to decide whether to trust the server or not. The use of TLS incurs some speed penalties as some additional overhead is sent along with each package, but as the application is neither speed dependent or overly traffic limited as opposed for example a real-time computer game, TLS or similar technologies is a good choice to protect the confidentiality of the sensitive data.

The *data transport* channel on the other hand is affected by the authenticity and integrity attributes. To consider *authenticity* in the solution, it must be made sure that the data sent over the transport channel is attributed to the correct sender so that an attacker is not able to impersonate a system user and retrieve his password or handle workflows in his name. Mutual TLS authentication, unique user identification tokens such as mobile IMEI numbers¹ or similar identification techniques can be used to ensure that attackers doesn't create their own requests and impersonates legitimate users.

Integrity on the data transport channel means that data and requests sent either from the client or service should not be able to be altered by 3rd parties without notice. This attribute can also be taken care of by the use of techniques that uses digital signatures to provide message integrity, such as TLS or Microsoft Windows Communication Foundation. This ensures that messages can not be tampered with without notice and solves the integrity issue.

The mobile client is mostly concerned with the *control* attribute. If the user loses his physical control over the mobile device, this should not result in a security breach. Here a choice exists between security (e.g. forcing the user to enter a password for identification) or usability (being able to run the application without entering a password every time) when designing the solution. Security wise the control issue would be resolvable with the use of some identification technique such as forcing the user to enter a password every time the application is started on the mobile device. This way, even if the control of the device is lost to an external part, no harm may occur to the system if the attacker has not also been able to gain access to the user's password. This approach could be more acceptable from a usability perspective if this was an application that was used only under special circumstances. But as the users should be able to manage AD-Tools workflows on a regular basis, the constraint to always enter a password would be too time consuming to be considered efficient [12, p.14]. An alternative but less secure solution is to create mechanisms to allow the users to lock the access to their accounts when they lose their mobile devices and reset their password. This works better when the user has simply lost his device than when it has been stolen, as the perpetrator most likely will have gained access to either the user's password or managed to approve/deny workflows in the user's name. Assigning correct and limited rights to each user is an important factor determining the safety of the system in cases such as this.

4.1.2 Security threats

Another way of viewing the necessary design choices security wise is to look at the system from a security threat perspective.

Figure 4.3 shows which kind of security threats that affects the parts of the solution. The *service* part is mostly affected by user security threats, as users must be authenticated and data sending restricted to real users. System and physical security threats are also applicable to some extent, but as there isn't much to do to secure the system where the service is run these kind of threats will be omitted from the discussion. It will be

¹The International Mobile Equipment Identity is a number unique to every GSM and UMTS mobile phone and is used by the GSM network to identify valid devices. The IMEI is only used to identify the device, and has no permanent or semi-permanent relation to the subscriber.

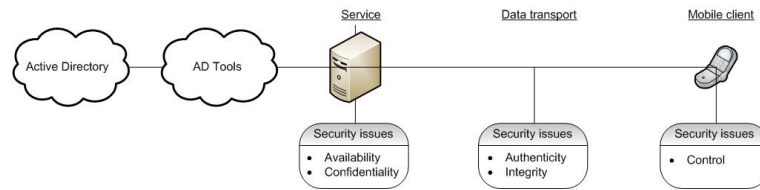


Figure 4.3: Security threats affecting the different parts of the system

assumed that the server environment where the service is hosted is adequate secured by WM-data already. To protect the system against user security threats, the system will use the “something you know” identification category to determine if it is dealing with an authorized user. Although this can be argued to be the weakest kind of identification mechanism, it is the most efficient and the only kind that is applicable in this kind of environment. As discussed earlier, this can be accomplished by the use of certificates, unique tokens or similar identification mechanisms.

The *data transport* channel is subject to network security threats, which means it must be protected from denial of service attacks and eavesdropping as previously discussed. The use of a firewall can reduce the exposure to DOS-attacks as data transfers can be restricted to certain ports and only allowed if it fulfills certain criteria. This is far from a fool-proof solution of any kind and must be used together with other precautions such as always using the latest communication protocols. This kind of infrastructure protection is already implemented at WM-data and therefore this specific kind of threat is unessential to protect the system against. And, as mentioned earlier, cryptography can be used to protect the data from eavesdropping by malicious intruders.

The *mobile client* is the part of the system that is subject to the most categories of security threats. First, it could potentially fall subject to user threats as a person that gains access to the mobile device has full access to the owners user privileges due to the usability criteria that do not allow the system to force the user to enter a password to gain access to the mobile application. Secondly, another danger is system threats, as applications can be installed on the mobile device that interrupts or store the data transmitted from the service. The final, and in most likely security threat, concerns the physical security of the mobile device. If the device is misplaced or lost, this automatically incurs users threats to the system and requires the owner to quickly report the loss to an authorized person that can lock that users system access and release the connection between that user’s account and their mobile device unique identifier (a IMEI number, phone number or something similar).

4.1.3 Transport layer versus message layer security

Does the need exists to implement transport layer security, message layer security or both? As mentioned in Appendix A.2, there are penalties associated with adding to much security features, but are they a concern or not?

The application will most likely not suffer from any major restrictions regarding how much data it is allowed to transfer. Even though prices are higher and bandwidths lower for mobile devices, the extra overhead should not be much of a problem. Time wise these security features probably will affect the performance of the application to a small extent as the CPU power of most mobile devices are rather limited, but it should not affect the user's experience so much that they will consider it lacking from a usability perspective either. So most likely any worries about the penalties associated with the security features can be disregarded, especially considering the large amount of similar secure applications that exists on the market, e.g. bank account management applications and poker clients.

It has already been discussed that for cases when the client communicates with the server directly without any intermediaries, a transport layer solution performs equally well to a message layer solution security wise. Now, given that adding both transport layer security and message layer security to the solution probably would not affect its performance negatively, is there any reason to discard the idea of implementing message security and only focusing on transport layer security?

That would be the case if it could be argued that the solution would only send data directly from client to server without any intermediaries involved. This, although it most likely most often will be the case, is not something that can be guaranteed as each of WM-data's customers will have their own network configuration and use the application in their own environment.

So to account for every different alternative, ideally both transport layer and message layer security should be implemented, even though transport layer security will suffice most of the times.

4.2 Considerations at the mobile device

In this section is discussed what had to be considered before implementing the most vulnerable part of the system, the application for the mobile device.

4.2.1 Platform

The first thing to consider is what platform the mobile application should run on. It must be taken into account that the service is predetermined to be hosted in a .NET environment due to the AD-Tools platform, and the advantages of a fully compatible solution involving .NET Compact Framework (CF) on the client side must be weighted to using the more widely supported Java2 Micro Edition (J2ME) platform.

After doing some research and having had a talk with the AD-Tools development team it was decided to implement the client application in J2ME as the support for .NET CF was close to nonexistent among the mobile devices used in WM-data. The goal was also for their customers to be able to use the application on a as wide range of devices as

possible. This choice was a major one, as it set the framework for which additional pieces of software are going to need to be included in the solution. This decided what kind of user interface that would have to be implemented as well as what kind of communication would be used between the service and the client.

J2ME comes with API's for developing a graphical user interface (GUI) and it turned out to be the one used to create the user interface for this application even though other choices exist. One worth mentioning is J2ME-Polish, an extensive and feature-filled graphical library that can be used together with J2ME. It contains a lot of extra graphical objects and possibilities to create a more advanced kind of GUI than the standard J2ME API's allow, but it also costs money in license fees and therefore it was abandoned.

4.2.2 Interface

As the application from the start would support both password retrieval and workflow management, it needed to incorporate both these features into the interface, while at the same time adhering to the usability goals discussed in Section 3.2.

Hopefully, a user that has the application installed does not lose his password that many times compared to how often he manages his AD-Tools workflows, so it makes sense to give more room to the workflow management than to the password retrieval part of the application. To present the active workflow requests immediately when the application starts would be one way to increase the *efficiency* of the application, as the user most often wishes to handle the requests and can take care of that directly. The times when the user needs to change his password, this can instead be accomplished by navigating through a menu, see Figure 6.2.

To consider the *safety* aspect of the usability goals, the application was made sure to implement confirmation dialogs whenever the user requests a new password or accepts or denies a workflow request. The placement of the accept and deny options in each screen and dialog is also consistent as not to confuse the user and allow him or her to choose the wrong command by habit. This goal of this consistency is to make the users confident in using the application. See Figure 6.2 for an example of how this can manifest itself.

To create a high degree of *learnability*, the application was made as small and easy to use as possible. As the extent of the application was rather limited, this was never a usability attribute that affected the development process to any high degree.

The same goes for the *memorability* criteria to some degree as the application's small size makes it very comprehensible and easy to remember how to use. The only problem is the password retrieval part that can be assumed will not be used as frequently as the workflow management part. Here it is important to be clear about what the user is supposed to do and make the process as easy as possible. As it was decided to omit identification of the user at the client side, the password request procedure is very simple and only requires the user to confirm his new password request.

The application has a moderate high amount of *utility* if seen from the perspective of a workflow approval program. The password retrieval functionality is useful in itself but will more seldom be used and is more like an extra feature in the coherency. It is also *effective* as it is able to carry out the designated tasks (password retrieval and workflow approval) in a satisfactory way.

Chapter 5

The Development Process

During the development process, different kinds of technical solutions have been tested to fulfill the design criteria discussed in Chapter 4. Four different implementation alternatives were considered and tested thoroughly in WM-data's development environment and in this chapter the pro's and con's of each, together with my own experience of how the different parts involved worked together in the solution is discussed.

5.1 Alternative 1: Windows Communication Foundation

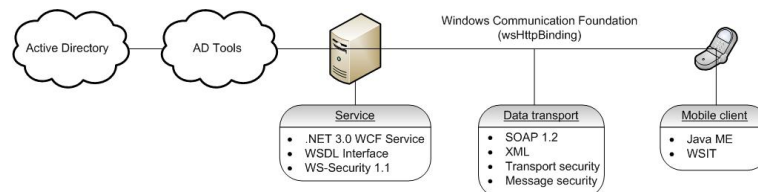


Figure 5.1: Conceptual drawing of the WCF security solution

Figure 5.1 shows the components involved in each part of the solution that relies on Windows Communication Foundation (See Appendix A.6.1) to provide the needed security. Basically, the idea is to utilize Microsoft's WCF service interface shipped with .NET 3.0 (See Appendix A.6) to enable a secure connection that provides transport and message security to the mobile device through the use of the wsHttpBinding. This binding is based on a traditional HTTP connection, but with the added benefits of transactions, reliable messaging and support for various WS-standard security solutions besides the basicHttpBinding's message security support.

Practically, this will be solved by hosting a service on WM-data's network that has

access to the Internet and allowing the mobile clients to communicate with it through secure HTTP connections by passing XML data embedded in SOAP messages¹. The `wsHttpBinding` uses the SOAP 1.2 protocol to exchange these XML-based messages and is a secure way of communicating between devices that support the standard. Using this binding in the security mode “`TransportWithMessageCredentials`” we can use both transport layer and message layer security to protect the data. To also use message layer security has the benefit of providing extra system security as data modification, data theft and eavesdropping gets harder for an attacker to do while the package is not being transported.

The solution provides *confidentiality* as the message or transport security mechanisms provided by the WCF binding encrypts the data when it is sent between the client and service, thus making it harder for an attacker to intercept and read the data, therefore preventing eavesdropping. WCF is also able to provide *authenticity* through the use of username, Windows account or certificate credentials that identifies users that access the service. Client authentication can be achieved either through client certificates, stored user name credentials or a unique identity token such as the mobile phone’s IMEI number. This sort of mechanisms provides user security and prevents identity theft, hijacking and fraud. TLS (See Appendix A.3) or WCF’s message security mechanisms also uses signing and digests to protect the data against tampering, thus providing *integrity* to the solution as well.

The problem is to enable the communication to flow between the .NET WCF service and the Java client as their web service implementations differs in a multitude of ways [11]. Although each kind of web service can be described by the use of the Web Service Description Language (WSDL), their inherited structural differences make them incompatible with each other. A Java web service client cannot understand the format of a .NET web service interface and vice versa.

To overcome this problem, a common denominator needs to be found that allows these two platforms to communicate with each other on the same terms. This denominator is the *Web Service Interoperability Technology*, also known as WSIT or Project Tango, see Appendix A.7. At the time when the work on this thesis started, this technology was relatively new (the first WSIT release dates back to late august 2006, 5 months before the thesis work started) and therefore largely undocumented and unsupported. Obviously, people had experienced some success with making the technologies interoperable, but some concerns were voiced on the WSIT forum about problems implementing some of the security features.

At the start of the project when WSIT first was used together with the Netbeans Java IDE, the most current version available was WSIT Milestone 2. Using a J2ME (See Appendix A.5) web service client, the Sun Java Wireless Toolkit was unable to generate any client-side stubs when using any kind of web service binding that required the use of SOAP 1.2, such as the before mentioned secure `wsHttpBinding`. When a standard HTTP based binding such as the WCF `basicHttpBinding`, that uses SOAP 1.1, was used everything worked correctly. After working with this for a long time while reading the few post available from people with similar problems, this solution finally had to

¹SOAP is a protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS.

be abandoned as time was running out and there seemed to be no way to make this solution work.

Instead focus was laid on the second approach described below where the security functionality provided by WCF was implemented by hand.

5.2 Alternative 2: Public Key Infrastructure

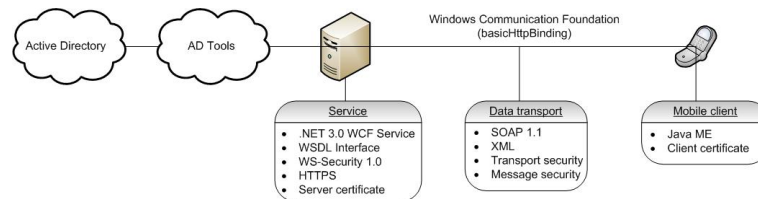


Figure 5.2: Conceptual drawing of the PKI security solution

Figure 5.2 shows the second solution, where a public key infrastructure (See Appendix A.4) is relied on to provide a secure way of transferring data between a Java based mobile client and the .NET WCF Service.

Instead of relying on one of WCF's secure bindings, certificates are used on both the client and server side to allow for authentication and reliable message transfers. WCF's `basicHttpBinding` is used to communicate between the service and the mobile client. This binding works much better together with J2ME as it uses SOAP 1.1 packages and a WSDL format that is compatible with Sun's format. It only follows the most basic security specifications though, so to enable transport layer security server and client certificates must be used to secure the communication channel. This creates a high degree of user security when implemented correctly as user certificates are hard to forge, and provides a fair degree of network security as well due to the encryption strength of TLS. It does not take any physical security precautions though as all of these software solutions, so if an attacker would get hold of the mobile device the certificate and through it the user's identity would be in his hands.

This solutions also takes care of the *confidentiality*, *authenticity* and *integrity* issues mentioned in Chapter 4 through the use of TLS at the transport layer. Certificates are used for authentication purposes, where the client has its own certificate the uniquely identifies it to the service. This solution does not support message layer security, which makes it more vulnerable to system security threats such as data modification and eavesdropping at intermediate servers between the client and service.

Installing a certificate on the server side is a simple matter. As WM-data hosts AD-Tools in Internet Information Services (IIS) 6.0, certificates can either be self-signed and generated through Microsoft Certificate Server or bought from a trusted Certificate Authority (CA) to lend credibility to the cause. The latter way is more suited for applications where the users simply do not know if they should trust the application, for

example if the application is accessed through the Internet. If the certificate is bought from a well-known CA, most browsers and operating systems already trust that CA and now in turn trust the buyer. These certificates cost around 1000\$ per certificate and year, so for applications that mostly will be used “in house” or by persons belonging to the same organization, self-signed certificates can be a viable alternative.

Either way, after a server certificate has been created or issued, a corresponding client certificate must be generated and the chosen CA (either an established CA or the machine used to generate the self-signed certificate) must be trusted by each of the clients. If a regular operating system was used to run the Java application communicating with the service, the IIS generated service certificate would have to be installed on the client machine to allow local applications to trust the remote site. As the application deployed is a *MIDlet*, a Java program for the Java ME virtual machine, aside from installing the self-signed certificate on the mobile device, the MIDlet also have to be signed using a client certificate generated from the IIS certificate. The security level of the MIDlet also need to be set before installing it to the mobile device for it to be able to communicate securely with the WCF service.

MIDlets can be signed using the Sun Java Wireless Toolkit (WTK) or through the command line tool jarsigner. To be able to test the application during the development process, the certificate needs to be added to a Java keystore that has been added to the Netbeans project. This way, the MIDlet can be run against the server using the WTK emulator and recognize it as a trusted source.

The problem encountered here was that it was hard to get certificates to work together with J2ME. This solution needs client certificates that is based on the certificate used on the server side, in this case IIS 6.0. It turned out that it was not easy to get these two kinds of certificates compatible with each other, forcing the use of third party software to convert them. Another problem was that self generated certificates, the type the solution aimed to use, was not supported by the Netbeans IDE which demanded that the issuer of the certificate should be consider trusted for the application to run at all, i.e. there is no option of using the application over a insecure channel.

This halted the development process and forced the use of temporary request trial certificates from Verisign that could be use to secure the communication.

5.3 Alternative 3: Encrypted identity token

Figure 5.3 shows the third solution. This solution discards client certificates as a means for authentication and instead relies on the International Mobile Equipment Identity (IMEI), a unique number available on every mobile phone, to provide client identification. A certificate on the server side allows us to communicate through HTTPS, thus providing transport layer security, and by encrypting and decrypting the sent data manually message layer security is provided as well. Providing encryption manually is a good way to add some system security to the solution as discussed earlier.

In this solution, the unique IMEI number is retrieved for each client that wishes to

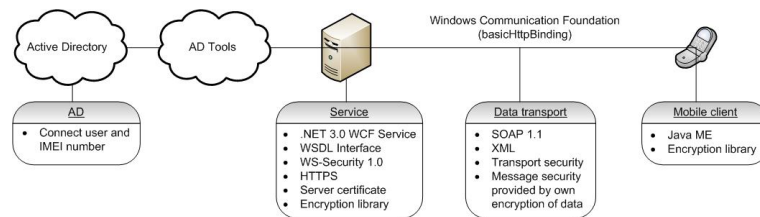


Figure 5.3: Conceptual drawing of the purely encryption based security solution

connect to the service and the number and data is sent together encrypted to prove the clients identity. This requires that a connection exists between each valid user and his or her IMEI number in the Active Directory so that the service can translate the IMEI number it receives into the correct user account. There is no standardized way to retrieve this number from all mobile phone vendors though, or even a guarantee that it will work on every type of phone. Retrieving the IMEI number may work on one kind of mobile phone from a vendor but not on another kind from the same company, so by identifying clients this way support for some mobile phones is neglected, and especially older ones.

Server authentication is performed by TLS, as well as confidentiality and integrity. This solution allows security on the transport layer using the WCF `basicHttpBinding` and TLS which works well together with J2ME, but to also enable security on the message layer with correct encryption or decryption of the sent data that is needed to let the client and server agree on a common cryptography algorithm. Microsoft provides many encryption algorithms in their .NET framework, but although this is supported in Java 2 Standard Edition it is not implemented in J2ME. This forces the use of third party encryption libraries such as BouncyCastle² to enable message security.

Adding the BouncyCastle API to a J2ME project is easy as the files necessary for the chosen cryptography algorithm is simply extracted and added to the project. To get the J2ME client to encrypt and decrypt the data in a way that the .NET service can understand turned out to be harder though. Microsoft supports many of the most common crypto algorithms in their .NET API, so in theory if one is chosen which exists in both the BouncyCastle and .NET API messages should be able to be both encrypted and decrypted. But as the cryptography engines are implemented differently, attention needs to be paid to things such as key length, block size and types of padding depending on the algorithm and platform chosen.

Even though these parameters were specified, it was hard to get the service and the client to decrypt a encrypted string back to its original form. After searching for a solution it was decided to try out the BouncyCastle C# API on the service side. The API that was still in beta testing at the time, but it was decided to use it to ensure that both sides at least used the same cryptography engine. This however did not help, and it was suspected that the problem could have something to do with the different string representations used by .NET and Java. Converting the text to different formats such as ASCII, Unicode and UTF-8 before sending them as bytes and converting them back

²<http://www.bouncycastle.org>

again when they arrived did not work out however, which set the project back to where it started.

Finally, as the time was running out this solution was abandoned and instead focus was turned to a simpler way of doing things that was bound to work and at least create a basic template that could be made more secure later on.

5.4 Alternative 4: Unencrypted identity token over TLS

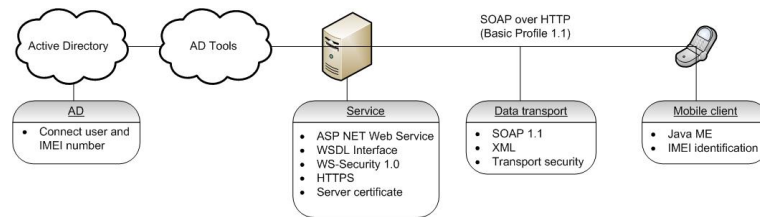


Figure 5.4: Conceptual drawing of the unencrypted solution

Figure 5.4 shows the final solution. This one relies on the same authentication mechanism as previously, but omits encrypting the data and IMEI number sent and relies on TLS's encryption features. This makes the solution somewhat less secure as it only implements transport layer security through the use of HTTPS and not message layer security. This could make it a target for some of the threats discussed in section 2.3, especially the system related ones. The protection against network and user security threats are still good though as they rely on the same kind of safety mechanisms as previously. This means that authentication is still performed, and confidentiality and integrity is still taken into regard by the use of TLS. By now, the service had been moved and implemented as a ASP NET Web Service, meaning that the WCF interface has gone out of the picture.

Chapter 6

AD Mobile Tools

This chapter describes the final result of the thesis work, a mobile application that allows some of the previously described functionality of AD-Tools to be accessed from a mobile device in a secure and usable way. As the development process already has been discussed in Chapter 5, this chapter will focus on providing a brief overview of the result together with pictures that illustrate the behavior of the system. The results will also be discussed with regards to the initial goals stated in Chapter 1

6.1 System overview

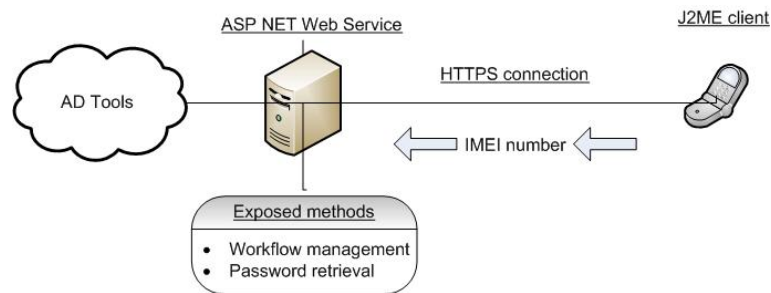


Figure 6.1: AD Mobile Tools

As can be seen in Figure 6.1 a subset of AD-Tools functionality is exposed through a ASP NET web service. This web service can then be accessed from mobile devices given that they can provide a correct IMEI number, a number that uniquely identifies every mobile phone. Every user supposed to use the system has previously been connected with his IMEI number in Active Directory, thus allowing AD-Tools to see which user that is requesting information.

The web service is accessed from Java Micro Edition clients, which is able to approve and deny workflows as well as retrieve a new AD-Tools password if they have lost theirs. Figure B.1 in Appendix B shows a flow chart describing how the different parts of the application is connected.



Figure 6.2: Workflow approval, information and password retrieval in AD Mobile Tools

Figure 6.2 shows the screen that greets the user when he enter AD Mobile Tools. Here he is allowed to manage new and previously unapproved workflows by either approving or denying them. A short description of the workflows is provided but the user can also view a more informative screen as seen in the picture in the middle.

If the user has lost his AD-Tools password, he can retrieve a new one as seen in Figure 6.2 in the picture to the right. Here the user has to confirm his choice and is then sent a new, randomly generated password that can be used to access the system again.

It should be noted that these screen shots are taken from the J2ME Wireless Toolkit emulator and may differ from how the application actually looks like when it is deployed on a real mobile device. This is because each vendor choose how to implement Java support for their specific phone model, which may result in differences in placement of objects, menu options etc.

6.2 Restrictions and limitations

The application turned out a little bit different in terms of security than what was originally planned and hoped. Instead of the full featured transport and message security solution hoped for the message security of the data transport mechanism was forced to be abandoned and focus was set on getting a simpler solution working. Time has been

a issue during this project and the more problems have been encountered the more time has ran away. Each time a secure way of communicating has not been able to be implemented the project have had to start over from the beginning. As the focus has been on implementing a secure transport mechanism the mobile client GUI suffered as the development of it was put further and further off.

Chapter 7

Conclusions

In this chapter the thesis work is wrapped up. Future possible work and the current situation at WM-data is mentioned along with overall thoughts about the project as a whole.

7.1 Conclusions

The mobile market is growing fast, but at the time of this writing the support for integrating .NET and J2ME through the use of web services was lacking. A number of different ways to account for this fact has been proposed and argued for in this thesis, as well as a discussion on implementation issues. Even though support and documentation for these kinds of projects are lacking currently, my recommendation for future projects of this sort is to use Windows Communication Foundation together with the Web Services Interoperability Technologies. This solution is the most secure and will probably be the choice for developers for a couple of years ahead as it will be made more and more easy to use.

7.2 Thoughts on the project

This thesis work has thought me a lot and provided a valuable experience on how a real life development process really works. Developing an application at the request of a company means dealing with the expectations and demands that will affect us when we start to work after graduating. Even though the project did not land where we expected it to, I would say that both WM-data and myself have benefited from the time spent together. WM-data got a working application with the specified features together with a extensive overview of different implementation alternatives and the problems associated with each. Myself, I have gained a greater understanding of the .NET and Java languages, as well as a workplace experience and social connections that will benefit

me in the future.

The issue that has caused the most problems in this project is without competition the choice to implement the mobile client in Java. This was done to increase the number of mobile devices that the application could be run at as the support for .NET Compact Framework still is lacking on many mobile phones. This decision created all the interoperability issues that have haunted us during the development process and I am certain that a fully operable solution would have been obtained at half the time if the clients had been restricted to those capable of running the .NET Compact Framework. When things have been tested between a .NET service and a .NET client during the development process everything has worked fine but as soon as we have tried to switch to a J2ME client we have started to encounter problems. This if nothing else shows the importance of the design choices made early in the development process, as they can make or break the solution.

7.3 Future work

At the time of this writing, WSIT Milestone 3 have been released. WM-data states that they have had many requests for this kind of application from their customers, so the application developed will most likely be upgraded and improved in the future when the integration support between WCF and J2ME has been improved. It is possible that this early version will be deployed as it still is relatively secure and it is hard to imagine a multitude of threats to the system.

Adding a secure data transport mechanism to the solution should be a simple matter of changing the connection settings on the server side to a more reliable Windows Communication Foundation binding that incorporates secure data transfer and updating the web reference on the J2ME client side as soon as this support works better.

Chapter 8

Acknowledgements

I would like to thank Thomas Nilsson, my supervisor at the department of Computer Science at Umeå, University for his advice when helping me to formulate this thesis paper. His feedback has allowed me to structure this paper in an appropriate way as well as sharpen the technical jargon.

I would also like to thank Nils-Petter Augustsson, Ulf Skoglund, Jonas Olsson and Jens Tinglev in the AD-Tools development team that has assisted me throughout the development process and made my time at WM-data easier. Thanks also to Magnus Forsell, Anders Abrahamsson, Johan Nordlund, Linda Sandström, Johan Jonsson and Lars Zingmark for a great time.

Bibliography

- [1] R. Andersson. Why information security is hard - an economic perspective. Technical report, University of Cambridge Computer Laboratory, JJ Thomson Avenue, Cambridge CB3 0FD, UK, 2003.
- [2] Carlsson B. *Hot och svek: Säkerhet hos människor och datorer*. Studentlitteratur, 2007.
- [3] Åberg R. *Vägen till Active Directory och Windows 2000/XP/.Net : Vad chefer och tekniker minst behöver veta för att börja använda Active Directory, Windows 2000, Windows XP och Windows .Net*. Rolf Åberg, 2002.
- [4] Fujitsu Siemens Computers. Mobile security white paper. Technical report, Fujitsu, <http://www.fujitsu-siemens.com/Resources/155/1082384674.pdf>, 2006.
- [5] Microsoft Corporation. Learn NET Framework 3.0. <http://msdn2.microsoft.com/sv-se/netframework/aa663309.aspx> (visited 2007-04-18).
- [6] Microsoft Corporation. Windows Communication Foundation. <http://msdn2.microsoft.com/sv-se/netframework/aa663324.aspx> (visited 2007-04-18).
- [7] Microsoft Corporation. Windows workflow foundation. <http://msdn2.microsoft.com/sv-se/netframework/aa663328.aspx> (visited 2007-04-18).
- [8] Nokia Corporation. Maximum security, optimum usability. http://sw.nokia.com/id/4ccd9dc2-2dc3-4906-aad3-bc3108c358ab/Maximum_Security_Optimum_Usability_v1_0_en.pdf (visited 2007-04-01).
- [9] Olsson F. *Säkerhet i trådlösa nätverk*. Pagina Förlags AB, 2007.
- [10] Carr H. Sun's project tango. <http://java.sun.com/developer/technicalArticles/glassfish/ProjectTango/> (visited 2007-04-18).
- [11] Hanson J. .NET versus J2EE web services: A comparison of approaches. <http://www.webservicesarchitect.com/content/articles/hanson01.asp> (visited 2007-04-12).
- [12] Preece J., Rogers Y., and Sharp H. *Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons, Inc, 2002.

- [13] M. E. Kabay. Security for telecommuters. <http://www.securitytechnet.com/resource/rsc-center/vendor-wp/trusecure/telecommuters.pdf> (visited 2007-04-04).
- [14] Bishop M. *Computer Security: Art and Science*. Addison-Wesley, 2003.
- [15] Anderson R. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Computer Publishing, 2001.
- [16] Sun. The java ME platform - the most ubiquitous application platform for mobile devices. <http://java.sun.com/javame/index.jsp> (visited 2007-04-18).
- [17] Wikipedia. Java Platform, Micro Edition. http://en.wikipedia.org/wiki/Java_ME (visited 2007-04-18).

Appendix A

Relevant Platforms and Frameworks

This chapter provides a short introduction to the technical topics covered in paper. Although many of the readers may already be familiar with some of these, it is a good idea to refresh their memory and at the same time explain the concepts to those that haven't encountered them before.

A.1 Active Directory

Microsoft Active Directory is one of the cornerstones in WM-data's AD-Tools. Active directory is Microsoft's own implementation of the Lightweight Directory Access Protocol (LDAP), an application protocol for handling directory services, i.e. applications that stores and organizes information about users and resources on a network or computer. An Active Directory stores information about an organization in a central database, where the smallest object can be either a resource (e.g. a printer), a service (e.g. e-mail) and users. These objects can carry different attributes that describes them and are accessed through a tree-structure [3, p.129-151].

A.2 Transport layer versus message layer security

Transport layer security is an approach where security features are handled by the underlying operating system or application servers. Transport Layer Security that will be discussed later is a commonly used transport layer approach used to provide encryption. Figure A.1 shows this approach in action, notice how the data sent is protected during each transmission between servers but not during the time a message is being processed by a server.

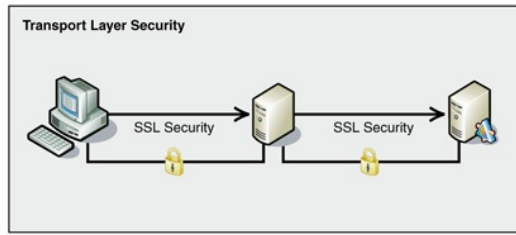


Figure A.1: Transport layer security

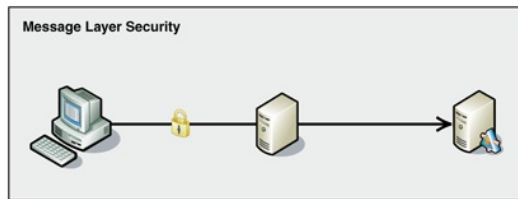


Figure A.2: Message layer security

Message layer security on the other hand as seen in Figure A.2 encapsulates all security information in the message, which enables messages to be protected from attacks even while lying idle on a server waiting to be sent. It has the advantages of flexibility, as different parts of the message can be encrypted for different recipients, the possibility of auditing as intermediaries can add their own headers to the message and it also supports sending messages over a wide range of protocols such as SMTP, FTP and TCP without having to rely on the protocol to provide security measures.

Worth noticing is that if there does not exist any intermediaries between the sender and receiver, a transport layer solution in essence works the same way as a message layer solution, without being as CPU intensive.

It is possible to implement both kinds of security, but one must be aware that these features come with both a size and time penalty added to them, as the messages sent are larger and it takes more time and CPU power to handle each package.

A.3 Transport Layer Security

TLS and its predecessor Secure Sockets Layer (SSL), are cryptographic protocol standards developed by Netscape Corporation aimed to provide secure communication on the Internet. TLS is designed to prevent eavesdropping, tampering and message forgery and provides authentication and privacy using cryptography. This protection is provided at the transport layer in the OSI model through the use of certificates [14, p. 291-298].

Most often, only the server side is authenticated, which means that the clients know whom they are talking to but not the server. To also ensure the identity of the client requires the use of mutual authentication in the form of a public key infrastructure deployment to the clients.

A special usage of TLS is when it is performed over a HTTP connection to form HTTPS. HTTPS is among other things used to secure web pages for the use of electronic commerce and protects the sensitive data during the phase when it is sent from a computer or mobile device to the server.

A.4 Public Key Infrastructure

A PKI is an arrangement that allows trusted third parties such as authorized Certificate Authorities (CA:s) to vouch for other server's identities by binding public keys to users in the form of certificates.

These arrangements allow computer users to be authenticated to each other and also to encrypt and decrypt messages based on each other's public keys. Identities are established based on certificate chains, where one trusted CA vouches for another which in turn vouches for yet another. This way a chain of trust is created, and if the CA that has vouched for the server is on the list of the client's trusted CA:s, the client in turn trusts the server and can establish a secure connection [14, p. 254-260].

Another usage is client authentication, where the client also has a certificate installed that verifies its identity. This way communication can be established between the server and trusted parts.

A.5 Java Platform, Micro Edition

Java Platform Micro Edition (J2ME) is a application platform for mobile devices that allows software to be ported to a broad range of devices without any altering of code. J2ME consists of a collection of Java APIs that is aimed at resource-constrained devices such as PDAs and cell phones [16]. Other similar platforms include Symbian, .NET Compact Framework, Palm OS and Pocket PC.

J2ME devices implement a profile that specifies what capabilities are available for the developer to use. The most used one are the Mobile Information Device Profile (MIDP) that is aimed at mobile devices and allows the application developer to use a GUI API and 2D gaming API among other things. Applications developed for this profile are called MIDlets, and this profile is now the de facto standard for cell phone games.

A profile is a superset of a configuration, of which there exists two: Connected Limited Device Configuration (CLDC) and Connected Device Configuration (CDC). These two are subsets of the Java class libraries aimed to operate on the resource restrained mobile

devices. The CLDC contains a stricter subset of classes than the CDC which contains almost all the Java SE libraries that are not GUI related [17].

A.6 .NET Framework 3.0

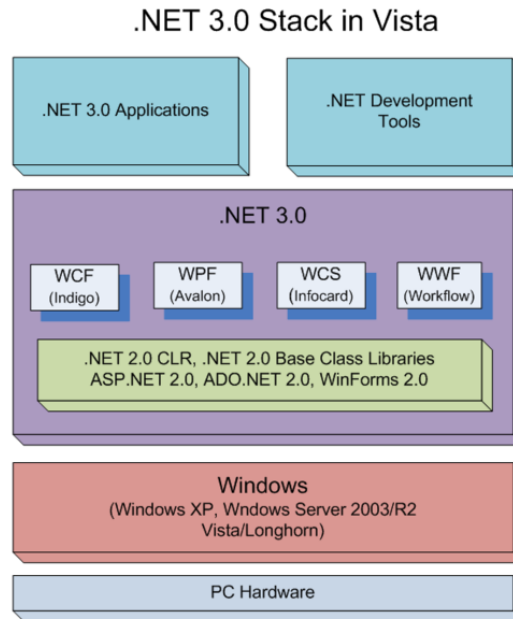


Figure A.3: .NET Framework 3.0 Architecture

Figure A.3 shows the architecture of Microsoft's .NET Framework 3.0. This is Microsoft's new managed code programming model for Windows that builds upon the .NET framework 2.0, and adds four new components: Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF), Windows Workflow Foundation (WWF) and Windows CardSpace (WCS) [5]. This paper is mostly concerned with WCF and WWF as these are the technologies used in the solution.

A.6.1 Windows Communication Foundation

WCF is a collection of .NET technologies designed for building and running connected systems. It uses a service-oriented messaging system to allow programs to inter operate locally or remotely in a similar way that web services does. The WCF programming model combines Web Service, .NET Remoting, Distributed Transactions and Message Queues into a single service interface that consists of the service to be hosted, a host environment and one or more endpoints to which clients can connect [6].

Message security in the Windows Communication Foundation context means that security mechanisms such as confidentiality, integrity and authentication is handled at the SOAP message level. Here, the SOAP messages used to transport information between a service and a client is signed and the data encrypted, so that an attacker that looks at the SOAP message will only be able to see the SOAP envelope while the body consists of encoded cipher text.

The alternative, transport security, is achieved when WCF allows TLS to handle the confidentiality, integrity and authentication of the transported data. No message security is provided but instead the WCF service have to be hosted along with an installed certificate so that TLS can secure the communication at the transport layer.

Both these alternatives is viable and can also be combined through the security mode “TransportWithMessageCredentials”. The difference is that message security is more flexible in terms of what credentials can be used for authentication purposes, while TLS is widely tested and accepted way of securing data transports. No matter which alternative is chosen, what kind of credentials used for authentication still has to be specified. The options available are using no credentials, providing a username and password, using existing Windows accounts as credentials, using certificates and using federation security tokens.

A client connects to a service through a binding, and two of them will be discussed here. First, there is the `basicHttpBinding`, which is the simplest and most interoperable and support the WS-I basic profile. This binding is insecure by default but can be made to run over HTTPS. Secondly, there is the `wsHttpBinding` which uses message security by default. This binding supports WS-Security which makes it more secure but less interoperable.

A.6.2 Windows Workflow Foundation

WWF is used to define, execute and manage workflows., which allows for task automation and integrated transactions. A workflow consists of one or more activities that can require or provide data and each workflow can later be hosted and communicated with through the use of WWF interfaces [7].

A.7 Web Services Interoperability Technologies

WSIT or Project Tango is a project hosted by Sun where engineers from Sun and Microsoft work together to create a common way to connect Java Web Services and WCF. This project ensures interoperability of both sides features such as security, reliable messaging, and atomic transactions specified in various WS-* specifications. The goal of the project is to ensure that interoperability is achieved with regard to bootstrapping, optimizing and securing communication between the two platforms, as well as enabling reliability and atomic transactions [10].

Appendix B

AD Mobile Tools Flow Chart

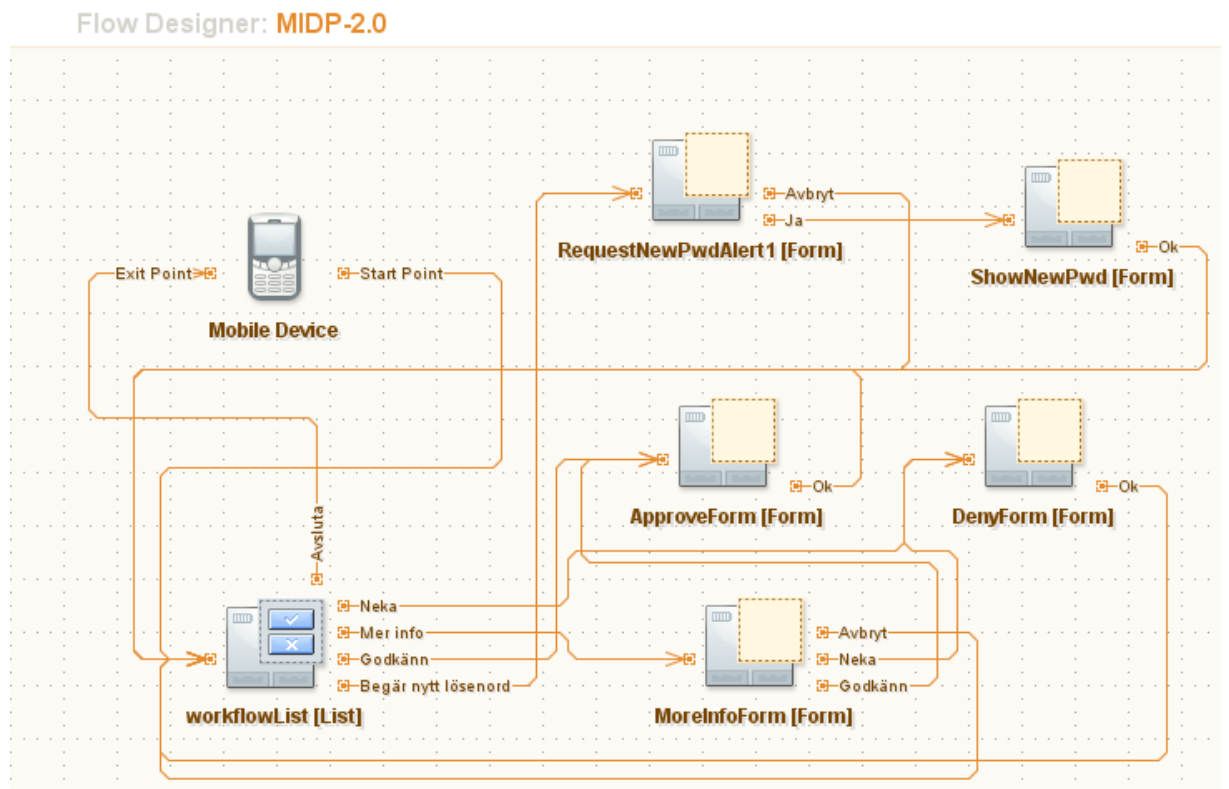


Figure B.1: AD Mobile Tools Flow Chart