Security of mobile TAN on smartphones

A risk analysis for the iOS and Android smartphone platforms

Master thesis

Author:	Laurens Koot (s4035186)
University:	Radboud University Nijmegen
Faculty:	Faculty of Science
Course:	Information Science
Graduation number:	160IK
Supervisor:	dr.ir. E. Poll
Date:	24-02-2012

Contents

Preface

I am glad that I had the chance to combine my personal interest with the research I have performed. I would like to thank my supervisor Erik Poll who provided very useful feedback and helped me to complete my thesis. Finally, I would like to thank my friends and family which supported me in writing this thesis.

Abstract

This thesis investigates if and how the threat model of the mTAN (mobile TAN) security mechanism is changed with the advent of smartphones. The research is divided into an overall analysis, to research the mTAN security mechanism itself, and a risk analysis which is about analyzing the security mechanisms for two important smartphone operating systems. The risk analysis is performed by analyzing the security model from iOS and Android and assumes no unknown bugs are present at these systems. From the overall analysis it becomes clear that in general two devices are involved when using mTAN: a computer and a (smart)phone. The following two goals are the most likely ingredients for an attacker to compromise a service secured with mTAN:

1. Install malicious app on the smartphone of the victim
2. Identify a computer and smartphone which are both owned by the victim

Both goals are needed for an successful attack because the required information is spread over both devices.

The risk analysis showed that the probability is high that the first goal can be achieved by an attacker. The most probable attack vector for Android is that the attacker clones a popular app, infects it with the malicious code and offer this for download at the Market. This app can be installed by the victim or remotely installed by the attacker when the Gmail credentials of the victim are obtained.
Installing a malicious app on iOS is more difficult than Android as an app for iOS has to be reviewed by Apple, before it can be installed to the device. The most probable attack vector for iOS is when the device of the victim is jailbroken. This give the attacker the capability to offer the malicious app and in some cases also give the capability for remote installation.

To achieve the second goal the most identification information can be found at an infected computer. This information is used to find the smartphone which belongs to the same user. With an iOS device, the identification information can easily be found. As iOS devices are required to be connected with the computer for activation or adding content it usually also makes a backup of the device through iTunes. This backup contains identification information which can be used by the attacker.
Android does not need to be connected with the computer but has the requirement to use a Gmail account to access the Market. The probability that this same Gmail account is used at the computer is high. This way the Gmail account can be abused to identify the devices from the victim.

As both goals could be achieved we can conclude the threat model changed with the rise of smartphones and mTAN is not a secure security mechanism when used with a smartphone.

1 Introduction

Authentication is a technique which we deal with almost every single day. When we travel, we authenticate ourselves to the customs with our passport and when we electronically pay in a shop, we authenticate ourselves using our bank card with the PIN.

In "real life" we can say this kind of authentication is safe. It is hard for criminals to create fake passports or clone bank cards which make use of enhanced security features like a smartcard. In the digital online world we have more trouble with this kind of authentication to protect our data.

For example, authentication in the digital world is needed when you want to have access to your e-mail account. In most cases only a username/password is needed for authentication. Because this kind of information can be leaked, companies are looking for alternative ways to secure the authentication in a better, but still cost effective way.

One relative low-cost solution to increase the security of the authentication, is to make use of mTAN. In practice this means that a user will not only have to authenticate himself using his username & password at his computer, but also must enter a mTAN received on his (smart)phone.

This kind of security is used for several kind of services because the solution is low-cost and many people can make us of it as most users are equipped with a mobile phone. The security mechanism mTAN is also used by banks to secure their online transactions. Recent reports [1] [2] showed there was a significant growth in the online fraud for online banking in the Netherlands. We cannot direct relate this fraud with the mTAN security mechanism but it shows criminals are targeting their attacks at online banking.

In 2010 the Radboud University [3] performed a research which suggested that mTAN cannot be used for sensitive data due to the vulnerability in the GSM encryption. The effect of this research was that the EPD was not launched as confidentially could not be guaranteed. When mTAN was introduced, it was designed to be used with a mobile phone. This mobile phone is being replaced with a smartphone in the last couple of years which could change the threat model. As other services still rely on mTAN to secure sensitive data, we will perform a risk analysis to analyze if and how the threat model changes with the advent of smartphones.

1.1 Research question

This research is focusing on the security aspect of authorization which rely on the usage of mTAN. To give this research certain guidelines the following research question is defined:
"To what extent is mTAN still a secure method to use with a smartphone?"

To research this security mechanism we have defined an end-goal which has to be achieved for an successful attack: *"Abuse a system which is secured with mTAN".*

To support the research question the following sub-questions are defined:

Overall analysis

1. *Which sub-goals are needed for the end-goal?*
2. *Which security mechanisms and vulnerabilities are in place?*
3. *What are possible attack vectors to achieve the end-goal?*

Risk analysis

4. *Which security mechanism prevents to achieve the end-goal?*
5. *Which vulnerabilities helps to achieve the end-goal?*
6. *What is the most likely attack vector to achieve the end-goal?*
7. *Which counter measures could be introduced to make the system more secure?*

In chapter 3 we will examine how mTAN works and how it is used. This information, in combination with a short research about the selection of two smartphone operating systems, will give us the basic information about the subject which the research will be further based on.

In the risk analysis, in chapter 7, we will try to answer the sub-questions within the secured model. We will research the security mechanisms within the current design and implementation of the mobile operating systems. We will not research low-level security vulnerabilities and assume there are no unknown bugs at the OS.

The research will be performed for devices running Android 2.3.3 and devices running iOS 4.3.3. These operating systems are selected as they are currently the most widely used in smartphone operating systems and expected to grow even further in the future. In chapter 4.2 we will further substantiate this selection.
By default Android and iOS devices are secured with security mechanisms which limits the user in their capabilities. Disabling important parts of these security systems is called rooting (Android) or jailbreaking (iOS) the device. Because of the high popularity of rooting/jailbreaking we will also research the impact of this in the risk analysis.

1.2 Method

This research has been divided into 2 phases. The first phase, which will be performed in chapter 2,3,4 is meant to give a clear view on the mTAN concept and provide background information about the involved techniques. We will also research the possible attack vectors, in chapter 5, to achieve the goal to attack an service which is secured by mTAN.

In the second phase these results will be used in an empirical research & literature study. The described techniques and the chosen attack vector will be used to describe the involved security mechanisms and vulnerabilities in chapter 6. This information will be used in combination with the threat model from chapter 5 to perform the risk analysis in chapter 7.
As an risk analysis may become complex and there are several methods to perform a risk analysis we have chosen to perform this risk analysis by using the method called Attack Tree modeling. Research from Sjouke Mauw and Martijn Oostdijk [4] showed that Attack Tree modeling is a method which can help to clarify the treats in an security system.

Attack Tree modeling is a technique which was introduced by Schneier in 1999 [5]. As the original design of Attack Tree modeling is rather vague, we use the report from Amenaza [6] which give some guidelines how the model can be used. As the report is about using the model to develop a secured system, but our goal is to evaluate an existing system we will use these steps in the following way:

1. Create an Attack Tree model
In chapter 5 the Attack Tree model will be defined. From this model we will select the most interesting attack vector and develop a new model.

2. Identify likely attacks using capability analysis
In chapter 6 we will research the security mechanisms and vulnerabilities in the system. Using this information we can define attacks which are added to the Attack Tree from chapter 5.

3. Evaluate the impact of attack scenarios &
4. Determine the risk level of each attack scenario
In chapter 7 we will perform the risk analysis which combines steps 3 and 4. In this risk analysis we will research the attack vector. We will use the Attack/Defense tree method from Patrick Schweitzer [7] to combine the security mechanism and vulnerabilities with the Attack Tree to perform the risk analysis.

5. Attack detection.
This step is meant to build detect in the Attack Tree as an counter measure. We will not use attack detection as a counter measure but define some counter measures from the risk analysis in chapter 8.

1.3 Relevance

If this research is relevant depends on several factors. Which and how many instances make use of mTAN, how big exactly is the evolution from mobile phones to smartphones and how big is the malware problem on mobile devices? Answering these questions can give us an idea if the threat model for mTAN will change and if the research about this subject is relevant.

1.3.1 Service providers using mTAN

mTAN is a security mechanism which nowadays is not only used for mobile banking but also used for securing access to other sensitive services. In the Netherlands the government is offering more and more services through the internet. A lot of those services are sensitive like the yearly tax declaration or a police report. To secure those services the government make use of the service DigiD, a service created by the government to securely offer these services to their citizens.

DigiD
DidiD offers a central authentication system to the citizens of the Netherlands for the following categories of services:
- National organizations
- Municipalities
- Provinces
- Water boards
- Police
- Health insurance

Over 600 institutions [8] make use of this system. The service relies on a username and password which is sent by the Dutch mail to the citizen. By sending this password using the Dutch mail they can guarantee the password is only received by the user of the system.
An institution can chose to increase this security by adding SMS verification for authentication and transactions which is called mTAN. When this security is enabled, a citizen will receive a mTAN on his mobile phone which he has to enter for transactions or authentication to the system. About 80 of the 600 institutions have enabled this enhanced security for protecting their services.

Beside the government, a number of companies are also in need of a central secure authentication system. Digidentity is the successor of DigiD which offers the services for the government and commercial companies.

Digidentity
Digidentity is a commercial service which stores your identity in a digital way and offer authentication to other services. Digidentity make use of the same technique as DigiD to authenticate a user to the system: A username and password and in some situations a mTAN.
Digidentity is fairly new and is not used by a lot of companies yet but by the support of the government, it could grow fast in the near future. It has the goal to offer their product for very sensitive services which would normally need a handwritten signature.

Banks

Most banks provide digital services to access digital services such as "internet banking". The data which the bank has to secure, money, is sensitive and very interesting for hackers. Hacking someone's bank account can be lucrative so banks try to secure their services with several security mechanisms. Most banks use 2 different methods to secure their authentication and their transactions:

Method 1: *Hardware based token with bank card:*

This method uses a hardware device which is sold/given to the client of the bank. This hardware device mostly consists out of a device where the user has to insert their bank card and enter their PIN as shown in figure 1. The device has a display which will generate a code which must be used for authentication and authorization with the bank.

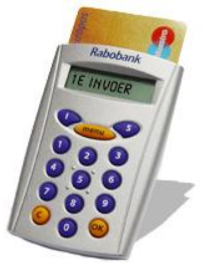


Figure 1

Method 2: *Username & password in combination with mTAN*

This method uses a username & password for authentication. For authorization of the bank transfers, mTAN is been used.

It is difficult to make a complete overview how much the mTAN method is used in banking around the world. To get an idea, we have searched for suppliers which offers this kind of security and researched the customers to determine if this security mechanism is being used. One of these found suppliers is Netinfo which is responsible for developing and implementing secure banking systems. Netinfo has implemented banking systems secured with the mTAN for the following banks:

- ING
- Raiffeisen Meine
- Bawag PSK
- Easybank
- Alpha Bank
- Banco Falabella
- Bank of Athens
- Bank of Cyprus group
- 1bank.com
- Eurobank ERG
- National Bank of Greece

As there are more suppliers than Netinfo this list is not complete. To get an overview of all the banks using the mTAN security mechanism further research is needed.

1.3.2 Evolution of the mobile phone

Besides how often the system is used, the evolution of the mobile phone is something else which determine the relevance of this research.
As described in the introduction, mobile phones encountered an evolution towards smartphones as we know them nowadays. Smartphones are capable of performing almost all the features which are available to computers. This evolution changes the threat model to mTAN as these new features also provide more capabilities to attack the system.

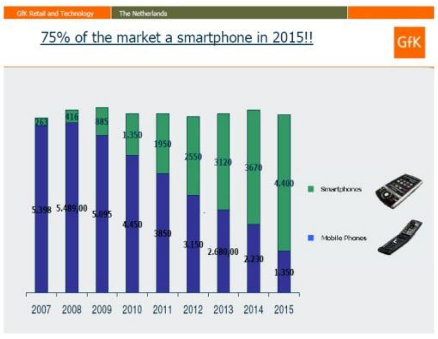


Figure 2

Smartphones are not new and already exist for several years, but the early models were very buggy and users did not globally accepted this smartphone as a replacement for their mobile phone. Figure 2 shows the prediction that most users will replace their mobile phones with a smartphone in the near future. From 2007 up to 2011 this growth is already noticeable which also confirms this prediction.

1.3.3 Malware

The Symantec security report [9] shows that Symantec recorded over 3 billion malware attacks in 2010.
Special attack kits were developed and sold to criminal organizations, so those organizations could easily abuse the vulnerabilities for criminal activities. With the help of these attack kits, Symantec recorded 286 million malware variants which are responsible for the 3 billion malware attacks.

Mobile threats
The Symantec security report [9] shows that in 2010 163 vulnerabilities were discovered for mobile devices. This is a rise of 74% comparing to the 115 vulnerabilities discovered in 2009. It is likely that this rise has a relation with the evolution of the mobile phone. The vulnerabilities however were not active abused as it was yet not economic interesting for criminal organizations. It is likely that when criminal organizations will get interested in these devices the malware will rise.
The McAfee Threats Report [10] shows a rise of malware for the mobile platform and therefore has announced 2011 as: "The year of mobile malware".

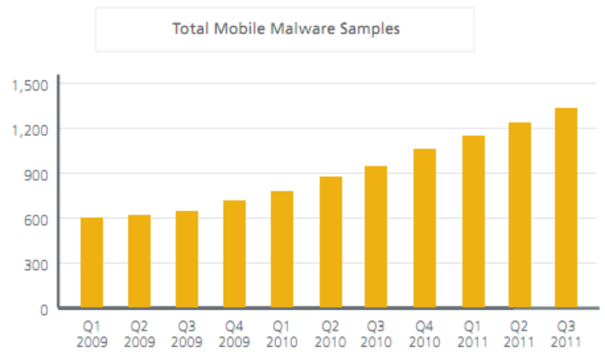


Figure 3

Figure 3 from this report shows that the subject mobile malware is something which is rising right now. With a number of over 1200 malware samples it doubled from 2009.

2 Authentication & authorization problem

In this chapter we will research and explain the problem of authentication and authorization in the digital world. As these are the motive that security mechanisms like mTAN exist, we can use this information to better understand why and how mTAN is designed and implemented.
We can assume that IT professionals are familiar with these subjects but for other readers the content of this chapter can help to understand this paper.

2.1 Authentication

Authentication is the process about identifying yourself to something or someone. Authentication is something we use in real life and in the digital world. We will explain the usage in the digital world by explaining the following examples:

2.1.1 Telephone

When we have to authenticate using the telephone, we mostly do this by communicating personal information which it is likely that only an genuine person would know this information. This kind of information mostly consists out of information like your name and date of birth. In some cases a secret code is asked but because the services accessible by telephone are not that sensitive, and the fraud (because the hacker is less anonymous) is relative low, codes are not frequently used.

2.1.2 Internet

When we have to authenticate using the internet this is mostly done by a username and password. A username & password is information which can leak and be abused by an attacker. Leaking is mostly done by malware installed at computers. This malware could contain key loggers which simply records the username & password and send it to the attacker. Another problem with these passwords is password sharing. Research from Microsoft [11] showed that a password from a user is re-used with about 6 different services. As the password is stored at a service this password could leak when an attacker attacks this system. Password sharing greatly enhances the chance an attacker can access other services from the user.

2.1.3 Possible authentication methods

In the last two sections we have explained two examples of authentication. In both ways the authentication was performed by something the user knows: a username & password or personal information.
There are other ways to authenticate yourself but in base they all belong to one of the three methods from table 1. A service could chose to only use one of these methods or combining the methods which is called two (or three) factor authentication.

Method	*Example*
Something you know	Username and password
Something you have	Smartcard, smartcard
Something you are	Fingerprint, iris

Table 1

2.2 Authorization

In paragraph 2.1 we have described what authentication is and how it can be used. Authorization is not about authentication a user to a system, but the actions which are performed by this user after he is authenticated. When we talk about authorization, this focus on subjects like "*is the user allowed to access this data*" or "*is the submitted data by the user valid*".

An example of this is used in transactions through internet banking. When you access the system, you first have to authenticate yourself in a way to access the data. When you make a payment, you create a transaction which has to be authorized by the user and the bank. mTAN is one of the available security mechanisms which are used to authorize these transactions. In chapter 3 we will further research this authorization method.

2.3 Identity theft

An attacker can do several things when he could bypass the authentication and authorization security mechanisms. One example of this is called identity theft [12]. With identity theft we mean that an attacker could use someone's identity to perform certain tasks. There are several forms of identity theft which are classified as follow by the Identity Theft Resource Center [13]:

- Criminal identity theft
- Financial identity theft
- Identity cloning
- Medical identity theft
- Child identity theft

All these kinds of identity theft can be used to perform criminal activities. In the way these activities can be performed depends on the amount of time an identity is stolen.

With full identity theft the attacker could loan money at a bank or shop with a false credit card which belongs to the victim. The outcome of this kind of theft is that the victim will lose a lot of money, but also have to prove in some kind of way that he is innocent. To perform these activities, the attacker must be able to authenticate and authorize himself as the victim.
With temporary identity theft the attacker will use the identity of the victim in an fairly shorter amount of time. Stealing all the money of an victim his bank account is an example of this kind of theft. It is mostly used by tampering with transactions which must be authorized. When an attacker tampers a bank transaction, but let the victim authorize this payment, we can speak of temporary identity theft.
These types of theft overlap as showed in figure 4. When an attacker can authenticate himself as the victim and authorize the transactions, but only do this once, this overlap occurs. As the focus of the research is mTAN, which is mostly used to secure an individual service, we will research the situations where this overlap occurs.

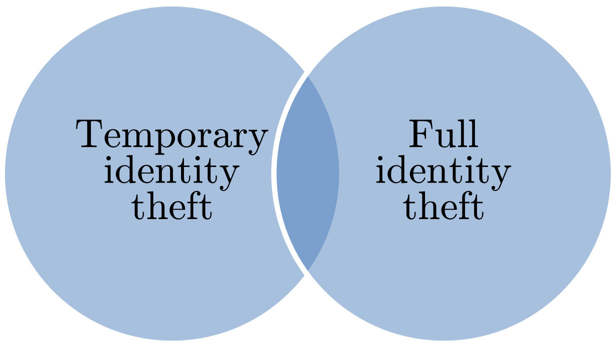


Figure 4

3 OTP, TAN and mTAN

In chapter 2 we have explained the authentication and authorization problem and introduced mTAN as a security mechanism to secure this kind of authentication and authorization. As mTAN is the main subject of the research paper and the definition of the technique is very vague and not used consistent, we will clarify the techniques in this chapter.
The content of this chapter is derived from an literal research and describes our perspective of the techniques used as nowadays. As the OTP (One Time Password) is the mechanism where the other mechanisms are partly based on, we first will explain this mechanism in chapter 3.1. In chapter 3.2 we will use this information to explain the TAN mechanism and explain the mobile variant , mTAN, in chapter 3.3.
These security mechanisms are often used in combination with another authentication method. This combination is called two factor authentication which will be explained in chapter 3.4.
To clarify the usage of this mTAN, we will introduce in chapter 3.4 a case which uses two factor authentication method in combination with mTAN. This case will be used further on in this paper to be able to give examples how an attack could be performed.

3.1 OTP

An OTP is an alphanumeric combination of characters which can only be used **once** for authentication, sessions or transactions. The code from this OTP does not have any relation with the transaction. The most of the time OTP is used to secure the authentication to the system by using a different password every time authentication is used. The advantage of this system is that it is not important anymore if the password will leak.

An OTP is mostly a randomized code which can be generated in several ways. To make sure the OTP is not read by an attacker the OTP can be delivered to the user in several ways. Table 2 shows these different kind of delivery methods with its description, advantages and disadvantages.
OTPs are generated in several kind of ways, but more important is the way the tokens are displayed to the user. In table 2 these different ways are described with their ad- en disadvantages.

Solution	*Advantage*	*Disadvantage*	*Description*
Pre-rendered tokens	- Cannot easy be stolen in a digital way. - Are (mostly) not stored at a computer	- Can be copied/stolen physically.	A list of pre-rendered tokens are submitted to the user. The user uses an index number to determine the new password.
Software tokens	- No paper list needed - As there is no list, it cannot be copied	-Software can be hacked - Are stored at a computer	The OTP is generated by a software program on the computer of the user.
Hardware tokens	- No list can be copied - Separate from computer - Dumb device, difficult to hack	- Expensive - Device always needed	A device is used to generate the token which is (mostly) not physical connected to the computer.
SMS tokens	- Cheap - Separate from computer - Mobile phone is a dump device	- Delivery of token can be intercepted. - Smartphones are not dump	The password is generated remotely at the server and transferred by SMS to a mobile phone.

Table 2

3.2 TAN

TAN (Transaction Authentication Number) is a type of OTP which is used with transactions. A real TAN is rendered as it is related with a transaction. A pre-rendered version is missing this kind of information simply because the transaction did not exist when the TAN was generated. Pre-rendered TANs mostly consists out of a list of consecutive TANs which can be used for any kind of transaction.

TANs are also “abused” to secure authentication to a system. Google, Facebook and DigiD are examples of companies using TANs for stronger authentication.

A TAN can be delivered to the user in the same 4 ways as an OTP which is described in table 2. When a TAN is pre-rendered (like the paper tokens) no transaction details can be used. When an attacker would copy a list of these TAN list he can abuse this to transfer money to his own bank account. In a Software, Hardware and SMS token transactions details are included in the calculation of the code. This means the TAN cannot be abused for transactions with other details.

3.3 mTAN

A mTAN is a pre-rendered TAN which is delivered through a SMS message. The contents of this SMS messages differs by the way it is being used. A SMS message has the ability to contain up to 160 characters to communicate this TAN. A TAN mostly consists out of about 8 characters so there is some room left for additional details. These additional details can be transaction details like the bank account number where the money is transferred to. In case of temporary identity theft, when the transaction is being tampered with, this malicious transaction details will be visible to the victim.

In section 1.3 we have described that a lot of companies make use of this mTAN. From an attacker perspective it is likely that the service which generates money instantly is the most popular service to hack, which is a bank. In section 3.5 a case will be introduced which is based on the real online banking service of ING from The Netherlands. The reasons why we have chosen this case, is because currently ING is the only bank in The Netherlands which make use of mTAN. Furthermore we have personal experience with this system.

3.4 Two factor authentication

Two factor authentication [14] is a method which combines exactly two forms of authentication. Mostly the method "*What you have*" is combined with the traditional "*What you know*" method. A basic example of this usage is the way people do their cash withdrawal in the Netherlands. Their bank card is combined with their PIN. The downside of this implementation of two factor authentication is that the device could be easily copied. That's why a new law forces that banking has been done by a new kind of bank card which make use of a smartcard.

The communication channel is another aspect of two factor authentication which can enhance the security level. At an ATM machine the bank card and the PIN are both entered into the same device. If an attacker hacks this device, he can capture information about the bank card and the PIN. With a mTAN secured service the username & password are mostly entered at the computer and the mTAN is received at the smartphone.

3.5 Case: ING Netherlands

The ING in the Netherlands is a financial institution which offers internet banking to their customers. A news report from ING [15] showed that about 2 million people use this service every day. Table 3 shows the available services from the ING, which could be accessed through internet banking. Not every service is secured with the same authenticated method which, is also displayed table 3.

Service	*Required authentication method(s)*
Bank transfers to other accounts	- Username & password and - mTAN
Balance inquiry at current and savings account	- Username & password
Creating and viewing insurance policies	- Username & password
Stock investing	- Username & password
Bank transfers to investing account	- Username & password and - mTAN
Bank transfers from savings to current account	- Username & password and - mTAN

Table 3

The username & password in this case cannot be chosen by the user, but is chosen by ING and sent by the Dutch mail agency to the victim. The ING trusts the Dutch mail agency that this username and password is delivered to the user and not read by someone else. This is guaranteed as the Dutch mail agency trusts its employees and the username & password will only be delivered to the user who can identify himself with a valid Dutch identity card.

Besides the username & password also a phone number is used for the mTAN security mechanism. To register or change this phone number the customer has to visit ING and identify himself with:
- A valid Dutch identity card
- The bank card from the ING
- The mobile phone number
- The PIN from the bank card.

The process of a bank transfer is schematic displayed in figure 5. It shows that the mobile device is only used to authorize the payment by entering the received mTAN.

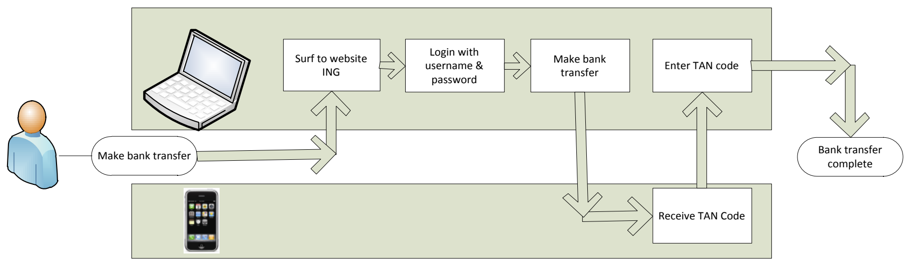


Figure 5

When an attacker wants to steal the identity to perform this process he has two goals to be achieved:
- Retrieve the username & password from the victim
- Read the mTAN sent to the registered number of the victim

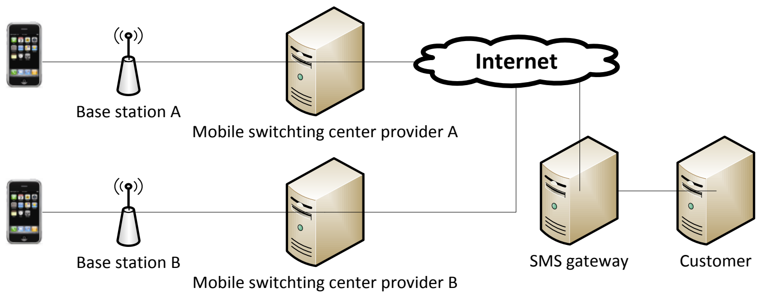
4 SMS & smartphone

In chapter 3 we have described mTAN and showed that the delivery of mTAN is done to a mobile phone through the mobile network. When an attacker wants to intercept the mTAN, this can be performed at several locations: the sender, transportation or the destination. In this chapter we will research the used techniques/devices to transport this mTAN. The research is performed using a literature research to discover the involved techniques/platforms for the delivery of the mTAN. We can use the results of this chapter in chapter 5 where we will define the threat model and search for security holes.
In chapter 4.1 we will describe the transportation (SMS) and in chapter 4.2 the destination of the device: the mobile phone/smartphone.

4.1 SMS

SMS is the mechanism used by mTAN to deliver the mTAN to the user. Phone networks offers alternative systems like MMS or internet to deliver messages to phones in their network. SMS is used as it is a standard which is supported by almost all mobile phones.

4.1.1 Architecture



Internet
Base station A
Mobile switchting center provider A
Base station B
Mobile switchting center provider B
SMS gateway
Customer

Figure 6

Figure 6 shows the relevant components in this paper which are used to be able to transport SMS messages. These are:

- A phone which is capable of sending/receiving SMS messages.
- The mobile switching center provider which is responsible for storing receiving and sending SMS messages.
- The SMS gateway which is responsible for transferring messages from the internet to the mobile phone providers.
- Optional: A customer which sends SMS messages through a SMS gateway to the provider which is not capable of receiving SMS messages.

4.1.2 Security

The way SMS messages are secured is not publicly known. From the mobile switching provider to the phone messages are mostly transported using GSM (2G) or UMTS(3G). These standards use encryption to secure the traffic.

The communication between a customer SMS gateway and the mobile switching providers is likely not to be encrypted. This is encountered due to our own experience as a technical engineer of a telecom provider. We have encountered that most security relies on IP whitelisting instead of encryption. The reason is that encryption has a great amount of overhead and will significant increases the delay and CPU power for delivery of these small messages.

4.2 Smartphone

As explained in chapter 1, the research will be performed on Android 2.3.3 and iOS 4.3.3. In this chapter we will substantiate this decision by research of the Market share/ and current existence of the mobile malware.

4.2.1 Market share

As there is a race in who is the biggest smartphone supplier, it is difficult to collect statistics about the Market share. One source which seems valid is Gartner [16] who also published their predictions of the market share of mobile phone operating systems which are showed in table 4.

OS	2010	2011	2012	2015
Symbian	37,6%	19,2%	5,2%	0,1%
Android	22,7%	38,5%	49,2%	48,8%
RIM	16%	13,5%	12,6%	11,1%
iOS	15,7%	19,4%	18,9%	17,2%
Microsoft	4,2%	5,6%	10,8%	19,5%
Other	3,8%	3,9%	3,4%	3,3%

Table 4

Table 4 shows that Symbian is currently the Market leader of OS in mobile phones. As Symbian is not really an OS for smartphones and Nokia announced in 2011 [17] that they will discontinue the development of Symbian, the market share will be in 2015 as low as 0,1%. Due to the rapid development of smartphones software we have decided to make the selection of the operating system based on the current market share in combination with the future market share as showed in table 4. Based on this information the top 3 will be (in order):

1. Android
2. Microsoft
3. iOS

As there is only enough time to research 2 of these operating systems we have selected 2 candidates based on our preference. As the operating system Windows Phone is relative new, and is still developing itself, we have chosen the operating systems iOS and Android.

4.2.2 Mobile malware

Malware is something we know for ages from the computers which made it quite common that a computer does not run without having antivirus software installed. As mobile devices were dumb it was difficult and not lucrative to write malware for the mobile Market. The past showed malware did exists for mobile phones but the numbers were relative low. With the evolution to smartphones this also give new opportunities for mobile malware.
In this chapter we will investigate the available malware for the operating systems. This gives an idea how popular the device is for an attacker and if the OS is vulnerable.

The total mobile malware by platform is showed in figure 7 which is derived from the McAfee threats report [9]. This figure shows that in 2011 the most mobile malware is detected on:
1. Symbian
2. Android
3. Other

It is noticeable that iOS is not specific included in this chart. The reason is that the amount of malware, which is low, is included in the “others” section. The number of Symbian are still the biggest but in the previous section we have explained that this system will have a Market share of almost zero in 2015. It is likely that when the Market share will become that small, the amount of malware for this system will also drop. The second biggest candidate for mobile malware is Android. When we look at the growth of this malware in the past 3 quarters of this year we see an explosive growth as shown in figure 7.

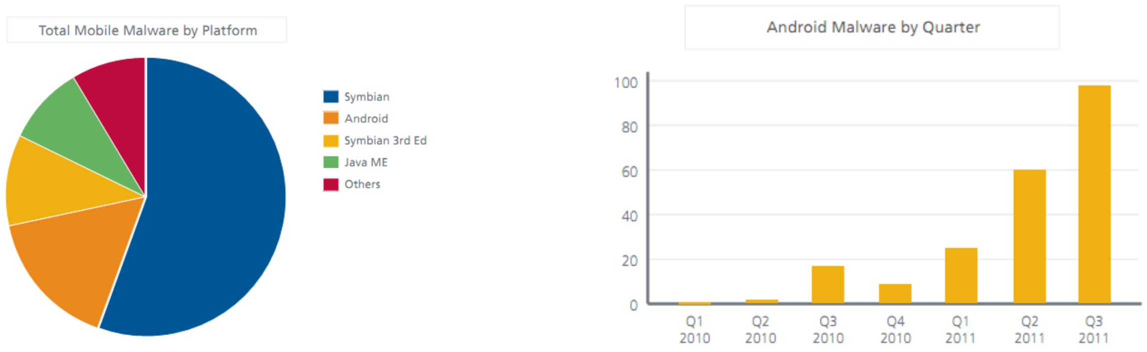


Figure 7

4.2.3 Selection of mobile OS

Based on the popularity, iOS and Android are the selected operating systems to be researched in this paper. The mobile malware analysis shows that Android is an interesting platform as it is not only popular in the market, but also for the attackers. As no other operating systems, besides Symbian which is deprecated, outstand in the malware report, we have selected iOS as the second operating system.
As security mechanisms and vulnerabilities differ in each version of the OS we will research the most used version. Statistics [18] [19] shows that iOS 4.3.3 and Android 2.3.3 are currently the most popular versions at the time this paper has been written.

5 Threat model

In chapter 3.4 we have introduced the case of the ING Netherlands. This case will be used as an example service which we will try to attack and achieve the goal: *"make an illegal transfer of money"*. As this goal is rather vague we will try to clarify it in chapter 5.1 and related it to our research question.
In this chapter we will make use of the Attack Tree method to define and find the possible attack vectors which can lead to this goal. The outcome of these possible attack vectors can/may be found in section 5.2. The possible attack vectors will be briefly analysed to determine the probability of each attack vector in chapter 5.3. In chapter 5.4 this probability is used to select an attack vector which will be further defined in greater detail and used in the risk analysis in chapter 7. Only some attacks vector will be selected as there is a limited amount of time available in this research thus it is not possible to perform a risk analysis for each available attack vector.

In this chapter we will answer the following sub research questions:

- *"Which sub-goals are needed for the end-goal? "*
- *"What are possible attack vectors to achieve the end-goal?"*

5.1 Attack goal

When the attacker wants to achieve the goal *"make an illegal transfer of money"*" based on the ING case from chapter 3.4 the following goals have to be achieved:

- Retrieve the username & password from the victim
- Read the mTAN sent to the registered number of the victim

These goals could be achieved by phishing but we will ignore this technique as phishing is always possible in an attack. The most logic way to retrieve the username & password is to hack the computer of the victim and install a key logger. To achieve the second goal the attacker must be able to read the mTAN from the victim. Important is that the attacker must be capable to retrieve the credentials and the mTAN from the same user.
In traditional online attacks an attacker starts with an IP scan to find candidates which are likely possible to be compromised. After this selection the attacker will try to compromise these systems in an automatized way. This could be smartphones or computers which will result in a list of computers and/or smartphones which the attack could "control".

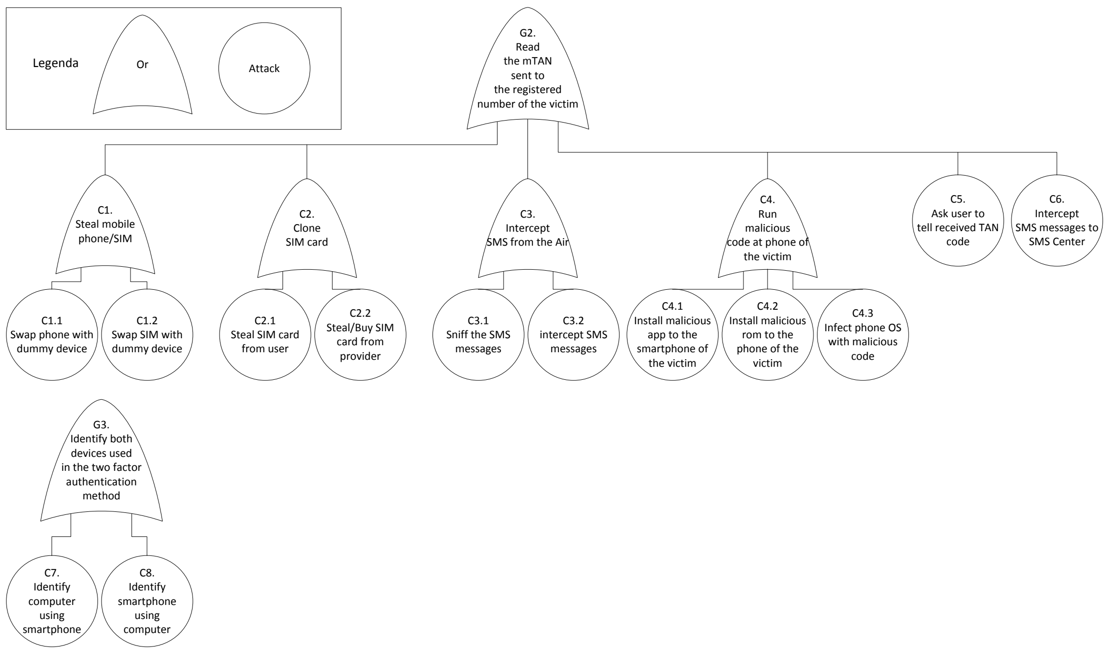
Researching if an operating system is vulnerable is not enough as for the attack it is important the attacker has access to the same mobile device as the computer.
From this conclusion we now can define the following goals:

- G1. Retrieve the username & password from the victim
- G2. Read the mTAN sent to the registered number of the victim
- G3. Identify both devices used in the two factor authentication method

The username & password will likely be used on the computer of the victim. As we assume computers are not safe we will not research if a computer can be infected with malware. Another option to gain bank account credentials is buying them from criminal people. The report about The Economics of Online Crime [20] shows that this kind of bank account credentials can be bought for about $10 to $100. Due to our assumptions we ignore this goal and only research goal 2 and 3. The Attack Trees to achieve these goals are constructed in chapter 5.2.

5.2 Attack Trees

The Attack Tree from this chapter is used to determine attack vectors to analyze the capabilities of the attacker to achieve the goal. This Attack Tree is based on literal study and brainstorming sessions.



Legenda
Or
Attack
G2. Read the mTAN sent to the registered number of the victim
C1. Steal mobile phone/SIM
C1.1 Swap phone with dummy device
C1.2 Swap SIM with dummy device
C2. Clone SIM card
C2.1 Steal SIM card from user
C2.2 Steal/Buy SIM card from provider
C3. Intercept SMS from the Air
C3.1 Sniff the SMS messages
C3.2 intercept SMS messages
C4. Run malicious code at phone of the victim
C4.1 Install malicious app to the smartphone of the victim
C4.2 Install malicious rom to the phone of the victim
C4.3 Infect phone OS with malicious code
C5. Ask user to tell received TAN code
C6. Intercept SMS messages to SMS Center
G3. Identify both devices used in the two factor authentication method
C7. Identify computer using smartphone
C8. Identify smartphone using computer

5.3 Probability attack vectors

In this chapter we will research the probability of the attack vectors C1 – C9. The attack vectors are divided into the different goals:

- Read the mTAN sent to the registered number of the victim
  Attack vectors: C1 – C6
- Identify both devices used in the two factor authentication method
  Attack vectors: C7 – C9

For each approach we will determine the probability of the attack vector. This probability will be used in chapter 5.4 where we will select the attack vector which will be further researched with an risk analysis.

C1 Steal mobile phone/SIM

When the attacker steals the mobile phone or SIM card from the victim he could access the mTAN. However it is likely that the SIM card is secured with an secret PIN and the victim will block his SIM at the telecom provider after he finds out it is stolen. This can be prevented by swapping the telephone/SIM with an identical one with a different phone number. This will extend the time which is available for the attacker to receive to read the mTAN.

Probability
This attack is not very likely because the attacker has to have physical contact with the victim and must be able to steam/copy the mobile phone/SIM.

C2 Clone SIM card
The SIM card can be cloned by the attacker to receive SMS messages with a duplicate SIM. To clone the device the IMSI and the authentication key (Ki) is needed. Expensive hardware is needed to dismantle the SIM and retrieve this number. This is because modern SIM cards are secured [21] to prevent cloning. Besides this difficulty, temporary access to the SIM card is also needed.
An alternative way is to steal/buy an SIM card from the provider. The provider can couple a number from its provider with another SIM which Is called a dual SIM. When the dual SIM is sold to the attacker the attacker could use this to receive the SMS messages. A dual SIM can simultaneous be activated but as they are designed that only one SIM can be in operation at a time a SMS message can only be delivered to one SIM.

Probability
It is unknown how much it will cost for an attacker to illegal clone a SIM of the victim. It is likely that it will be expensive when the employee is traceable because he could his job due to this.

Also a cloned SIM does not guarantee that the mTAN will be received at the cloned system. The network will only send the message once to a SIM. If both SIM cards are enabled the factor luck and last used SIM will determine which

SIM card will receive the message. This is encountered from own experience by using a dual SIM.

C3. Intercept GSM/telephone signal over the air
SMS messages are sent through the air encrypted by the A5/1 encryption when using GSM and the A5/3 encryption when using UMTS. A5/1 encryption can be broken by using hardware from $15 [22] [23] [24]. The disadvantage of this approach is that the attacker has to localize his victim and connect to the same base station. To be able to perform this attack the base station where the victim currently is logged onto must be found. When in range an attacker can sniff and decrypt the messages from the air or setup a honeypot to perform a man in the middle attack [25].

Probability
This attack is not very likely because the attacker has to be in range at the time of the attack. The costs for the hardware are low when the victim is connected to the GSM network as this encryption can easily be cracked.

C4 Run malicious code at phone of the victim
In chapter 1.3 we have explained that there is an increase in mobile malware especially for the Android platform. This existence of malware however does not guarantee that the attack goal can be achieved. Security mechanisms in the design of the platforms prevents the capabilities of malware.

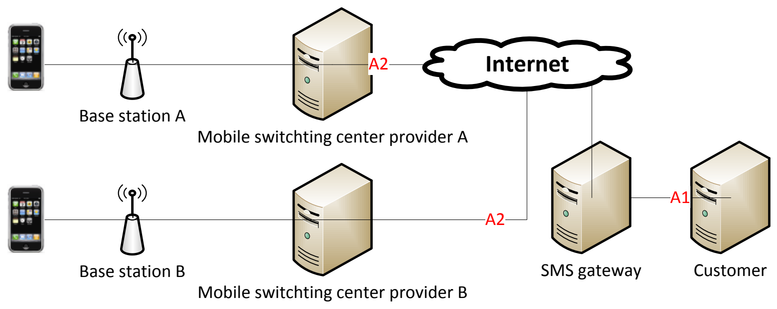
Probability
Because malware attacks are (almost) always possible to run automatically, it could be performed at large scale. The attacker does not have to be physical in range and the costs are low if the attacker can develop the malware himself. Alternatively an attack kit can be bought for around 25$ [8] which will help the attacker in a "click and go" way to install the malware.

C5. Ask user to tell received mTAN
A non-technical approach to achieve the goal is simply ask the user. We decided to exclude phishing attack in our research but we are investigating the case to show what the probability of the attack would be. Banks are doing their best to warn their users for this kind of phishing.
In the case of ING (section 3.5) the mTAN message begins with the following warning text: "Never give your TAN-code to someone else".
Second: people are still very distrustful when it comes to online or mobile banking [26].

Besides the distrustful, some people does not know better and will give the mTAN to the attacker. Due to the time limit of a mTAN this has to be done within a certain time. When an attacker makes a false payment, a mTAN will be sent to the victim. This mTAN has to be entered in a short amount of time (not public available) before it will expire. As most phishing attacks are performed using email [27] the time between the delivery of the email and the reply of the victim with the mTAN will be too long to make this approach usable.

Probability
As explained earlier in chapter 5 phishing will always exists. It is likely this attack will occur but phishing is out of scope of this paper.

C6. Intercept SMS Messages to SMS Center
As described in chapter 4.1 SMS centers mostly communicate with each other through the internet. When an attacker wants to intercept these messages he will either has to intercept the internet traffic between the service who sent the SMS and the SMS Center (A1) or between the SMS Center of the service and the SMS Center of the provider of the victim (A2). In figure 8 these attack points are displayed as A1 and A2.



Base station A
Mobile switchting center provider A
A2
Internet
Base station B
Mobile switchting center provider B
A2
SMS gateway
A1
Customer

Figure 8

When an attacker wants to intercepts these messages it is required the attacker is capable of sniffing the internet traffic at specific connections. Traffic between these centers is not encrypted as explained in chapter 4.1.2. To sniff the traffic the attacker has either to hack into the global infrastructure of the internet or the network of the service. Only when the network of the service could be hacked there would be fairly more easy methods available in stealing money/data/services as intercepting the SMS message.

Probability
This attack is not very likely as attacks to the internet backbone for sniffing is an attack which is not very common. Also the hacker has to find out in which data center the SMS gateway is located of the service to sniff the traffic which is not likely to occur.

C7. Identify computer using smartphone
In this attack the attacker has installed an malicious app at the smartphone of the victim and is using this smartphone to identify the computer of the victim.

Probability
Because iOS devices are required to connect the device to a computer, it is

likely the smartphone would contain identification information about the computer.

C8. Identify smartphone using computer
This approach is the same as C7 but instead of assuming the smartphone is compromised, the **computer** is compromised.

Probability
As described in C7, an iOS device is required to connect with the computer. As a compromised computer has more capabilities to access information using this connection, we think this attack is very likely.

5.4 Selecting attack vectors

For a successful attack the attacker must achieve attack goals G2 and G3. The most likely used attack vectors are:

- C4. Run malicious code at the smartphone of the victim
- C7. Identify computer using smartphone
- C8. Identify smartphone using computer

Attack vector C4 could be achieved by successfully completing one of these attack vectors:

- C4.1 Install malicious app on the smartphone of the victim
- C4.2 Install malicious rom on the phone of the victim
- C4.3.Infect phone OS with malicious code

Because of the limited available time it is not possible to perform the risk analysis on all attack vectors. This short analysis gives an idea that achieving the goals through these attack goals is likely to be performed by an attacker.

6 Security mechanisms & vulnerabilities

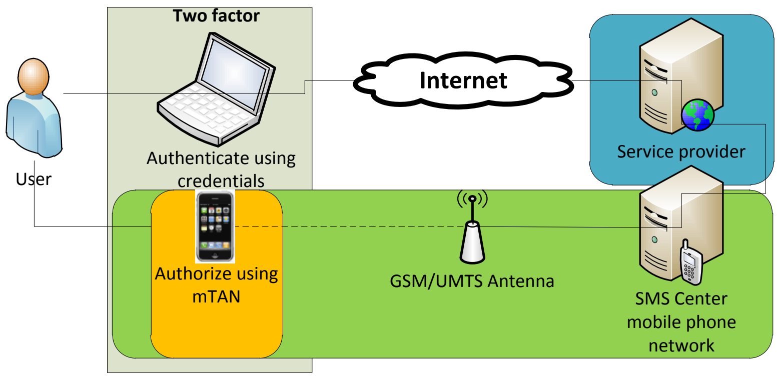


Figure 9

In this chapter we will examine the security mechanisms & vulnerabilities of the following technologies shown in figure 9:

- Section 6.1 - Mobile phone network
The items which are covered in this section are marked in the green section in figure 9. The security mechanisms of the mobile phone network are designed to protect the communication between the telephone operator and the smartphone.

- Section 6.2 - mTAN
The security mechanisms of mTAN protects malicious transactions. This section is marked in the blue area in figure 9.

- Section 6.3, 6.4, 6.5 - iOS and Android
The security mechanisms of iOS and Android protects sensitive information (like a mTAN) for the attackers. This section is marked in the yellow area in figure 9.

Because iOS and Android are both mobile operating systems we will also try to find security mechanisms and vulnerabilities, which are valid for both operating systems in section 6.3. In section 6.4 and 6.5 we will perform the research for the individual platforms.

The list of the security mechanisms & vulnerabilities in this chapter are defined by analyzing the Attack Trees from section 5.2. Security mechanisms & vulnerabilities which are related to the goal in this model are found and described using literary and empirical research. For the security mechanisms, vulnerabilities were found which could be abused to bypass the security mechanisms. Besides these related vulnerabilities some vulnerabilities were also found which would help provide the goals of the attacker, Read mTAN & Identify device, but are not secured by any security mechanisms whatsoever.

This chapter helps understanding some of the security mechanisms and vulnerabilities in the selected techniques. This chapter will answer the sub research question:

"Which security mechanisms and vulnerabilities are in place?"

6.1 SMS

In this section we will describe the relevant security mechanisms and vulnerabilities found for the technique: SMS.

6.1.1 Security mechanisms

S1. Encryption SMS over GSM
GSM is the 2G standard which mostly operates aside the new deployed 3G (and 3,5G) networks. In 2011 4 billion people used GSM as theirs communication protocol.
When SMS is being used over the GSM protocol, the SMS messages are secured with the A5/1 encryption standard [24]. The design of this encryption is not published, but the design was reverse engineered in 1999 by Marc Briceno [28]. The encryption resists on 52 bit encryption using cipher key streams. The master key is exchanged only the first time in a session the mobile phone is connected with the network and afterwards sessions keys are used. This makes it hard to intercept data because the key is continuously changed.

S2. Encryption SMS over UMTS
The newer UMTS/3G network make use of two different parts for the security systems. KASUMI is used for encryption and Milenage is used for authentication [29]. The encryption is greatly increased by using a 128bit key with a block size of 64 bits. Almost all phones are nowadays equipped with both GSM and UMTS capabilities.

S3. Security design SMS Center
The security used between the SMS centers is not publicly known. What we do now, is how companies can communicate with these SMS centers to deliver SMS messages.
Most SMS centers provide several communication methods like a direct telephone connection or an internet connection. A SMS center could use a technique like SSL to encrypt the traffic.
Due our own experience we have encountered that most SMS centers do not provide very strong encryption/authentication methods because of the overhead. In our own experience we mostly encountered IP whitelisting to secure the communication.

6.1.2 Vulnerabilities

V1. GSM encryption can be broken
There are several attacks known to the A5/1 encryption. Research [23] showed that a time-memory tradeoff attack was possible which allowed to construct the secret key. The disadvantage of this attack was that it was very time consuming and needed a lot of data to be successful.
In 2005 the attack was further optimized with the result that decryption of GSM traffic could be done in less than a minute [30].
To make use of the A5/1 encryption attack, you needed the rainbow tables for decryption and hardware which has an total cost of $50.000. In 2009 this changed by the publication of the rainbow tables needed for decryption and the release of the USRP device. The USRP device is capable of intercepting the GSM traffic in a much cheaper way because it only costs $600 dollar.
In December 2010 a security group showed at the Chaos computer club congress that similar attacks were possible by using a $15 smartphone [21]. However the group did not release their software but parts of the libraries used in the software are public available from osmocomBB [31]. When an hacker has the technical skills it would be possible to use this vulnerability in a very cheap way.

6.2 mTAN

In this section we will describe the relevant security mechanisms and vulnerabilities found for mTAN.

6.2.1 Security mechanisms

As mTAN is not really a standard, it is difficult to research the security mechanisms because each service provider use it in a different way. In this chapter we use the case of the ING bank which is introduced in section 3.5.

S4. Generated
Because the mTAN is delivered when a transaction has been created it is possible for the ING to generate a mTAN based on transactions details and some secret (random) data. Transactions details could be the target bank account number, amount of money and a timestamp. These details helps authorize the transfer as it is impossible to abuse the mTAN for other transactions. Another transaction could be a transfer to the bank account of an attacker.

S5. Unpredictable
The mTANs generated by the server are not (easily) to predict by the attacker. As explained in the previous section the mTAN is generated with a random number. Due to this randomness it becomes very difficult to predict the mTAN.

S6. Only valid in period
When the mTAN is created this mTAN is also saved at the server of the ING. Besides the transaction details and the mTAN the timestamp will also be saved. This way the ING can validate the mTAN has not expired yet. It differs from service providers but according to our own experience the expiration time is about 5 minutes at the ING. This security mechanisms force the attacker to operate in a limited time span.

S7. Transaction details visible
As the mTAN is generated with transaction details it also became possible to show these transaction details to the victim. As details like the amount of money are included in the SMS message where the mTAN is encapsulated with, the victim could notice malicious transactions. In the case of phishing it is likely the victim will see fake transaction details at the computer screen. As these details are also showed in the SMS message the chance is higher a victim will notice the malicious transaction.

6.2.2 Vulnerabilities

V2. Man in the middle attack
With the mTAN security mechanism a man in the middle attack is still possible. The attacker could setup a fake banking site and convince the victim to access this site instead of the official. When the victim thinks he is doing a valid payment the attack could change the bank account number and in some cases the total amount.
In the case of the ING Bank this total amount of money will be noticeable in the SMS message of the victim (as the message is not likely tampered). When this amount will be changed the victim could get suspicious and aborts the money transfer. However not every service is including transaction details into the SMS message.

When the service is including several transaction details some subtle changes are possible like changing the value of 20,97 to 2097,00 which are more likely to be undetectable by the victim.

6.3 iOS and Android

In this section we will describe the relevant security mechanisms and vulnerabilities found for the operating systems iOS and Android. As both operating systems uses some security mechanisms and vulnerabilities which are alike we have combined these in this section to avoid repetitions.

6.3.1 Security mechanisms

S8. App sandbox

A sandbox is a secured environment inside the global system of an OS. It provides limited resources to the app as a subpart of the whole system.

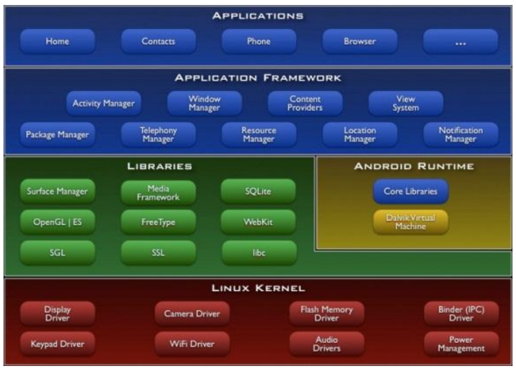


Figure 10

Figure 10 shows Android consists out of several parts [32]. Everything runs at top of the modified Linux kernel which is derived from the Linux 2.6 kernel and which is responsible for all the low level OS operations. On top of this OS pre-installed and user-installed apps run at the device.

Android implements the sandbox security mechanisms by using the sandboxing security mechanisms from UNIX. In UNIX every object in the file system has an owner, group and file mode. This file mode contains the read, write and executes permissions for the owner, group and other users of the file system. Because Android uses this mechanisms, every app has an owner user which has its limited access to the system. If an app wants to access data written in the folder from another app the user has to be included in the signing process. This way two apps can run as the same user to share system resources. When an app wants to communicate outside his sandbox communication has to be done through an API.

iOS also runs inside a sandbox as explained in the Apple iOS 4 security evaluation report [18]. It differs from the Android sandbox as it does not use the UNIX user security model to secure the system. In iOS all apps are running

as the UNIX user "mobile". Instead of using the UNIX sandbox model like Android, it uses the kernel extension "*Sandbox.kext*" developed by Apple.
Every application is installed into an "Application Home Directory" which is a separate location in the file system. The app can use an API to access resources from the system.

S9. Inter-app communication
Both iOS and Android offer solutions for inter-app communication. This could be useful to share data like a street address which have to be opened with another app at the device like Google Maps. The inter-app communication of Android is researched by Berkeley [33] which showed us the following information:
An Android app can publish a component which is capable of receiving communication. This publication has to be done through the manifest file which is a settings file for Android where settings like permissions are stored. When an app has published this component another app can send data to this app.
Another way of sharing data is using or publishing a "*Content Provider*" [34]. This "*Content Provider*" could be seen as some kind of database with its own read and write permissions for the shared component.

iOS uses a different approach for sharing data between apps. When an app wants to communicate with another app this is possible through special URI schemes [35]. Every app can register its own URI and use it for communication. Example: when appA wants to communicate with appB the appA could call the url: "appB://data=true".
Another way of communicating between apps in iOS, is to register a file type with an app [36]. This way this app will be launched and the content of the file will be sent to the app. It is not possible to register every file type as some are reserved for the iOS system.

S10. Code signing
Android & iOS requires apps to be signed.. This helps the user to identify the author of the app which is used for the permission model and updates. Unsigned apps cannot be distributed to the Android Market and App store and cannot be installed to the device.

The signing will guarantee that the code installed to the device is not modified in any way. After the app is developed the developer can sign this app by himself with the private key from his certificate. This certificate can be obtained from Google or Apple by registering as a developer.

6.3.2 Vulnerabilities

V3. Relation between developer and apps is not clear
The developer of an app is not clearly visible in the repositories where the app is downloaded from. When a developer offers a free and a paid app the user could check this by looking up the owner of the app. As the apps of the same developer are not related to each other this is a vulnerability as an victim can easily mistake downloading the wrong app.

V4. The smartphone is not secured by a firewall
The iOS and Android system does not have a built-in firewall for regulating incoming and outgoing connections. This way an app can communicate without the user is capable of detecting or blocking this kind of communication.

V5. The smartphone is online
By default iOS and Android will be connected to the internet (when not roaming). The internet connection could be entirely disabled in the OS but as most functions of the device require an internet connection it is not likely that it will be disabled.
Most phone providers assign an internet IP address to the connected. As most phone providers do not block any traffic to this IP address, the device is directly connected to the internet. In 2009 this vulnerability is abused in combination with another vulnerability to gain access to iOS devices [37].

V6. Apps can run as background process
In iOS and Android apps can run as a background process. A background process in iOS cannot access the same resources as when it is active but there are enough resources available to execute malicious code.

6.4 Android

In this section we will describe the relevant security mechanisms and vulnerabilities found for the operating system Android.

6.4.1 Security mechanisms

S11. Permission model
As described in section 6.3.1 the apps runs in a sandbox for security reasons. In this sandbox the app cannot access any resources outside this environment. Apps which cannot communicate with the internet or access information from the mobile device are not very useful to the user. As full access to the device would bring great risk Android implemented a capability-based permission model.
Each app is included with a manifest file where the needed permissions are stored. When a user installs the app the user must grant the app to use all of these permission. It is not possible to only allow or disallow certain permissions. Agreeing with the permission is only possible at time of installation of the app.

It is not possible to allow or deny certain permission every time the app needs a permission.
There are 122 different permissions [38] which an app can use. Not every permission can be used by a developer because 40 of these 122 permissions can only be used by the manufacturer of the phone. This is guaranteed by the fact that when an app wants to use one of these 40 permissions the app must be signed with the same key which is used to sign the platform.

The system which controls these regulations is called the Binder Interface. For these regulations Android uses the open source software OpenBinder [39].

S12. 3rd party installation source restriction
An Android device has an option which allow or disallow installation of apps from another location then the Market. This is a switch which could always be switched by the user without the need the device is rooted. As the switch blocks installation from 3rd party locations by default this secures the device.

S13. Controlled app distribution
When a developer wants to release an app for an Android device he has to follow certain rules/guidelines. The app has to be developed in Java, C or C++. As the app runs inside a sandbox the API from the NDK/SDK has to be used to gain access to functionality from the device.
These developed apps can be installed to the device by running the installation file at the device but most apps are distributed through the central Android Market repository. This repository can be accessed in two ways:

- Market
The Market is a pre-installed app at an Android device. This app can be used to search and install apps from the Android Market. It is only possible to install an app to the device which the Market is accessed from.

- Web Market
The Web Market can be accessed using a web browser from a computer. After authentication the apps from the Market can be remotely installed to the devices which are coupled with the used Google Account.

Before a developer can upload an app to this Market he has to be registered as a developer by providing their personal information and pay $25 registration fee. The developer also has to agree the distribution agreement [40] which has some rules about which software may be uploaded to the store.

Because the app is distributed in a central way, Google is able to have an overview of most apps which are installed at Android devices. When Google receives reports about a malicious app they can remove the App from the Android Market. After removal from the Market the security mechanisms S12 also makes it possible to remove the installed app from devices which already downloaded this app.

S14. Remote app installation & removal

Android Market is a central repository which stores Android applications. Authentication to this Market is provided through a Gmail Account (for details see the security mechanism S16).

The Android Market service condition [41] states that Google has the right to remotely remove or install apps at an Android device. The remote install feature is used to push updates, or gave the ability to the user to remotely install an app which can be done from the Google Web Market.

The remote removal feature is used to remove apps from devices which violates the Android Term & Conditions. As apps are not reviewed before granting access to the Market, they use this method to remove malicious apps from the Market and the Android devices afterwards. As far as we know Google can only remove apps which were installed through the Market and not through 3rd party Markets.

S15. Code obfuscation

One technique to make the decompiled code from another application less useful to the attacker is obfuscation. Obfuscation can protect the logic from the app and "hide" readable code to an attacker. Obfuscation makes it difficult to find the code which an attacker wants to infect with malicious code. Obfuscation can be done in two different kinds of levels [42]. In the first level, all names in a source of methods, classes, variables and other identifiers will be renamed. The second level will also add random methods with no functionality. Research showed [42] that only the first level is used in smartphone apps. To obfuscate the code several tools could be used. In the default NDK/SDK from Google, the tool ProGuard is included to perform this kind of obfuscation. As ProGuard is included in the Android Build System but not enabled by default we can assume that most apps in the Market won't be obfuscated.

S16. Antivirus

Antivirus software is a way to detect and prevent the installation of malware. Android offers multiple antivirus software but Lookout is one of the most popular systems as its installed on about 30.000.00 devices.

Antivirus on the smartphone is currently not very effective. The antivirus is an ordinary app which runs within the same sandbox as another app. As the sandbox prevents an app to access files from another app it is impossible to scan the files from the app. A possibility is to scan the installation files from an app as they are accessible with the right permission. When a(n) (malicious) app is started this is reported in the Android log. The permission "read logs" gives an antivirus app access to this log to recognize the malicious application by its behavior or way.

Due to this mechanisms the antivirus cannot detect suspicious behavior from an app but only detect blacklisted apps.

S17. Lockscreen

The lockscreen is by default a mandatory login security mechanism which require the user to enter a 4 digit PIN or draw a pattern to gain access to the device. This lockscreen has to be completed after the device wakes from standby mode. In Android 2.3 this security mechanism is mandatory and cannot be disabled by default. Some phone manufactures however has chosen to remove and disable this security mechanism.

S18. Google Account
The Google Account is a user account to provide access to services from Google. An Google Account is identified by an e-mail address which can be a Gmail address or any other address. A Gmail account is automatically a Google account but it is also possible to register a Google Account with a non Gmail address.
The email address in the Google Account is used for identification only and not authentication. Authentication has to be done with Google with the chosen password and/or a two factor mTAN security mechanism.
After creation the Google Account can be used to access a broad range of services from Google. In the scope of the research these services are Gmail, Backup and the Market. When a user wants to use the Market a Gmail Account is mandatory. A Google Account with a non Gmail address is not capable of using the Android Market.
Coupling the Market with the Google Account also register the device to a list of devices of the account. This registration is used in the Remote app installation (S12). When a user installs a paid app from the Market the password has to be entered onto the device. If the app is free and/or downloaded from an alternate location the password is not needed.
Google has some built in security mechanism when suspicious behavior is performed with the account. This includes faulty login attempts or logging in from an different location. The user will be asked to authenticate himself with a mTAN or by answering a secret question. When exactly these security mechanisms will be triggered is not known.

6.4.2 Vulnerabilities

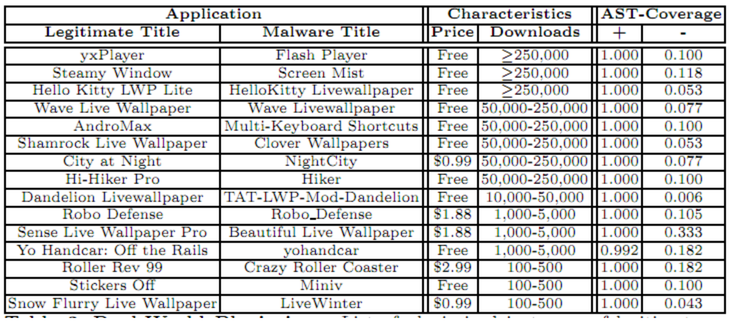
V7. App can run as service
Android has a special kind of component named a service which is responsible for running tasks in the background. This kind of service is , except for the GUI, not limited in any way. It could be abused by a malicious app to run hidden in the background.

V8. Apps are not researched before uploading to the Market
Apps uploaded to the Market are not checked by Google for malicious code. Google fully trusts the security model of the smartphone to prevent malicious apps. As the code is not checked an malicious app could be uploaded to the Market.

V9. Apps can be decompiled and recompiled with malicious code
The source code of Android apps can almost fully be decompiled to the original source with tools like dex2jar [43] and a Java decompiler [44]. With the

decompiled code, the attacker can add his malicious code in an easy way. Normally an attacker would search for the class which is fired at a certain event to execute the malicious code. With the decompiled app it is possible to add a class next to the existing decompiled classes with malicious code. As the decompiled app also includes the manifest.xml which can be used to register a class to an event this can be used to execute the code in the class. As the logic of the original source code is not needed it does not matter if the app is obfuscated (S13. Obfuscation).
Standard tools like Apktool [45] offers the solution to decompile, add the malicious code, and recompile the Android app.
Using these tools it should be easy for an attacker to (automatically) retrieve popular apps from the Market, inject the malicious code, change the manifest.xml and sign the app with a duplicate fake name.

Figure 11 shows a small list of cloned instances of legitimate apps that have been found in the Android Market.



Application		Characteristics		AST-Coverage	
Legitimate Title	Malware Title	Price	Downloads	+	-
yxPlayer	Flash Player	Free	≥250,000	1.000	0.100
Steamy Window	Screen Mist	Free	≥250,000	1.000	0.118
Hello Kitty LWP Lite	HelloKitty Livewallpaper	Free	≥250,000	1.000	0.053
Wave Live Wallpaper	Wave Livewallpaper	Free	50,000-250,000	1.000	0.077
AndroMax	Multi-Keyboard Shortcuts	Free	50,000-250,000	1.000	0.100
Shamrock Live Wallpaper	Clover Wallpapers	Free	50,000-250,000	1.000	0.053
City at Night	NightCity	$0.99	50,000-250,000	1.000	0.077
Hi-Hiker Pro	Hiker	Free	50,000-250,000	1.000	0.100
Dandelion Livewallpaper	TAT-LWP-Mod-Dandelion	Free	10,000-50,000	1.000	0.006
Robo Defense	Robo_Defense	$1.88	1,000-5,000	1.000	0.105
Sense Live Wallpaper Pro	Beautiful Live Wallpaper	$1.88	1,000-5,000	1.000	0.333
Yo Handcar: Off the Rails	yohandcar	Free	1,000-5,000	0.992	0.182
Roller Rev 99	Crazy Roller Coaster	$2.99	100-500	1.000	0.182
Stickers Off	Miniv	Free	100-500	1.000	0.100
Snow Flurry Live Wallpaper	LiveWinter	$0.99	100-500	1.000	0.043

Figure 11

V10. App can write file to external storage

In section S9, the permission model, we have explained a permission exists to gain access to the external storage. As this access regards full read/write access it is possible for a malicious app to write a malicious executable to this storage. When a computer is connected with the device it can share its external storage with the computer. When this malicious executable is present at the external storage this could infect the host computer.

V11. Lockscreen can be disabled

The Lockscreen (S15) is a security mechanism which is mandatory at the Android platform. From the Market several apps could be downloaded to disable this security mechanism with an app like NoKeyGuard.

V12. Install free app without password

Access to the Android Market is provide with a Gmail account. The first time a user access the Market he must provide the credentials to gain access. By default these credentials are stored in the phone for further access.

When a paid app is installed this password has to be reentered to prevent unauthorized purchase of an app. For free apps this password is not needed. When the attacker has physical access to the device, this vulnerability could be abused to download and install a malicious app from the Market.

V13. Lockscreen pattern can be retrieved
The lockscreen (S15) uses a pattern or a security code to secure physical access to the OS of the device. When a pattern is used, research [46] showed that in about 90% of the cases the pattern could be retrieved. This is done through the grease points which stays remained at the surface of the screen after a user has used the lockscreen.

V14. Installation of apps cannot be disabled
An Android device is always capable of installing apps. Some people don't want this functionality as they don't need it or want to disable it for security reasons. It is not possible to disable this functionality in the device to keep it original with no apps installed.

V15. Any app can be signed by anyone
In section S11 we have explained the development process for Android. From this section it becomes clear that anyone can register as an developer to sign an app. As is does not matter which code is signed, it is possible to sign a cloned app from another developer.

V16. Root Android
Rooting an Android device is an technique to disable some security mechanisms and gain full access to the Android smartphone. The result is the same as root access on a Linux environment. The security policy of Android does not really limit the user in its capabilities thus rooting is not very popular. People with technical skills mostly root their phone because it allows them to make big modifications to the system or replace the main system rom with an alternative version. As most Android users does not have these technical skills it is likely that only a small percentage of the Android devices are rooted.

V17. Silent remote installation Web Market
The Android Web Market [47] provides the capability to remotely install an app to an Android device. When using this feature the chosen (malicious) app from the Android will be silently installed to the Android device. As no notification or confirmation is shown to the user when an app is remotely installed it could be abused by an attacker. When an attacker has the credentials of the Google Account from the device of the victim he can abuse this vulnerability to silently install apps to the device without the knowledge of the victim or physical access to this device.

V18. Execution of app triggered by event
When a malicious app is downloaded and installed to the device due to a vulnerability it will be useless unless the app will be executed. Execution could be done (accidently) by the victim or from an event. Android provides four

events [38] which can trigger an app to launch (a certain class). This way the app can be started when the device is started up or a SMS message is received. This vulnerability can be abused by the attacker to launch the app after installation to the device.

V19. App can be hidden from apps
Android provides a list of apps which are used by the user to launch an app and have a quick overview of the installed apps. If a malicious app is somehow installed to the device this app would become noticeable due this overview.

When the app is developed as a service or the "*launcher*" property is removed from the manifest it is possible to hide the app from the launcher menu. This way the malicious app will be hidden. The app will however remain visible to the installed apps menu. As this overview is located in the settings of the phone it is not likely a user will notice the app in this menu section.

V20. App can be installed from 3rd party location
Besides the Google Market it is possible to install an app from another (unsafe) location. BlackMarket is an example of a 3rd party location which offers a great amount of pirated apps. The Android system has a switch which can be turned on or off to allow installation from these 3rd party locations.

V21. App can read SMS messages from the device
S9 explains that a permission exists to read and receive SMS messages from the device. As private and sensitive data can be present in these SMS messages this can be dangerous to the user. When the received SMS message is captured by an event it can prevent visibility of the SMS message to the victim by cancelling al further events. This vulnerability makes it possible to capture and forward SMS messages to the attacker without being noticeable to the victim.

V22. Capability leak
As explained in S9 inter-app communication in Android is possible. Research [48] showed that this inter-app communication could be abused when not properly implemented. The research show that certain apps (pre-installed or custom) make it possible to access the inter-app communication channels to access recourses without having the permission for the resource. When an app leaks an interface to access SMS messages this vulnerability could be abused. As the research did not focus on this particular permission (reading SMS messages) we cannot determine if this vulnerability is likely to be abused thus we will not use in the risk analysis. The research introduced some tools which could be used in further research to research if the permission to read SMS messages is leaked.

6.5 iOS

In this section we will describe the relevant security mechanisms and vulnerabilities found for the operating system iOS.

6.5.1 Security mechanisms

S19. Permission model
iOS make use of the capability-based permission model but limits the resources which can be accessed. With Android and iOS an app can access for example the screen to display some graphic elements without any permission. If an app wants to have access to the internet this permission has to be granted in Android before the permission can be used. As explained in the permission model(S9) Android can access almost every resource of the device when it is granted by the user. iOS uses a different approach as it only have two permissions which could be granted by the user. For example access to the internet is granted by default but other permissions (like accessing SMS messages) is not possible. The only two permissions which can be granted are the permission to access location data or the permission to send notifications. Notifications are push messages about an app which are displayed at the device when the app is not active. For example this could be an incoming chat message. The permissions are granted by the user the first time an app is started. A user can always change the permission in the settings menu to revoke the granted permission for an app.
Another difference with the Android permission model is the way resources are accessed. For example Android can read all the photos from the library and use this for a representation in any way. iOS cannot access the photos directly but can delegate an "*ImagePicker*" which launches the iOS photo selector and only returns the selected file of the user. This way it is not possible for an app to collect all the photos from a user without any notice. This mechanism however is not implemented for every resource. The "*Address Book*" for example can be completely read without any notice.
This model limits the functionality of an app but also limits the ability to leak privacy sensitive information.

S20. Background processes
iOS has the functionality of multitasking at some devices (iPhone 3GS,4,4S). When an app is minimized it still can run as a background- or inactive process. In this mode, the app functionality is limited as it cannot receive input from the user and an inactive process cannot receive any events. For performance reasons a suspended app has the risk to be killed by the OS to free up memory. To enhance the chance the apps keep running and can access certain resources an app can have these special modes as stated in the Developer Guide [49]:

Audio
The app plays audible content to the user while in the background. (This content includes streaming audio or video content using AirPlay).

Location
The app keeps users informed of their location, even while it is running in the background.

Voip
The app provides the ability for the user to make phone calls using an internet connection.

Newsstand-content
The app is a Newsstand app that downloads and processes magazine or newspaper content in the background.

External-accessory
The app works with a hardware accessory that needs to deliver updates on a regular schedule.

In all modes an app can run any code but the mode determine which recourses can be accessed and which events can be received.

S21. Controlled app distribution
Just like Android, iOS has a controlled app distribution security mechanism. Apple has several developer programs to deliver apps to iOS devices. In this paper we focus on the non-enterprise developer program which is about develop apps that can be downloaded and installed by an iOS device through the App Store. The App Store is the main repository from Apple where Mac OS and iOS apps can be offered. In this paper we focus on the App Store where the iOS apps are offered. This store can be accessed from an iOS device with an Apple ID. If a user wants to download and/or pay an app the password of the coupled Apple ID has to be entered for verification or purchase.
The difference with the Android Market is that apps uploaded to this store has to meet the App Store guidelines [50] & license agreement before the app is added to the store. That an app follows these guidelines and license agreement is guaranteed because the app is reviewed before added to this store. This review is likely be performed by using automatic code testing to check if no direct file access or undocumented libraries are used. The app will also be launched by the tester to check if the accessed recourses are needed for the functionality of the app. When Apple wants to remove the app from the store after submission, this is possible but they cannot remove it from devices which already downloaded and installed the app.
Before a developer can upload the app to the Store, he has to be registered as an Apple developer. Personal information & credit card data has to be given to register the identity of the developer.

S22. Irreversible code
As Apps are developed in Objective C and compiled into the Mach-O format, decompilation to the original objective C source is not possible. As every executable can be decompiled into assembly a tool like IDA [51] can help to

generate some helpful objective C code out of it. The result can be used to show some secrets from the code but makes it hard to inject malicious code.

S23. Lockscreen
Apple has a lockscreen which is disabled by default. By default this lockscreen consists out of 4 digits but can be changed to an alphanumeric code. When, enabled this security mechanism has to be completed after the device is turned on/wake from standby or the screen is unlocked. The passcode from the lockscreen is also used to enhance the built-in hardware encryption by using the passcode [52].

S24. Apple ID
The Apple ID is the equivalent for the Google Account from Android. The Apple ID is used as an authentication system to access multiple services from Apple. An Apple ID consists out of a username (which can have the format of an email address), password and an email address. The email address is verified by an activation link.
As explained in S20 the only way to install an App to an IOS device is through the App store. Access to this App store is only provided with an Apple ID. An Apple ID can be disabled when suspicious behavior occurs. When this will happen is not publicly known.

S25. Antivirus
Just like Android, iOS does not have a built in antivirus solution. Several app developers tried to develop an antivirus app but they were all refused to the app store until July 2011. The reason why they refused an antivirus app is unknown but our guess is that they think their security model is good enough and they don't want to compromise on user experience. Antivirus mostly means the device is slowing down and it will drain more battery. The antivirus app which currently exists is called "*Intego*". This antivirus app encounter the same problems as the antivirus apps for Android: It runs inside a sandbox without special permissions. As explained in the permission model iOS offers less resources that can be accessed. The Android antivirus mostly relies on the temporary download location and the logbook to detect apps. The only possibility for an iOS app, is the file type registration as explained in S17. Using this mechanism the app can register itself with the allowed file types. When a user opens a file from the email or another location the user can choose to open the file with the antivirus app and it will be scanned for malicious software. As apps are reserved file types it is not possible to register an app installation file to open it with the antivirus app. Due to this reason it is impossible for an iOS antivirus app to scan for malicious apps.

S26. App installation restrictions
As explained in S20 Apple has a controlled app distribution mechanism. An important part of this is the security mechanism "*App installation restriction*". Apps for iOS can only be installed from the App Store. For enterprise and beta testing there are solutions that installation could be done directly to the device bypassing this store [53]. For beta testing the developer needs the unique

identifier of the iPhone to add the device to the beta section. The user can install the app through his computer and iTunes.
To develop an app which can be downloaded by anyone the app has to be installed from the App Store.

S27. Keychain
iOS offers a keychain to store data at the device in an encrypted way. The passcode from the S22 is used in the security algorithm to encrypt the data. It uses a SQLite database to store data as password, email accounts, Wi-Fi keys etc. An app can also use this keychain to store custom data. The security of this keychain is done through access groups. Each group has a unique group name andit can only access its own data. System applications (like the email, VPN etc.) uses the "apple" access group and can access all data which is coupled with this group.

S28. Backup encryption
Through iTunes it is possible to make a backup of the iOS device. This backup is stored in an unencrypted way to the hard-drive. As the backup contains sensitive information it is possible to encrypt the backup with a password. This password is different from the used passcode in S22.

6.5.2 Vulnerabilities

V23. SMS messages could be read from the Sandbox
As explained in the sandbox (S8) security mechanism, an app runs inside the sandbox but under an central user: "mobile". Because the SMS message app also runs as the user "mobile" the SMS messages are readable by any app.

V24. iOS has internal secret API
Apple provides an API/library to access resources at the device. By dumping the headers of the library, private methods are revealed and combined into a database [54]. These methods can be abused to access extra resources which can be abused for an malicious app. When an app is uploaded to the App store these private methods can be detected and the app will be likely rejected. However this vulnerability could be abused for easy access to sensitive data when the app is installed though an alternative way.

V25. iOS stores SMS messages unencrypted
iOS stores its SMS messages in an unencrypted way to the SQLite database of the device. If the database would be encrypted the decryption key to access the SMS messages should be somewhere stored. If the device is completely compromised this encryption will be useless.

V26. 10% of iPhones devices are jailbroken
The version which we are researching, iOS 4.3.3, can easily be jailbroken by using a USB connection with a computer or opening a special webpage at the browser of the iOS device. When the device is jailbroken all designed security mechanisms can be bypassed by a malicious app.

There are two reasons why a user would jailbreak his device. As explained in section 6.5.1 iOS security mechanisms limits the user and developer heavily which resources could be accessed or adjusted. This is called Psychological Acceptability [55] which can cause the users will search for a way to disable some security mechanisms.
The second reason is piracy. The App Store contains three times more paid apps then the Android Market [56] . This makes it interesting to the user to jailbreak their device to download the apps from 3rd party stores for free.
It is difficult to get a clear idea how big the Market share of jailbroken iOS devices is. There is more information available about the percentage of jailbroken iPhones. Several articles claims this number is around 10%. Because of the piracy we think the chance an iOS device is jailbroken is medium and this vulnerability could be abused.

V27. Apple ID can easily be changed
An iOS device is coupled with an Apple ID to access the App Store. When the App Store at the iOS device is used the password of the Apple ID is also needed for the download and installation of an app. The Apple ID of the device could easily be changed to one of the attacker as the password of the old Apple ID does not have to be entered. When there would be an malicious app available in the App store and the attacker has physical access to the device this vulnerability can be abused to install this malicious app.

V28. Device has OpenSSH installed with default credentials
When an iOS is jailbroken, the 3rd party app repository Cydia is automatically installed. Cydia does not contain any pirated apps but mostly apps which were not allowed to the App Store repository. Some software which installs the jailbreak does not only install Cydia but also installs the OpenSSH server. This server is started at boot of the device and have root access enabled with a default password. Besides this automatic installation of OpenSSH some users will install the OpenSSH server manually from the Cydia repository for remote administration. As a lot of users forget to change this default password it becomes possible to remotely control the device.

V29. iOS assembly can be structured viewed
Security mechanism S21 code explains that it is very difficult to decompile the source to the original objective C code. However, every programming code can be viewed in assembly so this is also possible for an iOS app. Tools like class-dump [57] can help in this process. As the source is hard to read and modified it is not likely an attacker will be abuse this vulnerability.

V30. Backup is unencrypted
Security mechanisms S27 showed there is a way to encrypt the backups at the computer. Because this setting is not turned on by default it is not likely it will be used by a user. Because the backup contains sensitive information this backup file could be read by malware at the computer.

V31. Serial number is visible
When an iOS device is connected to the computer it identifies itself as a "Portable Device". This device contains the serial number as metadata which can be read from any application on the computer. This way identification information can be read through malware.

V32. Lockscreen is disabled by default
The lockscreen from S22 is disabled by default. It is likely that a lot of people will not have this security mechanism turned on. This could be abused when gaining physical access to the device.

V33. Backup contains identification information
The backup file contains all the information which is stored at the iOS device except the passwords. For example the following information could be retrieved:
- Apple ID
- Installed e-mail accounts
- IP addresses of last connected Wi-Fi/3G networks

7 Risk analysis

In this chapter we will perform the risk analysis about the two goals with theirs sub-goals which were selected in chapter 5, the threat model:

- **Read the mTAN sent to the registered number of the victim**

  Sub goal: Install malicious app to the smartphone of the victim.

  The risk analysis will be performed by analyzing the sub goal for the operating systems Android and iOS. Because a malicious app is needed for the attack we have defined this malicious app as follows divided in must haves and nice to have for the attacker:

  Must have:
  - Access received SMS messages
  - Forward received SMS messages through the internet

  Nice to have:
  - Invisible to the user
  - Only read SMS messages when a SMS message is received by the device

- **Identify both devices used in the two factor authentication method**

  Sub-goal: Identify computer using smartphone
  Sub-goal: Identify smartphone using computer

  These sub-goals are selected in section 5.4 and are about identifying the device. In the risk analysis we will analyze how and if it is possible to identify the other used device when the attacker has already compromised one of the used devices. We make the assumption in this research that the victim will only use one computer and one smartphone for the transactions where the mTAN is needed. That means that when both devices are identified also those both devices will be used with an service provider which is secured with mTAN.

The risk analysis will be performed by using parts of the attack and defense tree [6] method. This method extends the original Attack Tree from section 5.2 by adding security mechanisms and vulnerabilities. The analysis will be performed by selecting each attack node form the Attack Tree and research what the possibility and the probability is that the attack node will be attacked. The found security mechanisms and vulnerabilities from chapter 6 will be used in this analysis. When an attack node can be attacked this does not automatically mean the goal could be achieved. As each attack node is related in a hierarchy tree with AND and OR nodes the outcome will be

determined by the result of an attack in combination with the relation to other nodes.
To make the results measurable, we have chosen to add some metadata. Attack Trees mostly uses money as a metadata to determine the probability of an attack. Because most of our attacks are software based money is not the most important factor in this research, we have chosen to classify the attacks with the following categories:

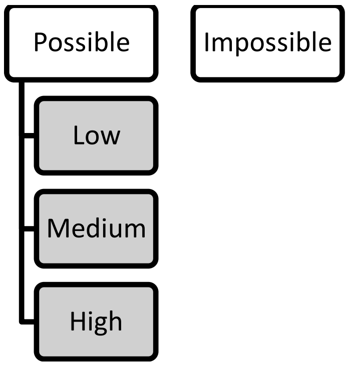


Figure 12

When the attack is possible the probability of the attack could be classified as: low, medium or high. The probability is measured based on the found security mechanisms and vulnerabilities. Because some attacks are not clear by the title itself we have chosen to add some description to the attacks where this is needed.

In a traditional risk analysis the risk is calculated as follows: *"risk = impact * probability"*. Because the main purpose of this risk analysis is to discuss the possible attack vectors and selecting the most likely attack vector we have decided to use the impact to determine the probability but not discuss it separately.

If a rooted/jailbroken device would make any difference, these attack nodes will be separately measured for both situations.

The result of the analysis will have multiple purposes:
- Answer to sub questions of the research question
By creating the attack & defense tree and measure the outcome the following sub questions can be answered:

> *"Which security mechanism prevents to achieve the end-goal?"*
> *"Which vulnerabilities helps to achieve the end-goal?"*
> *"What is the most likely attack vector to achieve the end-goal?"*

- Room for counter measures
When the weak points have become clear from the risk analysis, new counter measures could be introduced which could help securing the goal. For a good

analysis these counter measures should be added to the tree and the analysis should be performed again. As there is not enough time available to do this, we have chosen to propose the these counter measures as future work.

7.1 Smartphone OS

To research the possibility and probability for the Attack Tree: "*Install malicious app to the smartphone of the victim* " we have extended the Attack Tree which was selected in section 5.2. This Attack Tree can be found in appendix A. In this chapter we will research the possibility and probability for the Attack Tree. Because the possibility and probability of some attack vectors are the same for Android & iOS, these will be examined in section 7.1.1. In section 7.1.2 and 7.14 we will analyze the attacks for the Android and iOS platform. In the sections 7.1.3 and 7.1.5 we will use this results for the conclusion.

7.1.1 Android & iOS

In this chapter we will research the attack nodes from the Attack Tree separately for the platforms: Android & iOS.

A1. Communication with smartphone is possible through internet

Security mechanisms	*Vulnerabilities*
	V4. The smartphone is not secured by a firewall
	V5. The smartphone is online

Probability
For iOS there is no security mechanism which could prevent the access to the internet. The vulnerabilities V4 and V5 shows that a smartphone is online almost all of the time and there are no firewalls that can regulate this connection. The probability of this attack is: **High**

Psychological attack vectors

The following attacks, A12, A2 and A3 are psychological attacks. These attacks are path of the attack vector: “Victim wants to install malicious application” as shown in figure 13. We assume that the victim does not know the app is malicious. As A6 is depended on the platform this attack is examined in the corresponding OS chapter.

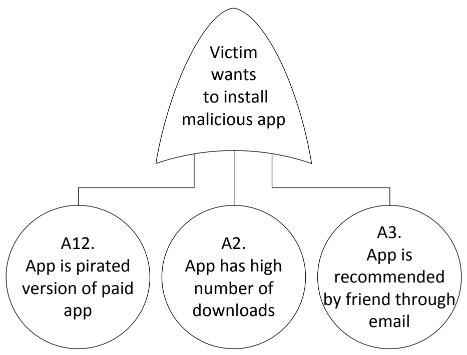

Victim wants to install malicious app
A12. App is pirated version of paid app
A2. App has high number of downloads
A3. App is recommended by friend through email

Figure 13

A2. App has high number of downloads

The Market has a “top downloaded” list of apps which is automatic shown when the Market is opened.

Probability

Most users who are searching for a new app looks at the most download section. Because it is likely the victim will download the app when it is popular ,the probability of this attack is: **High**

A3. App is recommended by friend through email

Probability

When an app is recommended by a friend (which could be achieved by a fake email from the attacker) it is likely that the victim will install the app. As it is also possible to link to an app in the Market/Store using a hyperlink in a message this could influence the victim to install the app. The probability of this attack is: **Medium**

A4. Victim does not care about permissions

Probability

When a victim installs the app, he already made up his mind that he wants to run this app. As users mostly agree app conditions without reading, because they do not care, it is likely the victim also does not care and will install the app. The probability of this attack is: **Medium**

A5. Attacker can upload malicious app to third party Market/Store
Third party Markets are apps which provide access to other repositories to download (pirated) app. There are several 3rd party Markets available but the biggest ones from Android & iOS are BlackMarket and Installous.

Probability
An attacker and even a user can upload a malicious app without any costs or registration to a 3rd part Market/Store. This app will likely not be checked for malware. The probability of this attack is: **High**

A6. Email is used at smartphone
Probability
The study by Google, Ipos and MMA [58] reports that in Europe in 2011 people use their smartphone for 70% of the time for email communication.
The probability of this attack is: **High**

A7. Spam filter accepts mail with app
Probability
Mails with binary attachments are often blocked by the mail provider. As the installation of an app is an executable, the chance is big the mail will be rejected by the spam filter. The probability of this attack is: **Low**

7.1.2 Android

In this chapter we will research the attack nodes from the Attack Tree separately for the platform: Android.

A8. Have (temporary) physical access to the OS
With physical access we not only mean the physical access to the device but also access to the OS of the system. This attack could be performed by loaning or stealing the device from the victim. After the device is in range for the attacker he has to gain access to the OS to be able to perform the attack.

Security mechanisms	*Vulnerabilities*
S17. Lockscreen	V11. Lockscreen can be disabled
	V13. Lockscreen pattern can be retrieved

Probability
The first part of this attack is that the attacker wants physical access to the device itself. The problem with the attack is that the attacker has to show himself to the victim. It is likely an attacker would try to avoid this situation.

When the attacker could conquer the phone, he also wants to gain access to the OS of the device. The security mechanism S17 shows that the OS makes it mandatory to secure the device with a lockscreen. The chance that vulnerability V11 can be abused is very small as this vulnerability only exists at a small percentage of the device but vulnerability V13 can be used in 90% of the cases.
Because the attacker has to show himself to the victim the probability of this attack is: **Low**

A9. Malicous app can run at smartphone
With this attack we mean that the malicious app, designed in the introduction of chapter 7, can be executed and perform the designed functionality. This attack analyses the fact if there are any security mechanisms which could prevent the execution of the app.

Security mechanisms	*Vulnerabilities*
S16. Antivirus	V14. Installation of apps cannot be disabled

Probability
The first demand for this attack is that the device can run an app (malicious or not). The vulnerability shows that it is always possible to install an app to the device. The antivirus security mechanism could detect the malicious app which could prevent execution. However only a small percentage have this security mechanisms installed. As the antivirus only detects apps which are marked as malicious instead of detecting suspicious behavior, the chance that the security mechanism will block the app is small. The probability of this attack is: **High**

A10. Smartphone can be controlled due to user installed app
With "controlled" we mean full access to the device with the ability to remotely download and install apps.

Security mechanisms	*Vulnerabilities*
S8. App Sandbox	V5. The smartphone is online
S11. Permission model	V4. The smartphone is not secured by a firewall

The first demand for this attack is a (internet) connection between the attacker and the victim. As the permission model does not provide a permission to open a TCP port, this means a direct incoming connection is not possible. As the smartphone is online and not secured by a firewall an outgoing connection can be used to make the connection. The second demand is that an app provides the capability to download and install an app. As this permission is not provided by the permission model and the app runs inside the secured sandbox this attack is: **Impossible**

When the device is rooted an app could run outside the sandbox, open the TCP port and provides the capability to remotely download and install an app. The chance this app will exists at a rooted device is medium. An example of this kind of app would be a SSH server which is a popular method for remotely administrating the device. Rooted this attack is: **Medium**

A11. App is installed using Market remote installation
In section 6.4.1 we have introduced the Google Web Market which can be used for remote installation of apps. This attack is about installing a app using this Market and the credentials of the victim.

Security mechanisms	*Vulnerabilities*
S18. Google Account	V17. Silent remote installation web Market

Probability
Vulnerability V17 shows that it is possible to remotely install an app to an Android device using the Gmail Credentials from the user (which are also used at the Android device). It is not possible to disable this remote installation feature which makes the vulnerability dangerous.

As explained in chapter 5 the attack will occur when one of the other channels is already compromised. In this section the compromised device is the computer. As Google tries to protect their account with several protection mechanism it is possible that the credentials cannot be used by an attacker. As the risk exists credentials can be leaked from an owner of an Android device the outcome of this attack is: **Medium**

A12. App is pirated version of paid app
In this attack we assume the pirated app is offered for free. This attack is achieved when a victim wants to install the pirated app instead of the original version

Security mechanisms	*Vulnerabilities*
-	V3. Relation between developer and apps is not clear

Probability
The outcome of this attack seems logic. However 57% of the Android Market contains free apps which is showed in the report from Distimo [56]. The paid apps that are available mostly also offers a lite version. When the victim is looking for this lite version it is possible he downloads the wrong app due to the vulnerability V3. The probability of this attack is: **Medium**

A13. Victim does not understand permissions
Android provides a big list of permissions. To make it more understandable for users they made some descriptions to the permissions. For the receive SMS permissions this description is as follows:

RECEIVE SMS
Allows app to receive and process SMS messages. Malicious apps may monitor your messages or delete them without showing them to you.

Probability
A warning exists in the description about malicious apps.
Most users do not know what malicious apps are, and they do not understand the warning it is likely that they will ignore it and agree the permission. The probability of this attack is: **High**

A14. Attacker can offer malicious app at official Market/store
For a successful attack, the attacker has to upload the malicious app to the Market and ensure it can be downloaded by the victim.

Security mechanisms	*Vulnerabilities*
S13. Controlled app distribution S14. Remote app installation & removal	V8. Apps are not researched before uploading to the Market

Probability
V8 shows that it is possible to upload a malicious app to the Market. In principle apps are available for an unlimited amount of time. However Android

can use the security mechanisms S14 to remove the app from the Market and remove it from the devices which already had installed the app. Currently Android does not make use of this mechanism to remove malware. We can assume the chance it will be removed is relative low. The removal will only be performed when users complain about the app so the goal for the attacker would be to make the malicious code as unnoticeable as possible. The probability of this attack is: **High**

A15. Smartphone accepts app as attachment in mail

Security mechanisms	*Vulnerabilities*
-	V20. App can be install from 3rd party location

Probability
The mail client from Android does accept all kind of mail attachments. As V20 shows an app can be installed from other locations besides the official Market like installation of the app through an attachment from an email message is possible. However a switch has to be turned on in the settings menu to allow installation from unknown sources. When users did not use pirated apps before, it is not likely this switched is turned on. The probability of this attack is: **Low**

A16. App can capture screen content
When an app can capture the screen content this could be used to capture the screen content when an SMS message is received. This screen could be forwarded to the attacker.

Security mechanisms	*Vulnerabilities*
S8. App Sandbox	
S11. Permission model	

Probability
The permission model provides a permission to read the framebuffer data which can be used to create a screenshot from the device. As this permission could not be used by a normal app the permission model prevents the screen to be captured. The permission is only prohibited to apps created by the manufacture of the device. The probability of this attack is: **Impossible**

When the device is rooted the permission could be used or direct access to the framebuffer data is possible. The problem is that the app has to capture the screen content when the SMS message is opened. This would require a high number of screen captures within an certain time period from the SMS message is sent to the device. This would mean a lot of data has to be sent to the attacker using the relative slow mobile internet connection of the device. The probability of this attack is: **Low**

A17. App can read SMS message from device

Security mechanisms	*Vulnerabilities*
S6. Only valid in period	V21. App can read SMS messages from the device
S8. App Sandbox	
S11. Permission model	

Probability
The vulnerability shows that there are 2 permissions which can be used to read the SMS messages from the device. The first permission provides access to the complete SMS inbox which can be used to discover old and new messages. The second permission provides an event which sent a new received message first to this app. Due to the time limitation of mTAN, as explained in S6, the attacker is only interested in new messages. This makes the event based permission the most suitable to use. As this permission is secured by the permission model the access is only possible when the user has agreed to the permissions. The probability of this attack is: **High**

A18. App can be decompiled, infected and be recompiled
When the attacker wants to offer a malicious app this attack helps by re-using an existing app to gain popularity and save time.

Security mechanisms	*Vulnerabilities*
S8. App Sandbox	V9. Apps can be decompiled and recompiled with malicious code
S11. Permission model	V15. Any app can be signed by anyone
S10. Code signing	
S15. Code obfuscation	

Probability
The security mechanisms S10 prevents the code of the app to be modified without breaking the signing and V9 shows the app can be decompiled to almost the original source code. Due to the completeness of the decompiled source code, it can be used to add the malicious code and recompile it to an malicious app. Code obfuscation could be used in the original source which will make the decompiled code harder to read. As the malicious code could be added to the decompiled code without the need of fully understanding the original code, this security mechanism does not really offer any protection. Before uploading the app back to the Market, the app has to be signed by a developer. V15 shows us that any app can be signed by anyone which means a cloned app from another developer could also be signed by the attacker. Due to the permission model, the permissions has to be extended needed for the malicious code when the original app did not had the needed permissions. The probability of this attack is: **High**

A19. App can be binary wise infected
Besides infecting an app by adding malicious code to the decompiled source, another method is infecting the binary of the application.

Security mechanisms	*Vulnerabilities*
S8. App Sandbox	V15. Any app can be signed by anyone
S10. Code signing	
S11. Permission model	

Probability
The security mechanisms S10 prevents the code of the app to be modified without breaking the signing. To infect the binary Dalvik bytecode [59] a technical skilled attacker would be needed. The modified code can be signed again to be uploaded to the Market. As most attackers are not that technical skilled, and a more easy attack is available, the probability of this attack is: **Low**

A20. New popular app is developed
The goal of this attack is that the attacker develops a new app from scratch. This new app has to be interesting to the users which would make it popular.

Probability
Developing an app which will gain a high rate number of downloads does not only takes a lot of time but also requires a unique idea. The Market offered 350.000 apps [56] in October 2011. This makes it difficult to create a popular app due to the high competition. Chances that an attack would develop a **popular** app by itself are: **Low**

A21. Victim can download malicious app from 3rd party location

Security mechanisms	*Vulnerabilities*
S12. 3rd party installation source restriction	V20. App can be installed from 3rd party location

Probability
As explained in V20 alternate 3rd party locations exists which mostly offers pirated apps. The security mechanism S12 prevents the installation of an app from these 3rd party installation sources. As this security mechanisms could be easily disabled from the settings menu we think it is possible to the security mechanism is disabled. As not all users will be familiar with this switch the probability of this attack is: **Medium**

A22. App can be hidden
When an app is hidden, this app can be abused to hide a malicious app at the device of the victim.

Security mechanisms	*Vulnerabilities*
S8. App Sandbox	V19. App can be hidden from apps
S11. Permission model	

Probability
The vulnerability V19 shows that an app could be hidden by removing the launcher attribute from the manifest file. This will only hide the device from launcher menu but not from the installed apps overview. This overview can be found in the settings menu which is not accessed a lot by a user. This makes the probability of this attack is: **Medium**

A23. App can send SMS without noticing the victim
When the malicious app has intercepted a mTAN, one method is to forward this using a SMS message to the attacker.

Security mechanisms	*Vulnerabilities*
S8. App Sandbox	
S11. Permission model	

Probability
When an app wants to send a SMS message this has to be done using the permission model. Because sending a SMS message usually costs money the victim is warned about this fact about this permission. It is likely the victim will also notice the malicious messages as they are probably visible at the bill of the victim. This makes the probability of this attack: **Low**

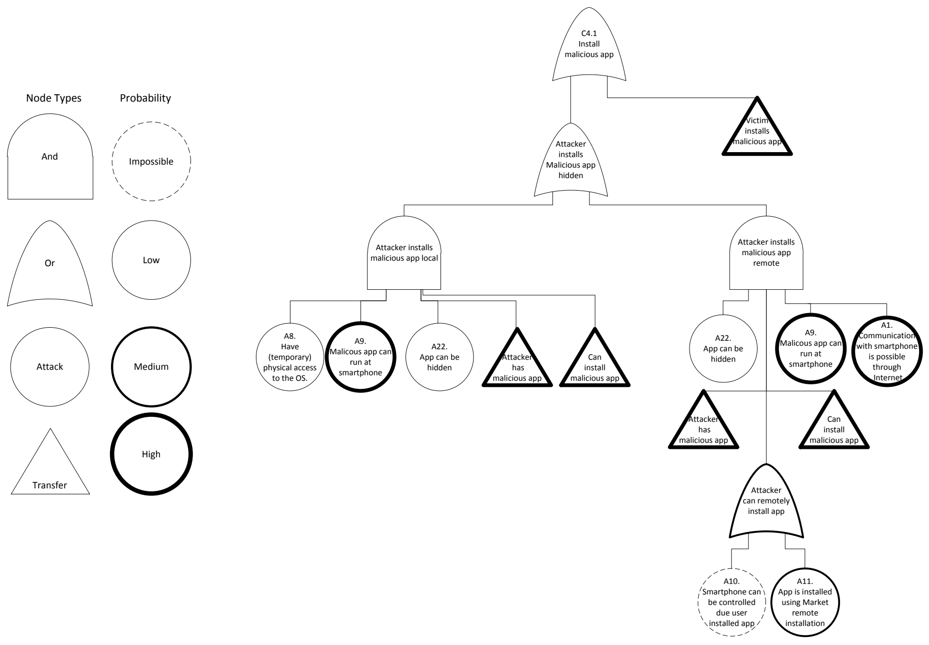
A24. App does not need permissions
This attack is about the situation that a app does not need a permission from the OS to access the needed resources.

Security mechanisms	*Vulnerabilities*
S8. App Sandbox	V22. Capability leak
S11. Permission model	

Probability
The capability leak vulnerability shows some preinstalled apps offer inter-app communication methods which can be used to access data. As there is no research performed in what way access to the SMS database is leaked in this way the probability of this attack is: **Low**

7.1.3 Conclusion

In section 7.1.2 we have researched for the Android platform the probability of the attack nodes from the tree: “Install malicious app”. In this chapter we will use this data in conjunction with the attack tree from appendix A. The outcome of this chapter gives an overview of the possible attack vectors. Besides the possibility the probability is added which gives an idea which attack vector will probably be used by the attacker. As explained in chapter 1 this research focus on the probability of an attack with the current security architecture. We assume the involved systems do not have any unknown bugs.



Node Types
Probability
And
Impossible
Or
Low
Attack
Medium
Transfer
High
C4.1 Install malicious app
Attacker installs Malicious app hidden
Victim installs malicious app
Attacker installs malicious app local
Attacker installs malicious app remote
A8. Have (temporary) physical access to the OS.
A9. Malicous app can run at smartphone
A22. App can be hidden
Attacker has malicious app
Can install malicious app
A22. App can be hidden
A9. Malicous app can run at smartphone
A1. Communication with smartphone is possible through Internet
Attacker has malicious app
Can install malicious app
Attacker can remotely install app
A10. Smartphone can be controlled due user installed app
A11. App is installed using Market remote installation

Figure 14

Figure 14 shows the attack is possible because the goal “Install malicious app” could be achieved by the attacker. The most interesting attack vector is where the app is installed remotely. With local installation the attack will consume far more time from the attacker as physical access is needed and the attack cannot be done in an automatized way. The remote installation can be automatized and the attacker will be more anonymous to the victim.
The remote installation feature from Android Market is the biggest risk which could be abused when an attacker wants to install the malicious app without the help of the victim.
When help is needed from the victim the biggest risk is because the malicious app can be offered at the Market. As this analysis shows the probability the attack could be performed is high, we have decided to not make the analysis for rooted device.

7.1.4 iOS

In this chapter we will research the attack nodes from the Attack Tree separately for the platform: iOS.

A8. Have (temporary) physical access to the OS

Security mechanisms	*Vulnerabilities*
S23. Lockscreen	V32. Lockscreen is disabled by default

Probability
The first part of this attack is that the attacker wants physical access to the device itself. The problem with the attack is that the attacker has to show himself to the victim. It is likely an attacker would try to avoid this situation.

When the attacker could conquer the phone, he also wants to gain access to the OS of the device. When the lockscreen is enabled, it prevents access to the device after it has woken from standby or it is powered on. Because the lockscreen is disabled by default we do not think it is very likely a user has enabled the lockscreen.
Because the attacker has to show himself to the victim the probability of this attack is: **Low**

A9. Malicous app can run at smartphone

With this attack we mean that the malicious app, designed in the introduction of chapter 7, can be executed and perform the designed functionality. This attack analyses the fact if there are any security mechanisms which could prevent the execution of the app.

Security mechanisms	*Vulnerabilities*
S8. Sandbox	
S19. Permission model	
S25. Antivirus	

Probability
The only security mechanism which could prevent the execution of the malicious app, is the antivirus. Due to the restrictions of the antivirus software it is not possible to scan a downloaded app for malware. The probability of this attack is: **High**

A10. Smartphone can be controlled due to user installed app

Security mechanisms	*Vulnerabilities*
S8. Sandbox	V4. The smartphone is not secured by a firewall
S19. Permission model	V5. The smartphone is online
S21. Controlled app distribution	V28. Device has OpenSSH installed with default credentials

Probability
Just like Android, an app at iOS is not capable of opening a TCP port to allow incoming connections. As the smartphone is online and not secured by a firewall, alternate methods can be used to establish a connection. Due to the Controlled app distribution these alternate methods could be detected by Apple as the app is reviewed. The probability of this attack is: **Impossible**

When a device is jailbroken, the security mechanisms could be bypassed. When a devices is jailbroken it is possible that a SSH server is installed at the device with a default password. The probability of this attack is: **High**

A12. App is pirated version of paid app

Security mechanisms	*Vulnerabilities*
-	V3. Relation between developer and apps is not clear

Probability
The report from Distimo [56] shows that over 73% of the apps are paid at the App Store. When the same app can be downloaded for free we may assume that the victim will download the app. Several paid apps also offers a lite version. When the victim is looking for this lite version it is possible he downloads the wrong app due to the vulnerability V3 . The probability of this attack is: **High**

A13. Victim does not understand permissions

Security mechanisms	*Vulnerabilities*
S19. Permission model	-

Probability
Compared with the permission model of Android the permission model of iOS is really small. Only two simple permissions exists which the victim probably will understand. The probability of this attack is: **Low**

A14. Attacker can offer malicious app to official Market/store

Security mechanisms	*Vulnerabilities*
S8. App Sandbox	-
S19. Permission model	
S21. Controlled app distribution	

Probability
The permission model and app sandbox does not provide access to the SMS messages. As explained in S21, apps that are uploaded to the App Store are reviewed before they are allowed. As the malicious app will be rejected the probability of this attack is: **Impossible**

A15. Smartphone accepts app as attachment in mail

Security mechanisms	*Vulnerabilities*
S26. App installation restrictions	-

Probability
The security mechanism S26 restrict the device that app can only installed from the official App Store. The probability of this attack is: **Impossible**

When the device is jailbroken this security mechanism could be turned off. The probability of this attack with a jailbroken device is: **Medium**

A16. App can capture screen content
When an app can capture the screen content this could be used to capture the screen content when a SMS message is received. This screen could be forwarded to the attacker.

Security mechanisms	*Vulnerabilities*
S8. App Sandbox	-
S19. Permission model	

Probability
It is impossible to capture the screen content as no permission exists in the permission model. The probability of this attack is: **Impossible**

When the device is jailbroken screen capture would be possible as the permission model could be bypassed. Because a screen capture uses much data it is not likely that an attacker will use this approach. The probability of this attack jailbroken is: **Low**

A17. App can read SMS message from the device

Security mechanisms	*Vulnerabilities*
S19. Permission model	V25. iOS stores SMS messages unencrypted
	V23. SMS messages could be read from the Sandbox

Probability
The permission model does not provide a permission to read a SMS message but vulnerability V23 shows access to these SMS messages is possible without permission model. The probability of this attack is: **High**

A18. App can be decompiled, infected and be recompiled

Security mechanisms	*Vulnerabilities*
S22. Irreversible code	-

Probability
The security mechanism S22 shows that is impossible to decompile the code back to objective C and add malicious code to it. Recompilation from decompiled code is not possible because the decompiled code is not complete. The probability of this attack is: **Impossible**

A19. App can be binary wise infected
Besides infecting an app by adding code to the decompiled source another method is infecting the binary of the application.

Security mechanisms	*Vulnerabilities*
S22. Irreversible code	V29. iOS assembly can be structured viewed

Probability
The vulnerability V29 can be used to find a place where the app can be infected with the malicious code. A technical skilled attacker would be needed to be able to read and modify this assembly. As most attacks are not that heavy technical skilled this attack has the following outcome: **Low**

A20. New popular app is developed
The goal of this attack is that the attacker develops a new app from scratch. This new app has to be interesting to the users which would make it popular.

Probability
Developing an app which will gain a high rate number of downloads does not only consume much time, but also requires a unique idea. The App store offered 500.000 in October 2011 [56]. This makes it difficult to create a popular app due to the high competition. The probability of this attack is:: **Low**

A21. Victim can download malicious app from 3rd party location

Security mechanisms	*Vulnerabilities*
S26. App installation restrictions	-

The app installation restrictions prevents an app to be installed from another location. The probability of this attack is: **Impossible**

When the device is jailbroken, the alternative repository Cydia is installed. In this repository homebrew apps can be found. Besides Cydia, Installous is a popular repository which is an almost copy of the App store only with pirated apps. The probability of this attack is: **High**

A22. App can be hidden
When an app is hidden, this app can be abused to hide a malicious app at the device of the victim.

Probability
The model of iOS makes it mandatory that an app is visible in the launch menu. This makes it impossible to develop a hidden app. The probability of this attack is: **Impossible**

When the device is jailbroken the app could be hidden which makes the outcome: **High**.

A23. App can send SMS without noticing the victim

Security mechanisms	*Vulnerabilities*
S8. App Sandbox	
S9. Inter app communication	
S11. Permission model	

Probability
When an app wants to send a SMS message this has to be done by using the Permission model. The permission model does not provide a way to send an SMS from an app but the Inter-app communication offers an alternative method. Using the Inter-app communication an app can create a SMS message

but not send it yet (which has to be done by the victim). As the victim will notice this message the probability of this attack is: **Impossible**

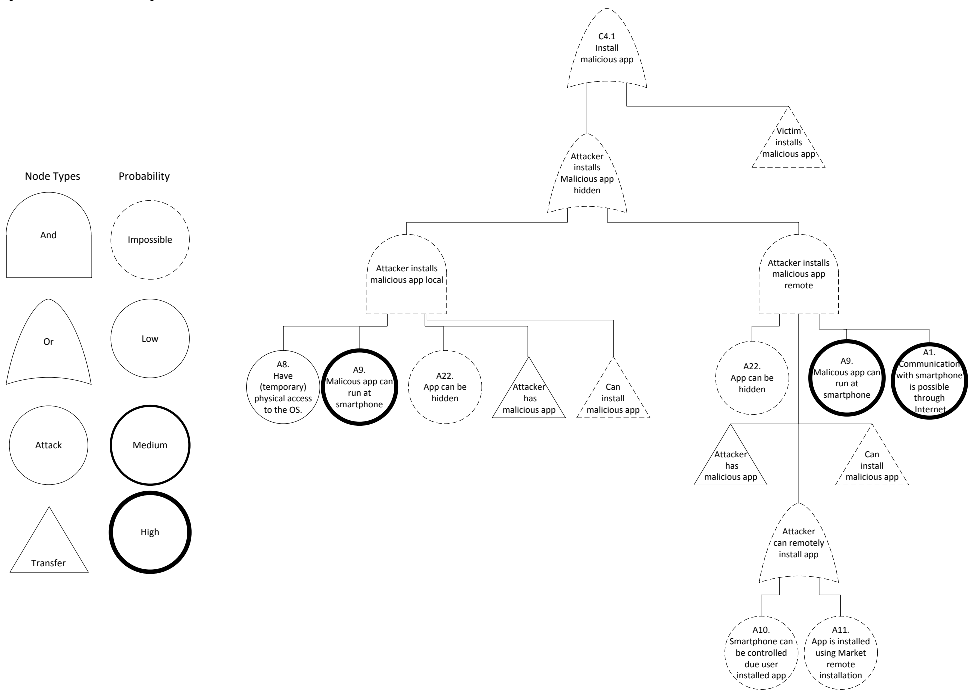
A24. App does not need permissions
This attack is about the situation that a app does not need a permission from the OS to access the needed resources.

Security mechanisms	*Vulnerabilities*
S8. App Sandbox	V34. Capability leak
S19. Permission model	

Probability
As explained in the permission model the iOS system only has 2 permission which a user has to grant. As reading SMS messages and access the internet is not a permission which have to be asked the outcome of this attack is: **High**

7.1.5 Conclusion

In section 7.1.4 we have researched the iOS platform for the probability of the attack nodes from the tree: "Install malicious app". In this chapter we will use this data in conjunction with the attack tree from appendix A. The outcome of this chapter gives an overview of the possible attack vectors. Besides the possibility the probability is added which gives an idea which attack vector will probably be used by the attacker.



Node Types
Probability
And
Impossible
Or
Low
Attack
Medium
Transfer
High
C4.1 Install malicious app
Victim installs malicious app
Attacker installs Malicious app hidden
Attacker installs malicious app local
Attacker installs malicious app remote
A8. Have (temporary) physical access to the OS.
A9. Malicous app can run at smartphone
A22. App can be hidden
Attacker has malicious app
Can install malicious app
A22. App can be hidden
A9. Malicous app can run at smartphone
A1. Communication with smartphone is possible through Internet
Attacker has malicious app
Can install malicious app
Attacker can remotely install app
A10. Smartphone can be controlled due user installed app
A11. App is installed using Market remote installation

Figure 15

Figure 15 shows the goal of the attack vector could not be completed as the possibility of the direct children are all impossible. In appendix A the complete outcome of the attack vector can be found. The main security mechanism which prevents the goal to be achieved is the Controlled app distribution (S20) which regulates that an app is checked for malicious code before it is uploaded to the App Store. When this security mechanism is bypassed it is possible to achieve the goal: Install malicious app.

As described in vulnerability V26 almost 10% of the iPhones are jailbroken. When the device is Jailbroken the goal can be reached. Due to this large amount of jailbroken devices we will focus on a jailbroken iOS device in section 7.2.

7.2 Identify devices

In section 7.1 we have researched if and how likely it was to install an malicious app to an smartphone to read the mTAN. As showed in chapter 5 a successful two factor authentication attack also needs a compromised computer. Reports showed that a lot of computers are infected with malware [9]. We think the probability an attacker can compromise a computer is high. When an attacker has a compromised computer and/or smartphone he has to identify the other device from the other. We named this attack “identify devices”.
We have divided this identification into two different categories which can be used for different purposes:

Targeted attack
With a targeted attack we mean if there is identification information available which can be used to target the attack at the device of the victim. This information can be used to communicate with the victim or his device.

Identification attack
The purpose of an identification attack is used to find two devices with the same user. This type can be used if an attacker has the control of a high number of compromised computers & smartphones.

For targeted and identification attacks, the following information can be used:

Device	*Information*	*Attack*
Smartphone & computer	E-mail address	-Targeted -Identification
Smartphone	IP address smartphone	-Identification
Computer	IP address computer	-Targeted -Identification
Smartphone	Mobile phone number	-Targeted -Identification
Smartphone	Serial number	-Identification

With device we mean the information which can be found ***about*** *the device, not* ***at*** *the device*

In this research we assume that a victim will use the same e-mail address at their smartphone and their computer.
To research the possibility and probability of the Attack Trees: “*Identify computer using smartphone*” and ”*Identify smartphone using computer*” we have extended the Attack Trees which were selected in chapter 5.The outcome of these Attack Trees may be found in appendix A.
Just like in section 7.1 we will research these Attack Trees in conjunction with the security mechanisms and vulnerabilities from chapter 6.
Because of the outcome from section 7.1 we make the assumption that the smartphone is one of the following devices:

- An Android device running a malicious App which can read mTAN
- A jailbroken iOS device malicious App which can read mTAN.

In section 7.2.2 we will research the probability of the tree *"Identify computer using smartphone"*. In section 7.2.3 we perform the same risk analysis but from the perspective from the other device of the tree *"Identify smartphone using computer"*. We will not research each goal separately for Android and iOS as a lot of results would be the same.

7.2.1 Shared attack nodes

In this chapter we will research the attack nodes which are used in both Attack Trees. These attacks can be used in both situations of the attack (*"Identify computer using smartphone"* and *"Identify smartphone using computer"*).

A30. Victim connects smartphone to computer
With this attack we mean a way to communicate to the victim that the victim has to connect the smartphone to the computer.

Probability
With iOS, Android and a computer it is possible to show a message when the malicious app is running (in the background). This message can be used to communicate with the user that the has to connect the device to the computer. Most iOS users will perform this action as it is quite common to connect your iOS device to the computer to activate the device or transfer new music. Android users are not used to connecting the device to their computer as it is capable of managing the device all by itself, but depending on the communication it is likely the device will be connected to the computer. The probability of this attack is: **Medium**

A32. Victim installs malware at other device
When malware is installed at the other device, this malware can send identification information about the device its installed at.

Probability
In attack A30 we already researched if it is possible to ask the victim to perform a task. When the user is asked to install malware at the other device to identify the device we can use the outcome from section 7.1. This section shows that the attack where the victim installs the malware is possible and very likely for Android and jailbroken iOS devices. The probability of this attack depends whether the victim will trust the message. We think a victim is suspicious installing software to one of their devices when it is asked by the attacker. This makes the probability of this attack: **Low**

A34. Victim opens webpage at other device
In attack A30 we researched if it is possible to ask the victim to perform a task. This task could be opening a webpage at the other device. This URL has to be unique to recognize the compromised device.

Probability
It is not realistic that a victim will retype the URL in the browser of the other

device. It will be far more logic to a victim to open the URL at the device it is being showed from. To make the attack more helpful the URL should also contain identification information. As this will enlarge the URL it will be too hard to retype it thus the probability of this attack is: **Low**

A35. Identify using web request

When a web page is opened at a device, the device makes a web request to the web server. This web request can be used in this attack to retrieve identification information from the device which is requesting the webpage.

Probability

When the webpage is requested, the browser information is showed and the IP address of the device requesting the page. This could be a proxy or the device itself. With the browser information we can determine if the page was being requested from a computer and the IP address can be used for identification. Because the web request only shows the IP address of the needed identification information the probability of this attack is: **Medium**

7.2.2 Identify computer using smartphone

These attack nodes assumes the smartphone is compromised and a malicious app is running at the smartphone. Due to the results of section 7.1 this malicious app will run at an Android or a jailbroken iOS device.

A31. Identify computer using USB connection

When the device is connected to the computer, this connection can be used to retrieve information about the connected device.

Probability

When the smartphone is connected to the computer it operates in USB slave mode. In this mode it is difficult/impossible for the smartphone to retrieve identification information about the computer. When the iOS device would be jailbroken it would become possible to act as a host device to communicate with the computer. As no useful information, without a service running at the computer, can be found using this connection probability is that low we can define it as: **Impossible**

A33. Identify computer using malware

This attacks describes if and what identification information can be retrieved by malware.

Probability

When malware is running at the computer this malware can retrieve the needed information from the computer. To link the computer with the smartphone this could be done by entering some identification information when installing the malware. As an Android app is capable of writing a malicious application to the removable storage part of the device, it is possible to add some identification of the smartphone to the installer of the malware.

This way the probability of identifying a computer using malware from a compromised smartphone is: **High**

A36. Find identification information at device
In this attack the identification information of the computer has to be found at the smartphone.

Security mechanisms	*Vulnerabilities*
S8. App sandbox	-
S17. Permission model (Android)	

Probability
Due to the App sandbox, an app cannot access information outside his sandbox. Android can access the profile data through the permission model. It is likely this profile will also contain the installed e-mail accounts at the device. With a jailbroken iOS device it is possible to access the storage where the e-mail address is stored. It is likely no other information about the computer can be found at a smartphone. This makes the outcome of this attack: **Low**

7.2.3 Identify smartphone using computer

These attacks nodes assumes that the computer is compromised and malware runs at the computer. The malware add the computer has full administrator rights. In this attack we will try to identify the smartphone using the computer.

A40. Identify smartphone using USB connection
When the device is connected to the computer, this connection can be used to retrieve information about the connected device.

Probability
When the device is connected to the computer it present itself as a portable device. This portable device has an attribute named serial number which can be read by any application on a Windows device (and likely the same for Linux and MacOS). The devices also register itself as a removable device which represent a memory folder from the device. Depending on the device it is possible there is some identification information available on this memory device. This makes the outcome of this attack: **High**

A42 Identify smartphone using malicious app
This attacks describes if and what identification information can be retrieved by malware.

Probability
In the risk analysis from section 7.1 we have showed that it is possible to install a malicious app to some devices. If we want to identify the smartphone, the victim must enter unique code to identify the devices. When the device is installed from the computer it is possible to add this unique information to the installer of the malware. The outcome of this attack is: **Medium**

A45 Find identification information at device
In this attack we will try to find identification information at the computer of the victim, about the smartphone.

Security mechanisms	*Vulnerabilities*
S27. Backup encryption	V40. Backup is unencrypted
	V41. Serial number is visible

When an iOS device is connected to the computer, iTunes prompts the user to store a backup at the computer. This backup is unencrypted by default and can be read by malware. As the option to encrypt the backup is not easy to find, it is not likely that the backup will be encrypted. In this backup file the following identification information can be found:
- Apple ID
- Installed e-mail accounts
- IP addresses of last connected Wi-Fi/3G networks

Instead of the computer Android relies on the cloud for backup and file storage. There is no standard software for every Android device which can be used to synchronize the device with the computer. Because of this we think the computer does not store much information about the device. As the research from section 7.1 showed a Gmail account is used a lot at Android devices this Gmail account probably will also be used at the computer. This way the Gmail account can be captured and be used as identification information.

The probability of this attack is: **High**

7.2.4 Conclusion

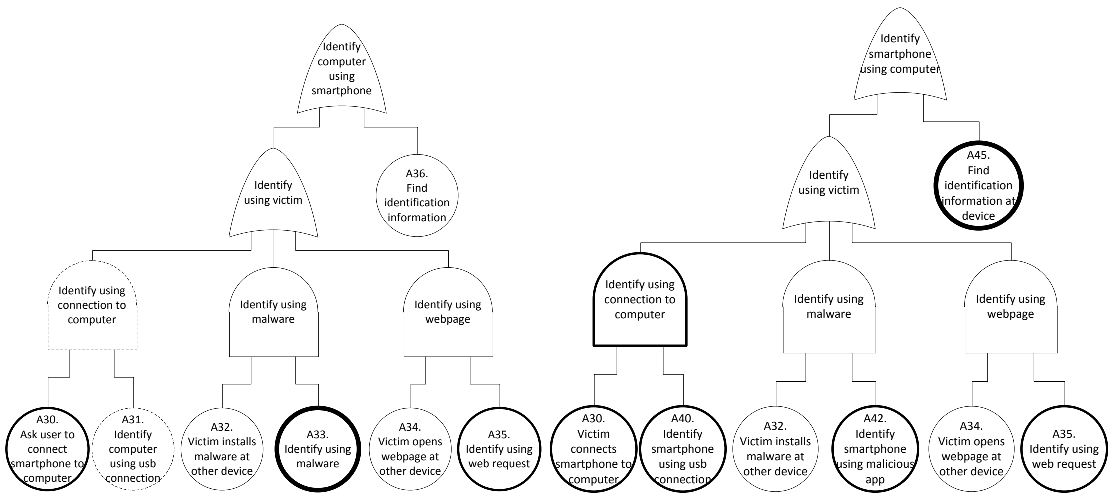

Identify computer using smartphone
Identify using victim
A36. Find identification information
Identify using connection to computer
Identify using malware
Identify using webpage
A30. Ask user to connect smartphone to computer
A31. Identify computer using usb connection
A32. Victim installs malware at other device
A33. Identify using malware
A34. Victim opens webpage at other device
A35. Identify using web request
Identify smartphone using computer
Identify using victim
A45. Find identification information at device
Identify using connection to computer
Identify using malware
Identify using webpage
A30. Victim connects smartphone to computer
A40. Identify smartphone using usb connection
A32. Victim installs malware at other device
A42. Identify smartphone using malicious app
A34. Victim opens webpage at other device
A35. Identify using web request

Figure 16

Figure 16 shows that it is possible to identify the other device from a computer and a smartphone. The most interesting attack would be the one where the victim is not needed.
The probability this attacked is performed using the smartphone is low. The reason is that a smartphone does not store much information about the connected computer and the security mechanisms prevents access to this information.
The probability of the attack is higher when it is being performed from a computer. Because an iOS device stores backup information at the computer, which contains identification information, this approach is more likely. For Android the Gmail account can be used as identification information which is probably used at both devices. This way it would become possible to perform a targeted and identification attack.

8 Counter measures

The risk analysis from chapter 7 revealed that there are several weak points which could lead to a successful attack to the service which the victim is using. In this chapter we will define some counter measures which the involved parties could do to make the system more safe. This information can be used by the parties and is an input for the future work. The following parties will be treated:
- 8.1 Customers
These are the customers of a service. For example this would be the customers of a bank in the ING case.

- 8.2 Smartphone OS manufactures
With smartphone manufactures we mean Android and iOS. Not the companies who build the smartphone and install the OS like Samsung.

- 8.3 Service provider
A service provider is a provider which uses the mTAN. An example of a service is ING.

These counter measures are defined by analyzing the risk analysis and brainstorming sessions. Literature research is used to find counter measures for some problems but the brainstorming sessions also generated some new counter measures. Because these counter measures must be researched before they could be used the subjects are also introduced in the future work section.

This chapter give answer to the sub research question:

> *"Which counter measures could be introduced to make the system more secure?"*

8.1 Customers

In this section we will describe the counter measures which customers from a service provider could use to make the system more secure.

8.1.1 iOS: do not jailbreak/root device

When the device is jailbroken this gives the customer the ability to install apps for free but some security mechanisms are also disabled. As some security mechanisms are disabled which also lower the security of mTAN it is advised to not jailbreak/root your device.

8.1.2 Minimize identifying information from computer

Making a device secure is difficult for the average user. The greatest strength of the two factor security mechanism is that the attacker has to identify both devices of the victim. By minimizing the information about your computer stored at your smartphone (and vice versa) this will increase the security.

8.1.3 Android: do not install apps from 3rd party Market

Downloading apps from 3rd party Markets/stores is mostly done to obtain an app for free. When a customer really wants this it is advised to try download the pirated app from the official Market. Due to the remote removal security mechanisms Google could remove the malware from your device after it has been infected. With an malicious app installed from an 3rd party Market this would not be possible.

8.1.4 Android: secure your Gmail account

The risk analysis showed a Gmail account is almost always present at an Android device. As this Gmail account is likely also used at an computer it could be used for identification of the device. When the credentials would be hacked/captured from the Gmail account these could be abused in combination with the remote installation feature. By securing your password (and your computer) from your Gmail account this would decrease the risk.

8.2 Smartphone OS manufactures

In this section we will describe the counter measures which smartphone OS manufactures could use to make the system more secure.

8.2.1 Android: redesign permission model

The current permission model is, due its size, hard to understand for an average user. Another downside of this permission model is that a user can only agree with all the requested permissions instead of agreeing particular permissions. Researchers developed a new permission model [60] which allows an user to agree to certain permission instead of accepting all the needed permissions. This will not solve all problems of the current permission model.

8.2.2 Android: detect suspicious combination of permissions

The current permission model uses categories to warn the usage of certain permissions. For example the "read SMS" permission is stored under a category "*Your messages*" and the "send SMS" permission is stored under a category "*Services which costs you money*". These categories are meant to warn the user about a permission. As learned from the risk analysis, the system warns the user when certain permissions are needed. This counter measures is about raising a warning to the victim when certain combinations of permissions are used.
For example we take an app which wants to have the following permissions:

- Internet
- Background processing
- Read SMS.

The combinations of these permissions is suspicious and an extra warning could be showed to the user.

8.2.3 Android: detect suspicious combination of permission and category

Each Android app is divided into a category in the Market. An app in all the categories is capable of requesting all kind of permissions. As it is not likely an app from the category "games" would need a permission to read the SMS inbox an system could be developed which detects suspicious combinations of permission within an category.

8.2.4 Android: automatic review of apps

Android differs from iOS as they are less strict which apps are offered to their users. Without giving up this philosophy completely it would help to review the apps as they are submitted to the store. Several studies developed mechanisms to detect malicious apps which leaks data, TantDroid [61], and mechanisms to detect an cloned and infected app [42]. The result of these mechanisms could be used to add warnings to the app or block the app from the Market.

8.2.5 iOS: detect jailbroken iOS devices

As explained in this research, jailbroken iOS devices can have disabled security mechanisms. Because jailbroken iPhones are not safe anymore to be used with two factor authentication, an mechanisms could be developed which detects jailbroken devices. When Apple could detect these devices and make this data available to services this could increase the security as jailbroken iPhones could be treated in a different way.

8.2.6 iOS: remove identification data from backup

The backup from an iOS device contains a lot of identification data which could be used for an identification or targeted attack. As most of this information is not necessary needed in a backup (for example: the last used IP addresses from an device) this information should be removed to make it more difficult to identify the smartphone.

8.2.7 iOS: enhance security model at device

The risk analysis showed that the most vital security mechanism for iOS is the controlled app distribution. When a malicious app would be allowed to the store by a mistake/bug this app is not heavily secured at the device. Because of the limited design of the security model the app would be capable to access information which is not allowed by Apple. An example is the capability to read the SMS messages. When Apple does not want an app to perform this behavior, it also should be blocked by the sandbox to access this data.

8.2.8 Add antivirus permissions

As explained in chapter 6, antivirus software exists for the Android and iOS platform. This software is not very helpful as there are no special permissions for the antivirus app. Because of this it is difficult for an antivirus app to detect and prevent malware. The OS could add some special permissions to the device which helps an antivirus app to fully operate. As these permissions could also be abused it, is important that these apps need special attention/review from the OS before they can be installed to the device.

8.2.9 Secure identification information

Currently Android and iOS does not focus a lot on securing the identification information. For example the e-mail address could be retrieved from the system to make an identification between the computer and the smartphone. By securing this information, the probability of this attack researched in this paper would be lower.

8.3 Service providers

In this section we will describe the counter measures which service providers could use to make the system more secure.

8.3.1 Flash SMS

Flash SMS, also called Class0 SMS, is a technique used to send a SMS message to the users which is immediately displayed at the screen of the user and not stored in the inbox. A side effect of this technique is that the permission from Android, “Read SMS” is not capable of intercepting these kind of messages. It is unclear how many devices/operators support the flash SMS technique as well if there are alternative ways to access these messages from an Android device.

8.3.2 Use dynamic format of messages

As showed in the case of the ING the format of a SMS message where a mTAN is encapsulated in is always the same. This helps an attacker to detect these messages and extract the mTAN. When multiple (unpredictable) mTAN formats would be used this would make the detection and extraction of the mTAN more difficult to the attacker.

8.3.3 Use app for receiving mTAN at smartphones

What we have learned from the risk analysis is that the architecture strongly focus on securing the data inside an app. Functionality from the device, like the camera or SMS messages, are seen as resources which can be used by an app. Because the mTAN is stored inside such a service, a SMS messages, it could increase the security by moving the mTAN to an app. Resources inside an app could not that easily be read by another app. If this counter measure is effective and how it should be implemented is something which could be researched in the future research section.

8.3.4 Add metadata to mTAN

The research showed that reading the mTAN is possible in certain situations. Tampering the messages which contains the mTAN is something which is not always possible. When the service provider would add extra metadata to the message, this could help the user to detect a malicious transaction.

9 Future research

In this chapter the subjects for future research are described. These subjects mostly are related to the counter measures chapter. When the counter measures would be researched this information could be used in combination with this thesis to perform the risk analysis again.
For each subject we reference to the chapter where it was discovered to get more information about the subject.

- Redesign permission model

  See section: 8.2.1

- Android: detect suspicious combination of permissions

  See section: 8.2.2

- Android : detect suspicious combination of permission and category

  See section: 8.2.3

- iOS: detect jailbroken iPhones

  See section: 8.2.5

- Add antivirus permissions

  See section: 8.2.8

- Flash SMS

  See section: 8.3.1

- Use app for receiving mTAN at smartphones

  See section: 8.3.3

- Capability leak of access to SMS messages

  See section 6.4.2 vulnerability V34

10 Conclusion

We have started this research with the research question: *"To what extent is mTAN still a secure method to use with a smartphone?"* Because social engineering is an attack which could always be used and systems always can have bugs we have decided to ignore these two attack from the research. Initially, the research focused on how the mTAN could be read by the attacker. Chapter 3 showed that mTAN is mostly used in a two factor authentication situation which means another authentication method is needed to access the service. This other authentication method is mostly a username & password which has to be entered on another device as shown in figure 17:

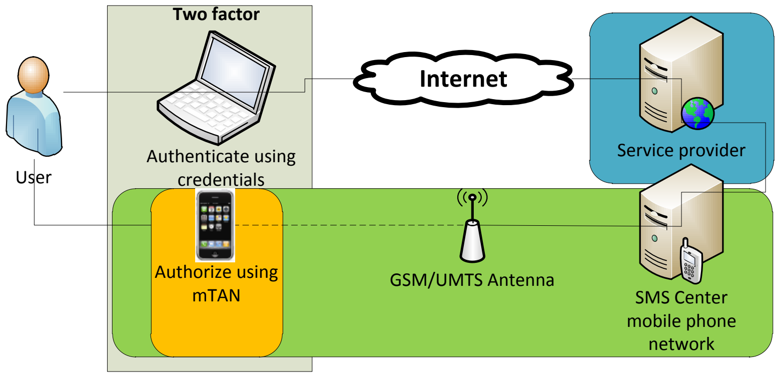


Figure 17

This outcome changed the focus on not only capturing the mTAN but also identifying the computer of the victim where the credentials are used (and vice versa). Identification is enough as we assume in the research the computer can be infected with malware to capture the credentials at the computer of the victim.

The risk analysis revealed that the probability for reading the mTAN using a malicious app is high for the Android platform. It is possible because there are several methods to install a malicious app and it is very for the attacker to develop a malicious app. The controlled app distribution from iOS prevents the malicious app to be offered at the App store as the app is scanned for malicious code by Apple before approved to this store. Exact numbers are not known but we guess about 10% of the iOS are jailbroken which add the capability to install the malicious app to the device of the victim. We can conclude that the Android platform is more secured at the device but the level of security depends on the knowledge of the user because he must approve the permissions.
To detect and remove malware we use antivirus software at the computer platform. For Android and iOS antivirus software does exist but this research

showed it is not very effective. As antivirus software needs almost full access to the system to analyze the behavior of software, it cannot operate from the restrictions from the sandbox. Android is designed to provide access to almost every resource of the system but it isolates apps from each other. The is that an antivirus app at Android is also isolated from other apps and it cannot really detect malicious behavior from other apps. At iOS only one antivirus app exists and due to the sandbox it is only capable of scanning documents and not the installation file of an app.

The risk analysis of the second goal in section 7.2 (identification of the devices), showed that the highest risk the other device from the victim could be identified is by looking for information at his computer. Computers which are coupled with iOS devices stores identification information which can be used for this attack. This information can be found in the backup folder of iTunes which stores mostly unencrypted backups of the (previous) connected iOS devices.
Android devices are not that likely to be coupled with a computer but the Gmail account of the victim could be used for identification. As the probability that the Gmail account is being used at the computer and the smartphone is high, this can be used for identification. When the credentials from this Gmail account are obtained they could even be abused in combination with the remote installation feature from Android, to install an malicious app to the device for identification (and infection).

Using the results of this research we can conclude the rise of smartphones changed the threat model and we can conclude that mTAN is not a secure security mechanism when it is used with a smartphone.

11 Reflection

In this chapter I will reflect on the approach, methodology and outcome of the research. The content of this chapter can be used in further research to prevent the mistakes I have made.

11.1 Approach

When I started the research I have created a research proposal. This proposal was well defined as the chosen subjects from this proposal are pretty much alike the subjects in this paper. The main problem was that I had too much work scheduled for the given time. To solve this problem I've scrapped some parts of the research and moved them to the future research chapter to avoid delay.

When I started the research I focused too much on the first phase where all the background information was explained and defined. This phase became too big and irrelevant and I had to remove a lot of written information from these chapters. The reason why I had this problem was because I didn't define the audience who would read my paper. The effect of this was that I wrote my paper in a way everyone could read and understand it. Due to the subject this is impossible as general information about information technology is needed to understand the paper.

To prevent this mistake in the rest of the paper I've decided to not fully complete the rest of my chapters but first write parts of information I was going to describe in the chapters. That created a clear view of the paper and allowed changes which were less time consuming.

11.2 Attack Trees

To perform the risk analysis I had chosen to make use of Attack Trees. This method helped me in the brainstorming sessions I've had with other experts.. Presenting the outcome of the Attack Trees however was quite more difficult. As attack nodes can be used more than once in an attack vector it was not possible to describe the outcome of the Attack Tree very easily. I have solved this by describing the attack nodes individually and only discuss the final outcome of the Attack Tree as shown in chapter 7. This has the downside that the relation of nodes, which determine the outcome of an attack, cannot be that clearly discussed. When all the Attack Trees had to be discussed this would become too large for the chapter. We have solved this problem by simplifying the Attack Trees and group attack vectors as a new related Attack Tree. This way we could present and discuss the outcome of the trees in the conclusion section from the risk analysis. If a reader wants to view the relation between these nodes I've added the Attack Trees to the Appendix

11.3 Background information

Due to the subject it was hard to find relevant information which I could use to use in the research. The security model from the operating systems is not public available by the companies. We had to use some sources which were not scientific proven. Those sources were from hackers/security engineers who reverse engineered the security model of the OS or websites which are stating some relevant facts. I've solved that problem by not making any concrete conclusions where these sources where used.

11.4 Most interesting outcome

When I started the research my main goal was to research if the mTAN could be read by an attacker from the mobile device. During the research I have found out that the use of two channels (computer and mobile phones) was an important security mechanism. When a computer and a smartphone can easily be compromised this does not mean the attacker could achieve its goal because he has to figure out which computer and mobile phone belong to the same user. That's why I had added section 7.2 to research in what way these devices could be identified.

12 Bibliography

[1] (2012, Feb.) Nederlandse Verereniging van Banken - Nieuwe campagne in strijd tegen toenemende fraude. [Online]. http://www.nvb.nl/index.php?p=918684

[2] (2012, Feb.) Webwereld - Fraude bij internetbankieren ruim verdubbeld. [Online].

[3] Radboud University Nijmegen, "Risicoanalyse EPD-DigiD," 2010.

[4] Sjouke Mauw en Martijn Oostdijk, "Attack Trees: Door de bomen de bedreigingen zien,".

[5] Bruce Schneier, "Attack Trees," 1999.

[6] Amenaza Technologies Limited, "Creating Secure Systems through Attack Tree Modeling," 2003.

[7] Patrick Schweitzer, "On Attack/Defense Trees," 2012.

[8] (2012, Feb.) DigiD - Wie doen mee. [Online]. http://www.digid.nl/burger/wie-doen-mee/

[9] (2012, Feb.) Symantec security report 2010. [Online]. https://www4.symantec.com/mktginfo/downloads/21182883˙GA˙REPORT˙ISTR˙Main-Report˙04-11˙HI-RES.pdf

[10] (2012, Feb.) McAfee Threats Report: Third Quarter 2011. [Online]. http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q3-2011.pdf

[11] (2012, Feb.) A Large-Scale Study of Web Password Habits. [Online]. http://research.microsoft.com/pubs/74164/www2007.pdf

[12] a Division of Reed Elsevier Inc. Economic Crime Institute of Utica College and LexisNexis, "Identity Fraud: A Critical National and Global Threat," 2003.

[13] (2012, Feb.) Identity Theft Resource Center. [Online]. http://www.idtheftcenter.org/

[14] Duncan de Borde, "Two-factor authentication,".

[15] (2012, Feb.) ING- Record aantal gebruikers Mijn ING. [Online]. http://www.ing.nl/particulier/nieuws-en-kennis/ing-nieuws/2011/05/20110525˙Record˙aantal˙gebruikers˙Mijn˙ING.aspx

[16] (2012, Feb.) Gartner - Mobile OS market share. [Online]. http://www.gartner.com/it/page.jsp?id=1622614

[17] (2012, Feb.) Nokia - Future of Symbian. [Online]. http://wl4.peer360.com/b/OZ4Y0HB4qe2y4P0Aobr0/main.asp

[18] (2012, Feb.) Android developers - Platform versions. [Online]. http://developer.android.com/resources/dashboard/platform-versions.html

[19] (2012, Feb.) iOS versions in the wild. [Online]. http://www.cocoanetics.com/2011/08/ios-versions-in-the-wild/

[20] Richard Clayton, and Ross Anderso Tyler Moore, "The Economics of Online Crime," 2009.

[21] Charles Brookson, "Can you clone a GSM Smart Card (SIM)?," 2002.

[22] (2012, Feb.) Wired - Break GSM with a $15 phone. [Online]. http://www.wired.com/threatlevel/2010/12/breaking-gsm-with-a-15-phone-plus-smarts

[23] Steve Gold, "Cracking GSM," *Network Security*, vol. 2011, no. 4, pp. 12-15, 2011.

[24] Adi Shamir, David Wagner Alex Biryukov, "Real Time Cryptanalysis of A5/1 on a PC," 2001.

[25] Fabian van den Broek, "Eavesdropping on GSM: state-of-affairs," 2010.

[26] Xiaoyan Li Sylvie Laforet, "Consumers' attitudes towards online and mobile banking in China," 2005.

[27] Vebjorn,Moen Thomas Tjostheim Kjell J. Hole, "Online banking security," 2006.

[28] Marc Brineco. (2012, Feb.) GSM A5 Files Published on Cryptome. [Online]. http://cryptome.org/jya/crack-a5.htm

[29] Rita Mahajan Anita Singhrova, "Performance Analysis of 3G Protocol Encryption and Authentication," 2006.

[30] Elad Barkan and Eli Biham, "Conditional Estimators: An Effective Attack on A5/1," *Selected Areas in Cryptography 2005*, pp. 1-19, 2005.

[31] (2012, Feb.) OsmocomBB. [Online]. http://bb.osmocom.org/trac/

[32] (2012, Feb.) Android - What is Android. [Online]. http://developer.android.com/guide/basics/what-is-android.html

[33] Adrienne Porter Felt, Kate Greenwood, David Wagner Erika Chin, "Analyzing Inter-Application Communication in Adndroid," 2011.

[34] (2012, Feb.) Android - Content providers. [Online]. http://developer.android.com/guide/topics/providers/content-providers.html

[35] (2012, Feb.) Apple - iOS - Data management. [Online]. http://developer.apple.com/technologies/ios/data-management.html

[36] Apple. (2012, Feb.) iOS - Introduction to Uniform Type Identifiers Overview. [Online]. http://developer.apple.com/library/ios/#documentation/FileManagement/Conceptual/understanding˙utis/understand˙utis˙intro/understand˙utis˙intro.html

[37] Tweakers.NET. (2012, Feb.) Nederlandse tiener kaapt iPhones via ssh. [Online]. http://tweakers.net/nieuws/63473/nederlandse-tiener-kaapt-iphones-via-ssh.html.

[38] (2012, Feb.) Android - Manifest.permission. [Online]. http://developer.android.com/reference/android/Manifest.permission.html

[39] (2012, Feb.) OpenBinder. [Online]. http://www.angryredplanet.com/~hackbod/openbinder/docs/html/

[40] (2012, Feb.) Android - Distribution agreement. [Online]. http://www.android.com/us/developer-distribution-agreement.html

[41] Android. (2012, Feb.) Android Market Service conditions. [Online]. http://www.google.com/mobile/android/market-tos.html

[42] Andrew Newell, Cristina Nita-Rotaru, Xiangyu Zhang Rahul Potharaju, "Plagiarizing Smartphone Applications: Attack Strategies and Defense Techniques," 2011.

[43] (2012, Feb.) Dex2Jar. [Online]. http://code.google.com/p/dex2jar/

[44] (2012, Feb.) Java decompiler. [Online]. http://java.decompiler.free.fr/

[45] (2012,Feb.) APKTool. [Online]. http://code.google.com/p/android-apktool/

[46] Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith Adam J. Aviv, "Smudge Attacks on Smartphone Touch Screens," 2010.

[47] Android. (2012, Feb.) Web Market. [Online]. https://market.android.com/

[48] Yajin Zhou, Zhi Wang, Xuxian Jiang Michael Grace, "Systematic Detection of Capability Leaks in Stock Android Smartphones," 2012.

[49] Apple. (2012, Feb.) iOS developer guide. [Online]. http://developer.apple.com/devcenter/ios/index.action

[50] (2012, Feb.) Apple - App Store Review Guidelines. [Online]. http://developer.apple.com/appstore/guidelines.html

[51] (2012, Feb.) IDA. [Online]. http://www.hex-rays.com/products/ida/6.2/index.shtml

[52] Apple. (2012, Feb.) iOS: Understanding data protection. [Online]. http://support.apple.com/kb/HT4175

[53] Apple. (2012, Feb.) iOS Developer Enterprise Program. [Online]. http://developer.apple.com/programs/ios/enterprise/

[54] (2012, Feb.) iPhone private frameworks. [Online]. https://github.com/kennytm/iphone-private-frameworks

[55] Michael Gegick Sean Barnum, "Psychological Acceptability," 2005.

[56] Distimo, "Smartphone apps report," 2011.

[57] (2012, Feb.) Classdump. [Online].

http://iphone.freecoder.org/classdump˙en.html

[58] Ipos, MMA Google, ”Smartphone usage and mobile attitudes in Europe,” 2011.

[59] Android. (2012, Feb.) Android - Dalvik bytecode. [Online]. http://source.android.com/tech/dalvik/dalvik-bytecode.html

[60] Sohail Khan, Xinwen Zhang Mohammad Nauman, ”Extending Android Permission Model and Enforcement with User-defined Runtime Constraints,” 2010.

[61] Peter Gilbert, Byung-Gon Chun William Enck, ”TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphone,”.

[62] Dino A. Dai Zovi, ”Apple iOS 4 Security Evaluation,”.

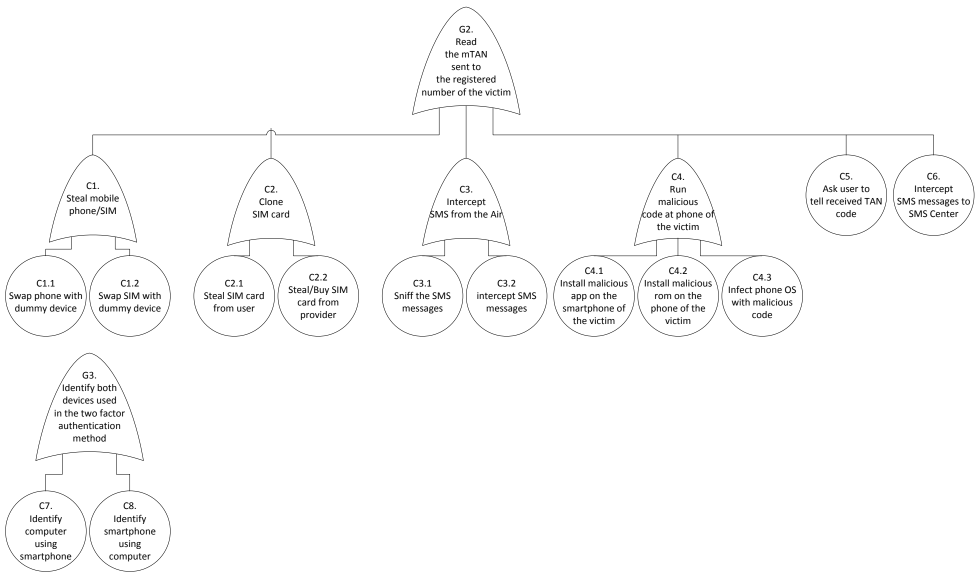
[63] Kunjan Shah, ”Penetration Testing for iPhone / iPad Applications,” 2011.

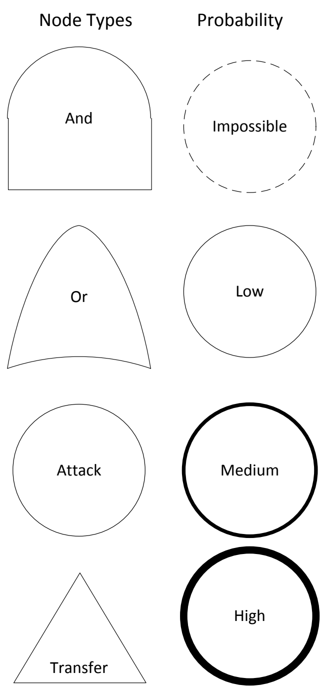
[64] Kurt Root, ”Short Message Service (SMS) Protocols and Architecture ,” 2008.

[65] W.P. van Cuijk, ”Enforcing a fine-grained network policy in Android,” 2011.

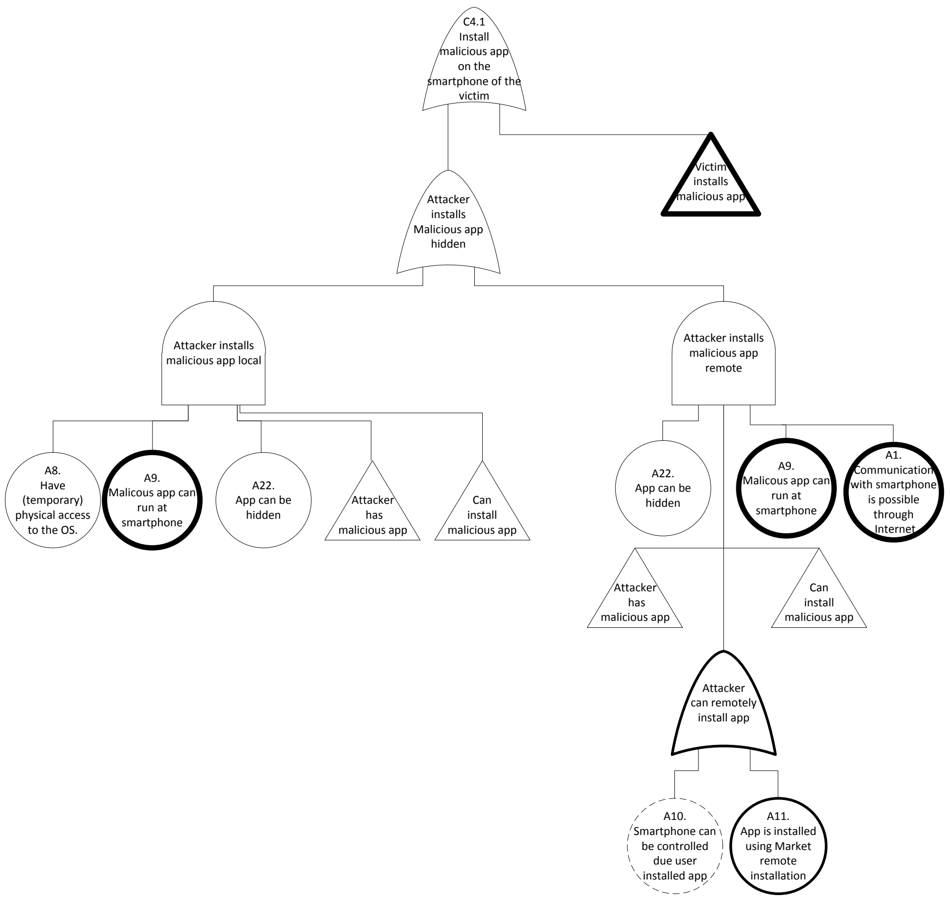
Appendix A: Attack Trees

Goals and legend of the attack

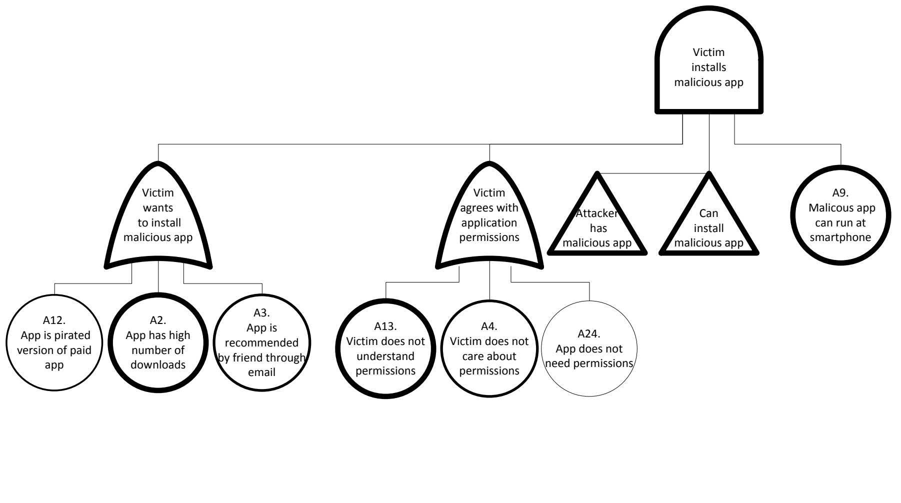

G2. Read the mTAN sent to the registered number of the victim
C1. Steal mobile phone/SIM
C2. Clone SIM card
C3. Intercept SMS from the Air
C4. Run malicious code at phone of the victim
C5. Ask user to tell received TAN code
C6. Intercept SMS messages to SMS Center
C1.1 Swap phone with dummy device
C1.2 Swap SIM with dummy device
C2.1 Steal SIM card from user
C2.2 Steal/Buy SIM card from provider
C3.1 Sniff the SMS messages
C3.2 intercept SMS messages
C4.1 Install malicious app on the smartphone of the victim
C4.2 Install malicious rom on the phone of the victim
C4.3 Infect phone OS with malicious code
G3. Identify both devices used in the two factor authentication method
C7. Identify computer using smartphone
C8. Identify smartphone using computer



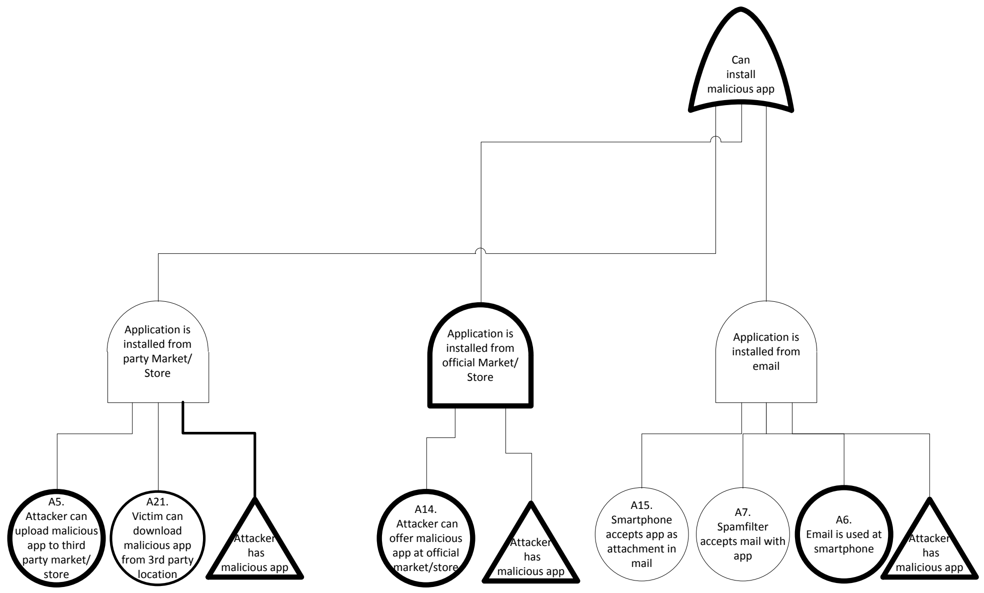
Analysed for: Android
Name: Install malicious app on the smartphone of the victim


C4.1 Install malicious app on the smartphone of the victim
Victim installs malicious app
Attacker installs Malicious app hidden
Attacker installs malicious app local
Attacker installs malicious app remote
A8. Have (temporary) physical access to the OS.
A9. Malicous app can run at smartphone
A22. App can be hidden
Attacker has malicious app
Can install malicious app
A22. App can be hidden
A9. Malicous app can run at smartphone
A1. Communication with smartphone is possible through Internet
Attacker has malicious app
Can install malicious app
Attacker can remotely install app
A10. Smartphone can be controlled due user installed app
A11. App is installed using Market remote installation

Analysed for: Android
Name: Transfer: victim installs malicious app

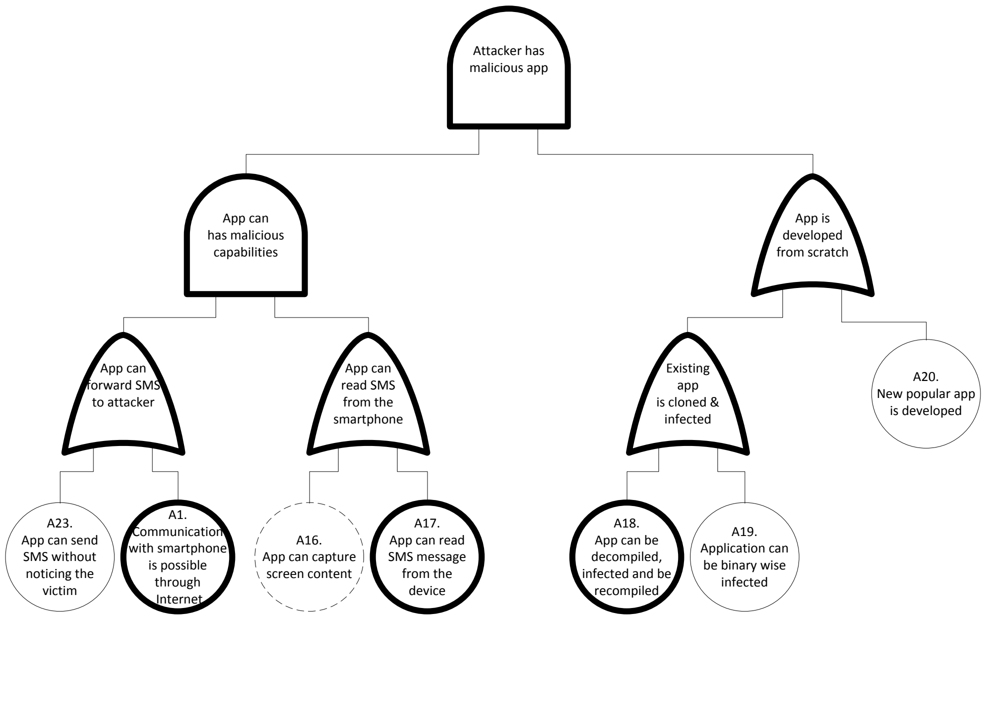

Victim installs malicious app
Victim wants to install malicious app
Victim agrees with application permissions
Attacker has malicious app
Can install malicious app
A9. Malicous app can run at smartphone
A12. App is pirated version of paid app
A2. App has high number of downloads
A3. App is recommended by friend through email
A13. Victim does not understand permissions
A4. Victim does not care about permissions
A24. App does not need permissions

Analysed for: Android
Name: Transfer: can install malicious app



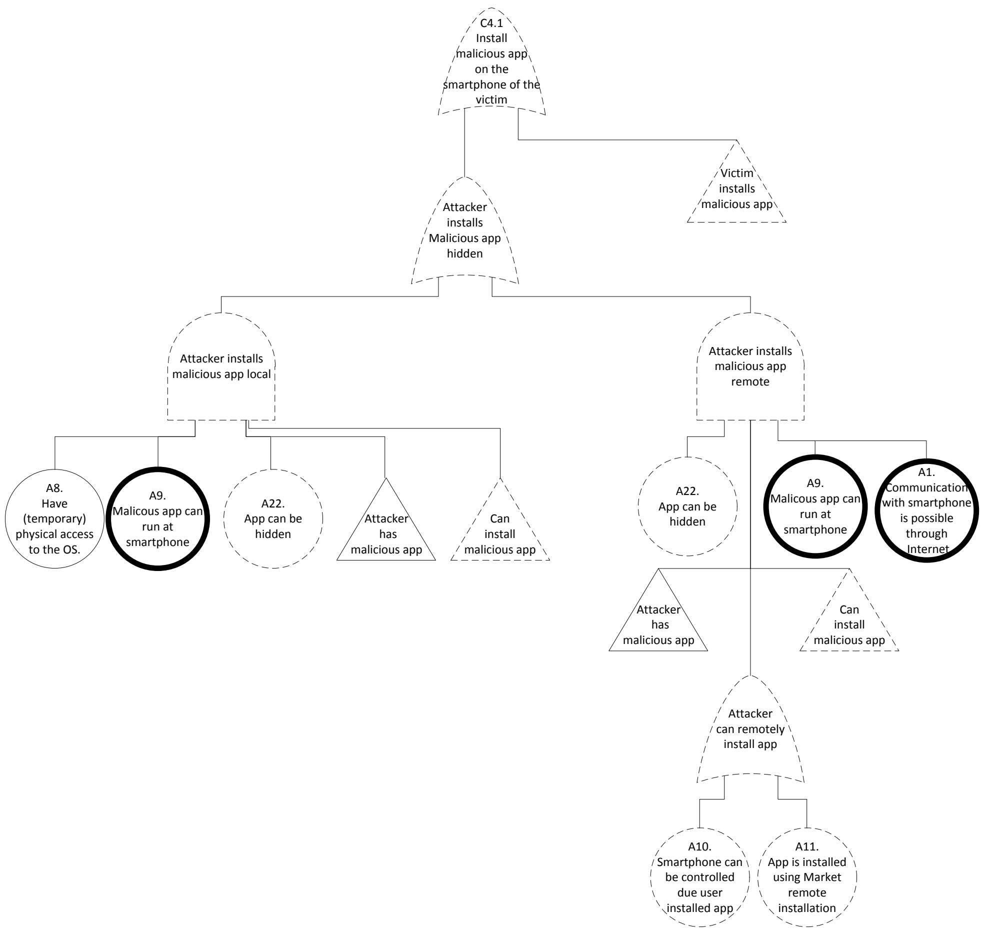
Can install malicious app
Application is installed from party Market/ Store
Application is installed from official Market/ Store
Application is installed from email
A5. Attacker can upload malicious app to third party market/ store
A21. Victim can download malicious app from 3rd party location
Attacker has malicious app
A14. Attacker can offer malicious app at official market/store
Attacker has malicious app
A15. Smartphone accepts app as attachment in mail
A7. Spamfilter accepts mail with app
A6. Email is used at smartphone
Attacker has malicious app

Analysed for: Android
Name: Transfer: attacker has malicious app



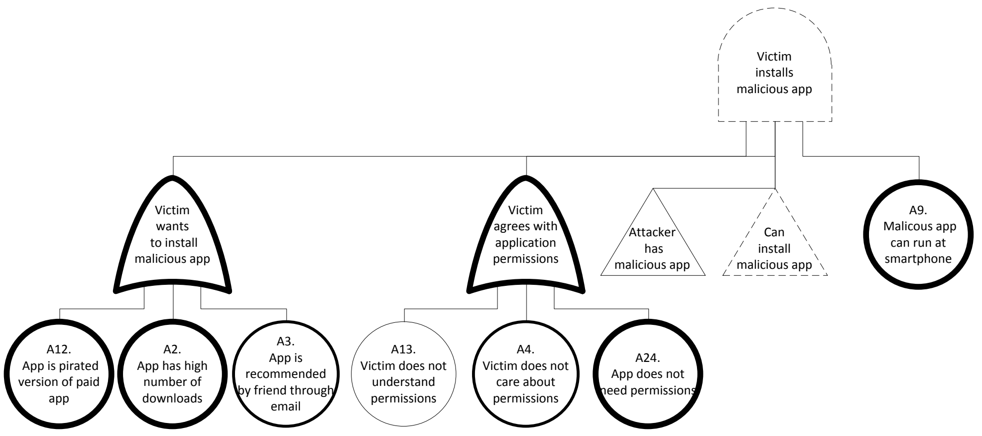
Attacker has malicious app
App can has malicious capabilities
App is developed from scratch
App can forward SMS to attacker
App can read SMS from the smartphone
Existing app is cloned & infected
A20. New popular app is developed
A23. App can send SMS without noticing the victim
A1. Communication with smartphone is possible through Internet
A16. App can capture screen content
A17. App can read SMS message from the device
A18. App can be decompiled, infected and be recompiled
A19. Application can be binary wise infected

Analysed for: iOS
Name: Install malicious app on the smartphone of the victim



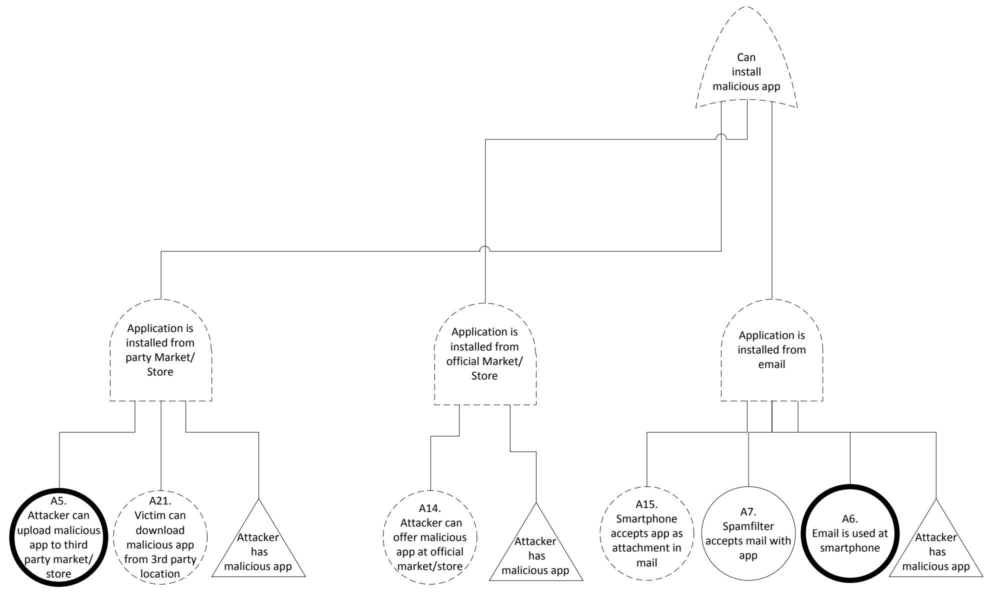
C4.1 Install malicious app on the smartphone of the victim
Attacker installs Malicious app hidden
Victim installs malicious app
Attacker installs malicious app local
Attacker installs malicious app remote
A8. Have (temporary) physical access to the OS.
A9. Malicous app can run at smartphone
A22. App can be hidden
Attacker has malicious app
Can install malicious app
A22. App can be hidden
A9. Malicous app can run at smartphone
A1. Communication with smartphone is possible through Internet
Attacker has malicious app
Can install malicious app
Attacker can remotely install app
A10. Smartphone can be controlled due user installed app
A11. App is installed using Market remote installation

Analysed for: iOS
Name: Transfer: victim installs malicious app

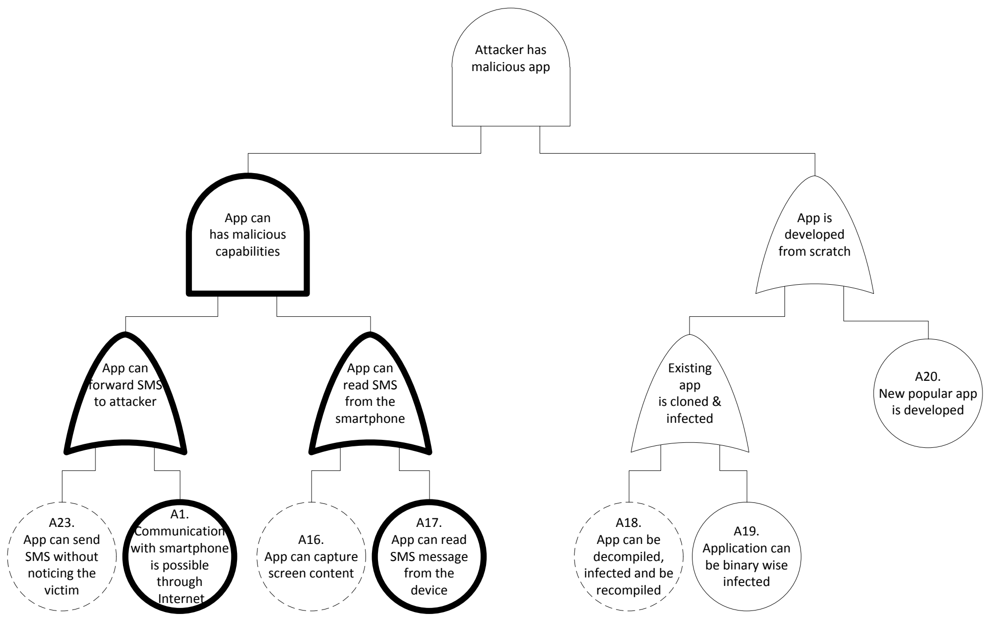


Victim installs malicious app
Victim wants to install malicious app
Victim agrees with application permissions
Attacker has malicious app
Can install malicious app
A9. Malicous app can run at smartphone
A12. App is pirated version of paid app
A2. App has high number of downloads
A3. App is recommended by friend through email
A13. Victim does not understand permissions
A4. Victim does not care about permissions
A24. App does not need permissions

Analysed for: iOS
Name: Transfer: can install malicious app


Can install malicious app
Application is installed from party Market/ Store
Application is installed from official Market/ Store
Application is installed from email
A5. Attacker can upload malicious app to third party market/ store
A21. Victim can download malicious app from 3rd party location
Attacker has malicious app
A14. Attacker can offer malicious app at official market/store
Attacker has malicious app
A15. Smartphone accepts app as attachment in mail
A7. Spamfilter accepts mail with app
A6. Email is used at smartphone
Attacker has malicious app

Analysed for: iOS
Name: Transfer: attacker has malicious app


Attacker has malicious app
App can has malicious capabilities
App is developed from scratch
App can forward SMS to attacker
App can read SMS from the smartphone
Existing app is cloned & infected
A20. New popular app is developed
A23. App can send SMS without noticing the victim
A1. Communication with smartphone is possible through Internet
A16. App can capture screen content
A17. App can read SMS message from the device
A18. App can be decompiled, infected and be recompiled
A19. Application can be binary wise infected