Aalto University
School of Science
Degree Programme of Computer Science and Engineering

Olli Jarva

Intelligent two-factor authentication

Deciding authentication requirements using historical context data

Master's Thesis
Espoo, May 13, 2014

Supervisor: Professor Tuomas Aura
Instructor: Professor N. Asokan

Aalto University
School of Science
Degree Programme of Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Olli Jarva

Title:
Intelligent two-factor authentication – Deciding authentication requirements using historical context data

Date: May 13, 2014 **Pages:** xi + 105

Professorship: Data Communication Software **Code:** T-110

Supervisor: Professor Tuomas Aura

Instructor: Professor N. Asokan

This thesis is a case study of designing and implementing a more secure authentication service for Futurice Ltd., which is a medium-sized software consulting company. Majority of its employees are located in various client offices and need to access internally developed web services over the Internet. In this environment, single password authentication does not provide a sufficient level of security. The standard solution is to require a second authentication factor, such as one-time code in addition to the password.

At Futurice, the old authentication service requested authentication once every 12 hours. These unnecessary interruptions annoy users, thus reducing their satisfaction with the tools. Furthermore, when performing complicated tasks, recovering from small interruptions may take several minutes. Deploying two-factor authentication has not been possible because of the increased negative effects of the interruptions. Therefore, the goal of this thesis is to minimize the number of authentication prompts the user encounters, while increasing the security by enabling two-factor authentication.

The new two-factor authentication service presented in this thesis uses historical activity data to decide when the user should be reauthenticated. With the new system, reauthentication was requested approximately once per two weeks (on average), resulting in 90% reduction in the number of authentication prompts, without compromising security. Two-factor authentication is not required when there is other evidence from the context data that the user is authentic.

A brief inspection of the EU and Finnish laws indicated that the data collection and processing for context based authentication is acceptable. Our results show that the time and effort spent on authentication processes can be reduced with relatively small effort. Similar results should be achievable in other companies and organizations. Thresholds for various algorithms may require tuning, and future work is needed to automate this.

Keywords: two-factor authentication, progressive authentication, risk-based authentication

Language: English

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan tutkinto-ohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Olli Jarva		
Työn nimi:	Älykäs kaksivaiheinen autentikaatio – Autentikaatiovaatimusten valitseminen automaattisesti historian perusteella		
Päiväys:	13.5.2014	**Sivumäärä:**	xi + 105
Professuuri:	Tietoliikenneohjelmistot	**Koodi:**	T-110
Valvoja:	Professori Tuomas Aura		
Ohjaaja:	Professori N. Asokan		

Tässä diplomityössä esitellään Futurice Oy:lle suunniteltu ja toteutettu turvallisempi autentikaatiojärjestelmä. Futurice on keskisuuri ohjelmistokonsultointiyritys, jonka työntekijät käyttävät erilaisia sisäisesti kehitettyjä www-palveluita asiakkaiden tiloista. Tällaisessa ympäristössä pelkästään salasanaan perustuva autentikaatio ei tarjoa riittävän korkeaa turvallisuustasoa. Yleinen ratkaisu tähän on vaatia toinen autentikaatiovaihe, yleensä kertasalasanalla.

Futuricen vanha autentikaatiojärjestelmä vaati salasanan joka 12. tunti. Tämä tarpeeton keskeytys ärsyttää käyttäjiä ja vähentää tyytyväisyyttä käytössä oleviin työkaluihin. Kun käyttäjät suorittavat monimutkaisia tehtäviä, lyhyistäkin keskeytyksistä palautuminen voi kestää useita minuutteja. Kaksivaiheisen autentikaation käyttöönotto ei ole ollut mahdollista, sillä se lisäisi entisestään keskeytysten haittoja. Tämän diplomityön tavoitteena on minimoida käyttäjän kohtaamat autentikaatiohaasteet, vaikka turvallisuutta parannetaan kaksivaiheisella autentikaatiolla.

Tässä työssä esitelty kaksivaiheinen autentikaatiojärjestelmä käyttää aikaisempia aktiviteettitietoja käyttäjän aitouden arviointiin. Uusi järjestelmä vaati uudelleenautentikoinnin keskimäärin noin kerran kahdessa viikossa. Tämä on 90% vähennys autentikaatiohaasteisiin – ilman turvallisuuden vaarantumista. Kaksivaiheista autentikaatiota vaaditaan, kun tiedot kirjautumisyrityksestä eivät tue riittävästi käyttäjän aitoutta. Lyhyt katsaus EU:n ja Suomen lainsäädäntöön osoitti, että tällaiselle tietojen keräykselle ja käsittelylle kontekstiin perustuvassa autentikaatiossa ei ole esteitä.

Tämän diplomityön tulokset osoittavat, että autentikaatioon tarvittavaa aikaa ja vaivaa voi vähentää suhteellisen yksinkertaisilla menetelmillä. Muissa yrityksissä ja organisaatioissa pitäisi olla mahdollista saavuttaa vastaavia tuloksia. Eri algoritmien raja-arvojen ympäristökohtainen optimointi pitäisi olla automatisoitavissa, mutta tämän selvittäminen vaatii jatkotutkimusta.

Asiasanat:	kaksivaiheinen autentikaatio, älykäs autentikaatio, riskianalyysiin perustuva autentikaatio
Kieli:	Englanti

Acknowledgements

I wish to express my gratitude to my supervisor, Professor Tuomas Aura for literally hundreds of insightful comments. Similarly I thank Professor N. Asokan, my instructor, for his valuable time and patience with this thesis.

I want to thank Teemu, Markus, Olli P., Senja and Sourav for their efforts to proofread and comment this text.

This thesis is available at `http://olli.jarva.fi/thesis.pdf`.

Espoo, May 13, 2014

Olli Jarva

Abbreviations and Acronyms

API	Application Programming Interface
CI	Continuous Integration
CSP	Content Security Policy
CSS	Cascading Style Sheet
DNS	Domain Name Service
DTW	Dynamic Time Warping
EER	Equal Error Rate
FFT	Fast Fourier Transformation
FNR	False Negative Rate
FPR	False Positive Rate
GPS	Global Positioning System
HSTS	HTTP Strict Transport Security
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICANN	Internet Corporation For Assigned Names and Numbers
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IO	Input/Output
ISACA	Information Systems Audit and Control Association
ISO	International Organization for Standardization
ISP	Internet Service Provider
LDAP	Lightweight Directory Access Protocol
MitM	Man in the Middle
NTP	Network Time Protocol
OS	Operating System
OTP	One-Time Password
PFS	Perfect Forward Secrecy

PIN	Personal Identification Number
PTY	Pseudo Terminal Device
RIR	Regional Internet Registry
RTT	Round-Trip Time
SAML	Security Assertion Markup Language
SIM	Subscriber Identity Module
SMS	Short Message Service
SSH	Secure Shell
SSO	Single Sign-On
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TOTP	Time-based One-Time Password
UA	User Agent
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VCS	Version Control System
VPN	Virtual Private Network
WGS84	World Geodetic System revision 84
Wi-Fi	Wireless local area network product based on IEEE 802.11 standards
XSS	Cross-Site Scripting

Contents

Abbreviations and Acronyms **v**

1 Introduction **1**

2 Background **5**
2.1 Strong authentication 7
2.1.1 One-time passwords (OTP) 7
2.1.2 Identifying the device 8
2.1.3 Location of the user 9
2.1.4 Biometrics 9
2.2 Single sign-on technologies 10
2.3 HTTP 12
2.3.1 TLS security 13
2.3.2 Cookies 15
2.3.3 Content security policy 16
2.4 IP address databases 16
2.5 Evaluating the security of information systems 17
2.6 Legal aspects 18

3 Operating environment **21**
3.1 Technical environment 22
3.2 Implemented data sources 25
3.3 Potential other data sources 27

4 Implementation **29**
4.1 System architecture 30
4.2 Data collection 35
4.3 Ratelimiting 35
4.4 Second-factor authentication methods 37
4.5 Browser plugins 40

5 Data analysis and evaluation **42**
5.1 Browser authenticity . . . 44
5.1.1 User-agent strings . . . 44
5.1.2 JavaScript fingerprinting . . . 44
5.2 Location . . . 47
5.2.1 Anomaly detection . . . 50
5.2.2 Typical location patterns . . . 51
5.3 Reputation of the IP address . . . 53
5.4 Device fingerprints . . . 55
5.4.1 Clock offsets . . . 55
5.4.2 Device uptime . . . 56
5.4.3 TCP flags . . . 57
5.4.4 Decision process . . . 58
5.5 Working times . . . 58
5.5.1 git commits . . . 60
5.5.2 Modeling and results . . . 62
5.6 Keystroke timing . . . 63
5.6.1 Dynamic time warping . . . 63
5.6.2 Analysis and results . . . 64

6 Results and evaluation **66**
6.1 Security goal . . . 67
6.2 Usability goals . . . 68
6.3 Response times . . . 71

7 Discussion **73**

8 Conclusions **77**

A CSP policy **90**

B Browser versions **91**

C Pre-deployment survey **93**
C.1 Questions . . . 94
C.2 Results . . . 95

D Post-deployment survey **97**

E Geocoding validation dataset **99**

F Validation dataset for employee working times **102**

G Example of WHOIS data **104**

List of Tables

3.1 Details of implemented user activity data sources 25

4.1 Authentication state downgrade mechanisms 30
4.2 Security evaluation in case of a security breach with read-only or read-write access 32
4.3 Ratelimits for different authentication methods 36

5.1 Thresholds for JavaScript fingerprinting 47
5.2 Geocoding errors against the validation dataset 49

6.1 Summary of the accuracy for user activity features 67
6.2 Summary of usability goals 69
6.3 Summary of response times 72

B.1 Visits by server version (2013-09-01 – 2013-09-30) 91
B.2 Visits by server version (2013-04-01 – 2013-09-30) 92

C.1 Survey answers: general questions 95
C.2 Survey answers: users using two-factor authentication 95
C.3 Survey answers: users not using two-factor authentication . . . 96
C.4 Survey answers: Why you don't use two-factor authentication? 96

D.1 Survey questions and answers 98

List of Figures

3.1 Overview of the environment 22

4.1 Authentication state machine 29
4.2 High-level overview of the system components before and after the implementation. All connections are over HTTPS. 31
4.3 Overview of system components. 34

5.1 Authentication decision tree 43
5.2 Browser fingerprint similarities for consecutive sign-in attempts 46
5.3 Histogram of Precision and GeoLite2 errors 49
5.4 Correlation between the accuracy reported by the MaxMind Precision database and the validation dataset. 50
5.5 Travel simulations with different accuracy thresholds. 51
5.6 Demonstration of Markov chains 52
5.7 False negatives on IP reputation with different expiration times 54
5.8 Overview of time offset calculation 56
5.9 Example of irregular uptime records 57
5.10 Working times entries for one user 59
5.11 Examples of recorded working times patterns 60
5.12 Illustration of DTW 64

7.1 Uptime records for two browsers running on the same computer 74

E.1 Reported accuracy of location validation dataset. 101

Chapter 1

Introduction

This thesis is a case study of improving the security and usability of the web-based authentication system at Futurice. Futurice is a medium-sized IT consulting company. Employees, contractors and a small number of customers authenticate daily to Futurice's web-based services. Currently, the authentication is performed once every 12 hours by entering username and password, so called single-factor authentication. After the user provides correct credentials, the authentication service generates a cryptographically signed authentication token for the browser.

This system was designed and built several years ago for a traditional business environment. The principal assumptions included a closed network, with remote access only over VPN, granted on special request. This is not the case anymore. Over half of the employees are working primarily at customer premises. To provide the required flexibility and convenience, almost all internal services are available over the Internet. In addition, nowadays every employee can configure VPN remotely using a self-service tool.

Requiring either working at the office or using VPN provided a second factor to the authentication. In essence, the users prove they know something secret (username and password) and possess something (access to the office, or VPN credentials). Together, these provide what is deemed as strong authentication. Both first and second factor could be satisfied with a number of other mechanisms. For more information, see section 2.1.

At Futurice, requiring VPN for working from off-site locations is not feasible. First, in some cases, VPN causes serious performance degradation. Second, not all customer networks allow VPN connections. Third, VPN connection complicates various development and testing procedures. For instance, connecting to a development phone over the network is more difficult when the VPN is connected. Fourth, requiring preconfigured VPN practically forbids using devices that are not specifically configured for remote

working. Therefore, an alternative second factor is required to provide secure authentication.

Requiring only a password for successful authentication has major flaws. If the attacker is able to obtain the password, there is no mechanism to prevent using it from any device. Even though passwords are supposed to be secure and private, this assumption is not true in practice. First, people tend to reuse their passwords on various services [11, 29, 48]. This happens despite the fact we regularly read reports of remarkable user database breaches at various services. Second, even if users could change the passwords immediately when exposed, they often make only minimal changes to satisfy system requirements of generating a different password[1]. Third, the password could be exposed in various mundane or sophisticated ways. For instance, looking over the shoulder of the user while they write the password could easily expose the password. Malware that logs keystrokes and form input is relatively common. The attacker could obtain a password from various side-channels, including from keystroke timing. For examples of sophisticated attacks against the widely used TLS encryption, see section 2.3.1.

Strong authentication mitigates or eliminates the effects of these issues. Copying a perfect second-factor component is impossible. Even though no such component is available, typical second-factor components are either relatively difficult to copy (e.g., smartcards) or expire quickly (e.g., timed one-time passwords). Additionally, even if the attacker can copy or steal the second-factor component, she also must acquire the associated secret before the second-factor component either expires, or is revoked.

Currently, the most commonly used second factor is one-time passwords (OTP). OTPs are automatically generated passwords used only once upon request. Today, mobile phones are often employed for provisioning the passwords. In Finland, the majority of the online banks use paper slips with numbered one-time password strings. For more information about OTP, see section 2.1. The disadvantages of requiring one-time passwords are the inconvenience and the effort required for the end user.

The goal of this thesis is to implement and evaluate an intelligent authentication system. Based on the data collected from earlier sign-in attempts and user activities on other services, the system will determine the risk level for each authentication attempt, and it will only challenge the user when the risk of illegitimate access is deemed too high. In addition to activity data collected from various services, multiple additional surveys were executed.

[1]Even the relatively well-educated and security-conscious users at Futurice admit they try to make only minimal changes when changing their password. 15% admit using their Futurice password on external services. See section 3 and appendix C.

For more information, see appendices B–F.

When a user signs in, she proceeds through three distinct steps. First, the user identifies herself by asserting a claimed identity. Second, authentication is required. During the authentication the user proves that she is who she claims to be. Finally, after the authentication, authorization takes place. The system now knows who the user is and has to determine what the user is allowed to do. This thesis concentrates on improving the authentication process. Authorization, accounting and system administration in general are not addressed.

The system will be evaluated using three measurements: security (minimizing false positives, i.e., likelihood that an attacker with a stolen session cookies would not be asked to reauthenticate), effectiveness (minimizing false negatives, i.e., the number of authentication prompts legitimate users encounter) and usability (minimizing the time spent on the authentication process). Optimally, no illegitimate access is allowed. An extreme example of perfect security is to deny all access. This is clearly not useful. Therefore, the effectiveness and usability goals are important. Hypothetically, when perfect security is achieved by denying all authentication attempts, other measurements grow indefinitely. Thus, obviously, a compromise is required. Determining the perfect compromise between these measurements is a hard problem that is not solved in this thesis.

Naively, the system could request one-time password (OTP) each time user authenticates. Even though obtaining and entering the OTP consumes less than 30 seconds, recovering from the interruption of the workflow takes significantly longer time. Repeated interruptions annoy people. In addition to the unnecessarily wasted time, technically competent people tend to build workarounds for laborious, seemingly unnecessary tasks. Building a system to enter the OTP automatically certainly shortens the time spent on authentication, but also easily compromises the security.

On the other hand, if the system accepts all available second-factor sources, circumventing the system becomes relatively easy. For example, entering a typical medium or large office without proper keys or ID card is usually straightforward. Thus, the authentication system described in this thesis analyzes all available data to make educated, automated decisions on the required level of authentication. The hypothesis is that in a corporate environment, users mostly follow relatively predictable usage patterns, at least compared to consumers using, for example, Facebook[2]. Consequently, reliable and accurate automatic decisions on the legitimacy of authentication

[2]On the other hand, even if the usage patterns are seemingly unpredictable, Facebook can take advantage of immense amount of data from millions of users.

attempts should be possible.

As the goal of this thesis is not to innovate on data mining methods, only algorithms provided by off-the-shelf libraries are considered. The goal for data processing is pragmatic, as the business environment is changing relatively swiftly and the available datasets are relatively small, thus complicating the implementation of highly specialized custom solutions. As the working environment is constantly changing, two-factor authentication decisions must be general enough to adapt automatically, without frequent manual intervention. Being able to understand the reasons behind the authentication decision is important for both user, and for the people responsible for maintaining the system.

This thesis is organized as follows. After this introduction, chapter 2 describes authentication technologies, basic web security concepts and legal aspects related to this thesis. Chapter 3 presents the case study environment. Next, chapter 4 explains architecture and technical implementation in detail. Chapter 5 addresses data analysis and evaluation of the different models. After that, chapter 6 presents overall results and the performance evaluation. Finally, chapter 7 discusses the some of the potential issues and future developments. The last chapter concludes the thesis.

Chapter 2

Background

Often, users do not intentionally compromise the security of the system they are using [1]. If systems give meaningful feedback to the user, she often maintains better motivation to follow the instructions and avoid taking shortcuts [96]. However, the tools and requirements provided for the tasks users want or need to accomplish are seldom well-designed. For example, users have to remember and use too many complex passwords, but the reasoning for this is rarely explained. Furthermore, both computers and smartphones provide only all or nothing authentication: either authentication is required before any action – even using a calculator – or nothing is protected[1].

Related to the matter of all-or-nothing authentication, in one small-scale study, participants were asked to select the 20 most important applications on their mobile phone and select a protection level for each application. On the average, roughly half of the applications were classified as not important for authentication [39]. Having to always authenticate, even to perform tasks that would not require it encourages users to disable authentication: 30% of mobile phone users have disabled lock screen authentication [83]. Most probably, in a typical corporate environment, not all services require high confidence in authentication. At Futurice, some instruction pages and services could even be publicly available. The authentication system should reflect this.

Riva et al. studied progressive authentication for mobile phones [83]. They found that authenticating mobile phone users is significantly different from computer users. First, computers are typically used continuously for longer periods of time, in contrast to mobile phones, which are used casually for shorter tasks. Second, mobile phones are usually used only by a single person. Third, data in smartphones is more confined to separate applica-

[1]Recently, some functionality has been exposed to lock screen: for instance, taking new pictures and controlling music playback. For example, for iPhone, see [4].

tions. Finally, mobile phones include multiple environmental sensors that could be used for profiling the environment and user behavior. These include Wi-Fi, microphone, camera, accelerometers and GPS. The prototype implementation challenged users with authentication only for sensitive tasks, and only when environmental sensor data indicated that user had probably changed. The number of required authentications was reduced by 42% without compromising security. With computers, requiring authentication only when a sensitive application is accessed is more difficult, as current operating systems are built around single authentication and shared storage. Environments with mainly web applications differ significantly: on the one hand, all authentication decisions are performed on the server side, and web applications are often running in containers, with restricted access to the data. On the other hand, servers have significantly less information available for performing authentication decisions. Furthermore, majority of the information is collected by an untrusted client (browser).

As mobile phones expose a large number of accurate and trusted environmental sensors, transparent authentication is easier. For example, it is possible to identify the user by observing phone movements [14]. Some smartphones provide authentication with face detection – with current hardware and detection schemes, arguably less secure than PIN, but significantly more convenient. Unfortunately, computers typically have fewer environmental sensors. In modern laptops native applications could use camera, microphone, nearby Wi-Fi networks, Bluetooth devices and accelerometer readings when available. Some of the data – camera, microphone and accelerometer readings – are available for web applications, but are subject to a relatively complicated user authorization process.

Security expert Bruce Schneier argued that the main security problem has shifted from the passive eavesdropping to active phishing and man-in-the-middle (MitM) attacks [85]. Unfortunately, two-factor authentication fails to protect against active attacks. For phishing and MitM attacks, user education is the primary response. In the web environment, technologies such as HTTP Strict Transport Security (HSTS) protect against man-in-the-middle attacks by denying insecure connections with unskippable error [42]. Globally, HSTS is supported by approximately 55% of the browsers [17]. At Futurice, support is at 96%[2].

[2]At Futurice, only relatively new hardware and software is in use. This is often the case when the hardware is leased for a limited period of time. See section 3.1 for more information.

2.1 Strong authentication

Strong authentication usually refers to two-factor authentication. During authentication, the user proves any two of the following:

- She knows a secret (often, a password)
- She has something that contains a secret (for example, generator for one-time passwords)
- She is something (i.e. biometrics)

This is also called as "two-step" or "multi-factor" authentication. Often, knowing something secret and having something is required. In this thesis, these are referred as "first factor" and "second factor", respectively.

Second factor for strong authentication could be provided by multiple different mechanisms. The following list is not exhaustive.

- One-time passwords
- Various mechanisms to identify the device used
- Location of the user (closed office space requiring authentication with a token such as ID card or key)
- Biometrics

The following sections describe these mechanisms.

2.1.1 One-time passwords (OTP)

One-time passwords are unique codes that can be used only once. The main advantage of the one-time passwords is protection against eavesdropping – even if the attacker eavesdrops the username, password and OTP, they are not able to reuse them later. Of course, if the attacker can obtain unencrypted server responses (for example, with a man-in-the-middle (MitM) attack), she could obtain authentication tokens, and thus, in some cases, bypass the need to possess any credentials.

Strong authentication using one-time passwords is currently deployed as an optional feature by all major Internet companies, for example, Facebook [89], Twitter [73], LinkedIn [88], Google [32] and Amazon [3].

For generating OTPs, multiple mechanisms exist. The following list is not exhaustive.

Paper slips are often used in online banking in Europe. The paper contains a numbered list of one-time passwords, securely delivered to the user. On the positive side, the lists are completely offline and thus secure against spyware. On the negative side, paper slips are easy to misplace and cumbersome to use. Also, delivering the paper slips to the users is expensive.

Hardware tokens are mainly used in the corporate world. Probably the most widely used and best known hardware token product line is SecurID [86]. The token has a small display showing a time-based OTP. Delivering and provisioning hardware tokens is more expensive than paper slips. On the positive side, copying hardware tokens is difficult.

Mobile phone applications. Google Authenticator [33] is currently one of the most widely used OTP applications for mobile phones[3]. Passwords are generated either from a timestamp or in response to a challenge provided by the service requesting authentication. The algorithms which Google Authenticator uses to generate OTPs are public [65, 66].

SMS/phone calls. While mobile phone applications are convenient, SMS messages are often used as a fallback in case the application is not available. OTP is simply delivered as an SMS or via phone call, using the cellular network as an out-off-band channel.

A very important feature of secure one-time password authentication is to deny the reuse of the passwords, even if time-based OTP is technically is still valid. OTP prevents replay attacks, but only if reuse of one-time passwords is never allowed.

2.1.2 Identifying the device

Identifying the device used could easily satisfy the requirement "user has something", assuming the device itself is protected adequately. For example, if biometric authentication is required to unlock the device – and this configuration is enforced – using the device is effectively proving the identity of the user. Of course, legitimate user could loan their device to another person. However, similarly, they could compromise any reasonable authentication if they wish to do so.

Smartcards are one rather simple mechanism to identify and authenticate the user [76, 99]. Again, of course, nothing but rules prevent loaning

[3]Supported by Google, Dropbox, Amazon, Github, and numerous others.

the smartcard. Sharing the smartcard is not possible, as the smartcard is continuously required. Unfortunately, smartcards require both software and dedicated smartcard readers, which are not generally available.

For the authentication system implemented in this thesis, identifying the physical device is not practical, as no software installation is possible. However, a few weaker indicators are available. First, when the device is connected to the office network, its MAC address[4] is available. Second, if backup client is installed, the IP address reported by backup service should match to the IP address of the user. Third, if the VPN is connected, again, the IP address could be matched to the user. Fourth, the computer uptime calculated from TCP timestamps [49] can provide additional identifying information. However, these indicators are not always available.

In addition to identifying the device, some indicators for browser identity are available. For example, plugins identify the browser relatively well [21]. More exotic and less accurate indicator is accuracy of the clock.

2.1.3 Location of the user

If the user is at a location, which only trusted people could access, that could serve as second factor for authentication. This could be the case with employee-only areas in office buildings. To enter Futurice office, an electronic key is required. However, as typical in low-security companies – companies not in e.g., defense or financial industry – there is no security guard at the door. Entering the office without a valid key or ID card is entirely possible.

One approach to mitigate this problem is to use location information as a second factor only if the user entered the office with their electronic key. However, typically, no mechanism to link electronic door keys to authentication attempt exists. In other words, even if the user entered the office with their electronic key, there is no way to validate that the same user is trying to authenticate. Therefore, only being at the office does not prove legitimacy of the user.

2.1.4 Biometrics

Probably the most widely used biometric is fingerprints. Other examples include voice matching, face or iris recognition, and heart rate [47]. In general, the most important issue with biometrics is irrevocability. If the attacker obtains enough data to mimic the biometric, it cannot be reissued, whereas

[4]MAC address is a hardcoded, physical address of the network card. Forging the MAC address is trivial.

reissuing compromised passwords or authentication devices is a standard procedure. Moreover, almost all biometrics either require a separate accessory, or are easily fooled. Iris recognition is hard to bypass but slow and expensive. On the other hand, sensors for voice or face recognition are available on almost all modern devices, but deceiving those with recordings is often straightforward.

HTML5 offers a media API for accessing the camera and microphone on the client device, and orientation API for device orientation. The camera and microphone APIs could be used for face and voice recognition. However, we did not implement these for the following reasons. First, both APIs are currently drafts and subject to change. Second, training face or voice recognition requires a considerable effort. Third, as noted above, fooling the recognition is easy, as no trusted path from the camera to the server exists. The malicious client can feed recorded video or photos of the user. Finally, implementing and testing the system requires a prohibitively large effort. This feature could be developed on a later date.

2.2 Single sign-on technologies

Single sign-on (SSO) is a general term for sharing authentication information between services. In practice, the user signs in only once to gain access to multiple services. Currently, probably the most widely used and known SSO service on the Internet is Facebook Connect [26]. This section concentrates on SSO protocols, not on service providers.

Multiple protocols for implementing the single sign-on are in popular use, or have received at least reasonable research and implementation effort. The two most widely used technologies are OpenID [79] and OAuth [38]. Both are open standards, the former being authored by the OpenID Foundation and the latter by the Internet Engineering Task Force (IETF). In addition to OpenID and OAuth, SAML 2.0 is used for providing authentication and authorization across organizational boundaries.

OpenID is designed for authenticating users between entities not trusting each other. Of course, some trust is required: the service requesting the authentication has to trust the identity provider to verify user identities properly[5]. Basically, the OpenID authentication only ensures that the user holds a particular identity on the identity provider. Identities are specified by unique URLs, which include the identity provider address. A malicious identity provider can not forge the identities of users in another identity

[5]Furthermore, users must trust the identity provider they choose.

provider.

OAuth is designed for authorization: users can delegate limited access for applications [38]. For example, the user can grant a scheduling application a limited access to the Google Calendar without entering the Google credentials to the scheduling application. OAuth is also used for authentication with a workaround: the service requests access to an API that only supports requesting the identifier of the currently authenticated user. After receiving the OAuth token, the service can then make another request to said API to identify the user. Using OAuth requires registering the application to the identity provider.

SAML 2.0 (Security Assertion Markup Language) [12] is an authentication and authorization framework for inter-organizational identity management. Shibboleth [69] is one implementation of the SAML identity and service providers. SAML provides plenty of features, including authentication, authorization and attribute transfer mechanism. Subsequently, SAML implementations are complex and difficult to configure[6].

In addition to OpenID, OAuth, and SAML 2.0, many other authentication and authorization protocols exist. Majority of the other protocols are either proprietary or have an insignificant user base. Therefore, other protocols are not discussed here. Kerberos [70] is one notable exception. Developed by MIT, Kerberos provides secure SSO over insecure channels. With Kerberos, client devices must be trusted. Furthermore, client devices must be in the same domain, rendering authentication over the Internet more difficult.

All the three protocols – OpenID, OAuth and SAML 2.0 – are designed for inter-organizational authentication and authorization. Company internal SSO systems often differ from inter-organizational systems fundamentally: both the identity provider and the service asking for authentication are trusted and controlled by the same entity. Therefore, simpler implementations could be more appropriate. Also, nothing prevents an organization from offering multiple interoperating SSO services: for example, a simple SSO implementation for internal services and SAML for authenticating to services offered by partners.

For example, domain cookies[7] are one of the simpler approaches. A product called mod_auth_pubtkt [51], among others, implements SSO using cryptographically signed domain cookies. The advantages include straightforward

[6]For example, Shibboleth identity provider is over 25 000 lines of code, excluding libraries. As a comparison, Python implementation for OpenID client and server is approximately 2 000 lines of code.

[7]Domain cookie is automatically sent with every request to the same domain, not depending on the hostname

installation and requiring no modifications to user environments. However, if an attacker can eavesdrop the cookie from any web service within a domain, she can impersonate the original user on any service within that domain [84]. This is different from more complex SSO systems.

Another approach for SSO is an authentication proxy [75]. In the proxy-setup, the browser is configured to transmit all traffic through the proxy. First, the user authenticates to the proxy. After that, whenever the user tries to access a service that requires authentication, the proxy inserts credentials to the request. The credentials are never transmitted to the user. The major advantage is having no requirements for remote services: the proxy can authenticate the user to any service, providing the credentials are available in the proxy database. Moreover, the proxy could allow secure logins from untrusted devices, for example, with one-time passwords. Although the proxy authentication offers certain advantages, disadvantages are even greater. First, the setup process is relatively complicated, requiring configuration of each user device – which could be more difficult than reconfiguring services. Second, the proxy acts as a single point of failure, both for availability and integrity. Finally, HTTPS connections either bypass the proxy, or a special setup is required to handle certificates correctly.

At Futurice, transparently to the user, three separate SSO technologies were implemented. First, SSO based on the domain cookie with mod_auth_pubtkt is used for low-security internal services. Second, OpenID is available for various external services, including hour reporting, and for high-security internal services. Finally, SAML 2.0 provides SSO integration for the Google Apps collaboration suite.

2.3 HTTP

Practically all[8] traffic between browsers and web servers is based on HTTP/1.1 (HyperText Transfer Protocol) [28]. HTTP by itself does not provide any availability, confidentiality or integrity guarantees. For confidentiality and integrity, HTTP over TLS (HTTPS) is used [82].

In HTTP, each request is a separate connection[9], without explicit session or state management. Sessions are implemented with cookies, which are essentially key-value pairs set by the server (or client-side JavaScript), and sent by the browser to the service with each HTTP request.

[8]Some recent additions, including WebSockets [27], provide almost raw TCP connections.

[9]Mechanisms such as keep-alive implement more persistent connections for higher performance, but no session management.

For normal web pages, the browser downloads pages written in HTML (HyperText Markup Language). Typically, all content, including HTML, JavaScript, styling and images are downloaded over HTTP(S). All content is evaluated in a single context[10]. For instance, JavaScript loaded from separate servers or inlined in HTML is executed without any privilege separation. To limit the sources that can provide content for the pages, content security policy (CSP) support was recently added to all major browsers.

The following sections cover TLS, cookies and CSP in more detail.

2.3.1 TLS security

The security of TLS is well known to be flawed [2, 10, 13, 59]. This section is not an exhaustive description of the attacks and solutions, but a limited view of some interesting developments. A full review of the current state of TLS security is an issue for another thesis. Many related topics, including properly configuring supported ciphers are well covered in the documentation of any popular web servers.

On a very general level, attacks can be classified to two distinct alternatives: either the client connects to an incorrect address, or the connection is hijacked. In the former case, the attacker could for example send an email containing a similar-looking address, which is known as a phishing attack. The only reliable solution is user education. In the latter case, either encryption is compromised or a man-in-the-middle (MitM) attack is performed. For the attacker, probably the easiest way to circumvent TLS is to disable it altogether.

In general, users enter the web pages either by following a link or by entering an address [59]. The link could be on the web page, in a bookmark, from email, or is a redirect from another page. When entering the address manually, it is rare to enter the "https://" prefix. When the user enters "example.com", the browser by default connects using unencrypted HTTP. If the server wants to use encryption, it redirects the user to "https://example.com". This, in turn, opens a possibility for MitM attacks, as the attacker can prevent user from switching to a secure connection and proxy unencrypted traffic to the server. The key for invisible MitM in this case is to intercept the connection before it is switched to TLS, thus preventing all security features, including certificate checks. A tool called sslstrip [60] does this. Fortunately, for detecting a generic SSL stripping attacks, a simple JavaScript code is

[10]There are ways to sandbox content, including iframes with the "sandbox" argument. However, practically all content is executed in the same context, including browser extensions.

enough[11]. However, if the attacker is targeting a specific service, disabling or altering relevant parts of the JavaScript code is straightforward.

HTTP Strict Transport Security (HSTS) [42] provides protection against SSL stripping attacks, as long as the browser is initialized properly. HSTS is initialized with a special header, which tells the browser to connect the server only over HTTPS, and to automatically reject all invalid certificates for that server. The browser stores this information. Unfortunately, HSTS only works if the initial connection is made securely, as the attacker can strip the header. Also, globally HSTS is only supported by approximately 50% of the browsers (weighted by popularity) [17]. At Futurice, for over 96% of visits HSTS is supported (see appendix B and section 3.1). As an additional bonus, once the browser has stored HSTS information for a server, all certificate issues – i.e. expired or untrusted certificate – raise an unskippable error dialog, thus preventing the user from ignoring potential attacks or configuration errors.

When HSTS is not initialized or supported by the browser, the only reasonable solution to prevent MitM attacks is a special browser plugin for validating the site certificate fingerprints against a static whitelist. Additionally, a browser plugin could prevent executing other, potentially dangerous plugins. However, not all browsers support plugins inspecting certificate information. The most popular browser at Futurice, Google Chrome, does not provide a reliable API for that. See section 4.5 for a more comprehensive evaluation of plugins.

Encryption provides Perfect Forward Secrecy (PFS) if an exposure of the private key of the server does not compromise earlier traffic [19]. In other words, even if an attacker has eavesdropped and recorded all encrypted traffic before obtaining the private key, past traffic is still confidential[12]. Enabling PFS inflicts approximately a 15% overhead [8] on the initial TLS handshake[13]. This property greatly restricts the consequences of leaked private key. For example, the recent "Heartbleed" vulnerability in OpenSSL might have caused exposure of the private key [74], but PFS still protects the contents of the earlier sessions[14].

[11]With JavaScript, current URL is available in "window.location" variable. Validating the connection is using HTTPS is straightforward. Automatically checking whether the connection is trusted – i.e., certificate is trusted and valid – is not possible.

[12]Of course, with the private key, the attacker can execute man-in-the-middle attacks and thus obtain unencrypted traffic in the future.

[13]As repeating connections are usually resumed, this performance degradation is in fact even smaller.

[14]However, exposed memory content of the server might still provide current confidential information, including passwords, to the attacker.

2.3.2 Cookies

Cookies are the only reliable mechanism for maintaining a session between the browser and the server. Cookies are key-value pairs set by the server or by a JavaScript code executed in the browser. The browser stores and sends all cookies with each request, according to the same origin policy. The policy dictates that cookies are only sent to the server that set them, and that the server can only set cookies for itself, or for its domain. As the browser stores cookies, user can see and modify all contents of the cookies. The recommended approach is to set only a random session ID in the cookie and to store all information on the server side [6].

Even the most recent IETF RFC on cookies admits that there are a number of shortcomings in the security of the cookies [6]. Two different cookie parameters mitigate these problems. First, setting the "secure" option prevents the browser from sending the cookies over an unencrypted connection. Unfortunately, browsers send the cookie even if the connection is untrusted, as long as any encryption is used. For example, if the server certificate is not trusted or is expired, the connection can not be trusted, but secure cookies are still transmitted. Additionally, even unencrypted connections can set and override secure cookies. Second, the confusingly named "httpOnly" parameter prevents JavaScript from reading the cookie. This seemingly bizarre feature prevents malicious JavaScript from obtaining the value of the cookie, while the browser automatically adds the cookie to each request, thus maintaining the session with the server.

Even though "httpOnly" is not a standardized parameter, all commonly used browsers either support or ignore it, instead of malfunctioning. A large number of major websites use it for session cookies [100]. Therefore, most probably no compatibility issues exist. However, large sites could avoid setting the parameter on browsers that do not support it. This was not tested. All browsers visiting the Futurice SSO service during 2013 supported httpOnly.

Together, "secure" and "httpOnly" restrict exposure of the cookies, but they do not protect against active MitM attacks.

In addition to cookies, the only ways to store a state in the browser are localStorage and sessionStorage [41]. LocalStorage is similar to cookies – key-value pairs, only accessible by the domain that set the values – but values are not sent automatically back to the server. Instead, a simple JavaScript API is used. Even though localStorage and sessionStorage contents are not automatically transmitted back to the server, active attacker can craft a simple JavaScript code to obtain the contents.

2.3.3 Content security policy

Content Security Policy (CSP) allows restricting sources of various web content: most importantly, JavaScript, but also all other types. Strict enough CSP effectively prevents cross-site scripting (XSS) attacks. XSS is one of the most commonly exploited vulnerabilities in web applications [98].

Strict content security policy also prevents some plugins from injecting JavaScript and CSS to the page. For example, various social bookmarking plugins rely on running external JavaScript to analyze each page. Unfortunately, some password managers malfunction as they rely on injecting CSS for animations and JavaScript for writing values to form fields.

The CSP policy used in the authentication service deployed at Futurice is further explained in appendix A.

2.4 IP address databases

Finding a physical location for IP address is a task that multiple services want to perform. For example, a web store could automatically offer an estimate for postage. The process of finding the physical location for an IP address is called geocoding.

Information about domain names and IP addresses is stored in the WHOIS databases, accessed with web services or over the WHOIS protocol. The protocol is specified in RFC 3912 [15]. However, only the network protocol is defined. Each domain registrar and regional Internet registry (RIR) operates their own WHOIS databases. No common data schema exists. No comprehensive studies or libraries for different data formats or for the accuracy of the data were found. For an example of WHOIS data, see appendix G.

In addition to accuracy of the information, validity is an important question. For generic top-level domain names, false contact information may be a reason for cancellation of the registration. All domain registrars must send yearly reminders to the registrants [43]. RIRs are either discussing or have implemented similar measures [52]. With country-level domains, local laws often require up-to-date contact information.

A company called MaxMind provides databases and services for geocoding [61]. The database is collected from multiple sources, including WHOIS. This approach requires constant updates as IP networks are repurposed and reassigned. The resolution of the database is only to a city-level, in the best case.

Both the WHOIS and geocoding databases store the location information of different sized networks of IP addresses. For example, a typical IPv4 block

for a consumer address pool is /16 (65536 individual addresses) or larger. Often, in the WHOIS records, these networks have only a single physical contact address, even though the IP addresses are routed to end users over a large geographic area. MaxMind's databases have information with better resolution and accuracy information.

For further discussion and analysis of the geocoding databases, see section 5.2.

2.5 Evaluating the security of information systems

Multiple frameworks for evaluating the security of information system exist. ISACA defines information security as follows: "[Information Security] ensures that within the enterprise, information is protected against disclosure to unauthorized users (confidentiality), improper modification (integrity), and non-access when required (availability)" [45]. Additionally, ISO/IEC 27000:2012 notes that "other properties, such as authenticity, accountability, non-repudiation, and reliability can also be involved." [46]

For the authentication system created for this thesis, all seven properties are important. Quotations in the following list are from ISO/IEC 27000:2014 [46].

Confidentiality: "information is not made available or disclosed to unauthorized individuals, entities, or processes". In practice, no information – data, results or authentication information – is available for unauthorized access.

Integrity: "protecting the accuracy and completeness of assets". For instance, attacker can not alter the data utilized in authentication decisions.

Availability: "being accessible and usable upon demand by an authorized entity". The system should be accessible for users. Single sign-on represents a single point of failure, as no service is available when SSO is not functional.

Authenticity: "an entity is what it claims to be". Authenticity is important both for users (users are who they claim to be) and for data (data collected from various locations is accurate).

Accountability: "assignment of actions and decisions to an entity". Accountability for both users, and for various data sources is important in case of problems.

Non-repudiation: "ability to prove the occurrence of a claimed event or action and its originating entities". Non-repudiation is closely related to accountability: when users are accountable for their actions, non-repudiation is usually achieved.

Reliability: "consistent intended behavior and results". For reasonable usability, consistent and proper behavior is essential. For example, if the system arbitrarily denies login attempts, users quickly grow frustrated.

Relation of the implementation and these properties are further elaborated in section 4.1 and particularly in table 4.2.

2.6 Legal aspects

For a multinational company, jurisdiction is an important question. However, in this case study, IT services are provided by the Finnish company and invoiced internally from subsidiaries. Furthermore, Futurice does not have subsidiaries outside EU[15]. Therefore, only Finnish and EU laws have to be considered.

In Finland, three laws cover the majority of the work related privacy issues: Act on the Protection of Privacy in Working Life [54], Personal Data Act [40] and Act on Co-operation within Undertakings [53]. The relevant parts are introduced and discussed below.

The *Act on the Protection of Privacy in Working Life* aims to clarify rules of processing private information at workplaces. The act specifies the requirement of the necessity of the data collection. No data – even with employee consent – may be collected if it is not necessary for the employee-employer relationship. In the intelligent authentication, the necessity of the data collection is clearly established. Providing an option for disabling the data collection and processing could be appropriate. However, disabling the data collection also means that no automatic decisions could be made and the default must be requiring two-factor authentication. The only other relevant clause in the law is the requirement for co-operation negotiations between employer and employees before taking technical monitoring into use.

[15]Except recently – at the beginning of 2014 – established subsidiary in Switzerland. Unfortunately, due to time constraints, this is not addressed in this thesis.

The *Personal Data Act* restricts how personal data can be collected and processed. Also, the act mandates creating of description of the data registry (chapter 2, section 10). The act provides five general requirements for processing the data (sections 5-9). First, a duty of care: the controller of the registry must act "lawfully and carefully, [and] in compliance with good [data] processing practice [...]". Second, the processing of private data must have a defined purpose. The purpose must be "appropriate and justified [...] in the operations of the controller". The purpose must be defined before collecting the data. Third, exclusivity of the purpose: the data can not be used for any other purpose than what was defined before collecting it. However, historical, scientific and statistical research are specifically allowed. Fourth, there are general prerequisites for the data processing, mostly not relevant for this case study. However, two relevant requirements are presented: the unambiguous consent of the data subject and a connection between the subject and the controller. For employees, the consent is given when signing the work contract, and at the same time, connection between the employee (the subject) and employer (the registry controller) is formed. Fifth, the act defines two data quality principles: the necessity and accuracy requirements. The collected data must be necessary for the declared purpose, and "erroneous, incomplete or obsolete data" must not be processed. Finally, the act specifies the subject's right of access to the personal data. In practice, in this case study, the right to access the personal data is achieved by providing an option to download a machine-readable data dump of the user's personal data.

The *Act on Co-operation within Undertakings* does not actually cover privacy issues, but the act specifies certain interactions between the employer and employees. Section 19 lists specific cases requiring co-operation negotiations, including "3) the purpose, implementation and methods used in camera surveillance, access control and other technical employee monitoring". It is not clear whether making automatic authentication decisions based on analyzing employee behavior falls under the domain of the act. However, the co-operation negotiations required by law are relatively straightforward, and fair practice from the company. In short, the employer has to tell the employees why and what is being implemented, and address any concerns raised.

As a summary, the Finnish law establishes certain restrictions for the data collection and processing related to intelligent authentication. The purpose of the data collection must be defined, and the data must not be used for any other purpose. Also, only relevant data should be collected and processed. However, the law does not fundamentally restrict implementing the system, and the bureaucracy and restrictions imposed by the law are reasonable.

The relevant European Union directive [25] is implemented by Finland's laws. The directive also specifies that EU member states may not restrict the movement of personal data ((9) on the preample). Therefore, the laws in EU countries should not restrict what is described above.

The European Commission proposed unified regulations and a directive in 2012 [24]. The goal is to unify data processing legislation to a single law that covers all activities in any EU member state, instead of a relatively loose directive, which is implemented differently in each country. The European Parliament voted for a strong support of the reform [23]. For a company operating inside EU, the only highly relevant change is that companies collecting personal data must obtain explicit consent from the users. In practice, the service – or application – must prompt for the consent. Implementing this to the authentication service is straightforward. For companies outside EU, the proposal contains major changes to the current state of the legislation. Currently, the company only follows legislations and regulations of the country from which it is operating. With the proposed regulations, any company offering services to users within EU must fully conform to the relevant EU law.

Chapter 3

Operating environment

Futurice is a medium-sized software company based in Finland, with small offices in Berlin, London and Lausanne, Switzerland. The Berlin, London and Lausanne businesses operate through local subsidiaries. The line of business is software consulting.

Out of the 191 employees, the majority are male (M159 / F32, 83% / 17%). The average age is 31 years. 30 (16%) work part-time. 163 (85%) are working in Finland. The majority of employees are software developers and technically oriented designers. Almost 60% have a Master's degree or higher[1]. Approximately 15% have a Bachelor's degree or equivalent. For the rest, the level of education is not stored in the HR database.

Three defining characteristics of the working environment are transparency, empowerment, and constant change. First, transparency is practiced at unconventional level. For example, large amounts of confidential client information and all financial information, including project financials, are available to all employees. Similarly, company board presentations are available. Second, the high level of transparency allows employees to make educated decisions. This is further assisted by employer-provided credit cards [30]. Third, as the company has grown rapidly and the business environment is changing, the working environment and organization structure are changing too. For example, over the past two years, the primary working location has shifted from the company office to customer premises for a majority of the employees.

There is a moderate amount of freedom in choosing working times and locations. Almost everyone goes to customer meetings and other work related events at least occasionally [30].

Before deploying the new authentication service introduced in this thesis,

[1]National average in Finland for similar age range is 13% [91].

a short survey was performed (see appendix C). The goal of the survey was to investigate the opinions towards two-factor authentication. In the survey, 56% of participants claimed they are using two-factor authentication on some service, in addition to online banking (appendix C). Users who were already using two-factor authentication were more receptive to the idea of using two-factor authentication at work: 96% of this group responded "yes", versus 85% for others. Also, of those already using two-factor authentication, 63% proposed two-factor authentication as mandatory for everybody, versus only 29% of the others.

3.1 Technical environment

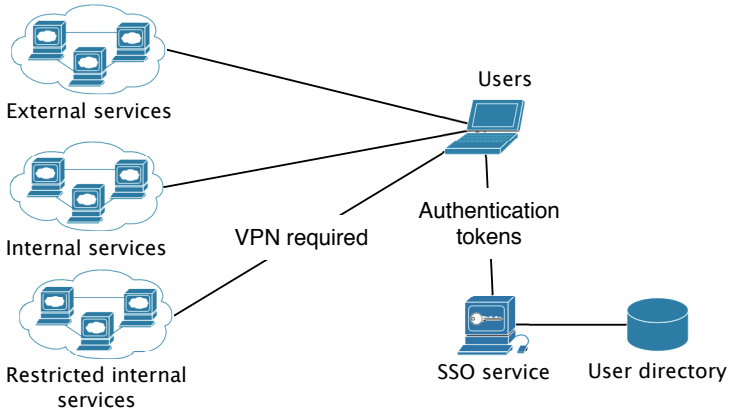

Figure 3.1: Overview of the environment

Overview of components important for this thesis is presented in figure 3.1. Almost all IT services at Futurice are offered as web applications, protected by SSO service. Internal services are managed – and often developed – by Futurice. Therefore, it is possible to perform any configuration changes, if necessary. As previously used SSO technology explicitly required HTTPS, all internal web services are protected by HTTPS, with certificates purchased from a widely recognized certificate authority.

In addition to the internal services, various third-party services are used. For introducing a new service for personal, team or project use, no corporate approval is required. Therefore, a large variety of different services are in use. In addition to various "unsupervised" services, the whole company uses Google Apps for Business for emails, mailing lists, shared storage and collab-

orative documents (Google Drive). The Google Apps user info and passwords are automatically synchronized from Futurice's user directory (LDAP).

For internal services not accessible over the Internet, a VPN service is available. Users can install the VPN on any compatible device, including devices not owned by the company. The VPN is configured through a web application. The authenticity of the request for a new VPN certificate is verified with a one-time password. The password is sent to the user's phone and subsequently entered to the web service. After installation, the VPN connection is authenticated using a certificate. While using VPN, the computer is connected to the office's LAN. The main concerns with the VPN are unsatisfactory performance and reliability, and connection problems from customer premises with restrictive access rules. The VPN service functions over UDP. However, even on the restrictive networks, HTTPS is typically permitted. Therefore, requiring VPN reduces the connectivity in some important cases.

Authentication and authorization information, along with various user attributes, is stored in an LDAP (Lightweight Directory Access Protocol [87]) database. The user password is verified by the LDAP server, and retrieving passwords from the database is not permitted. In addition to password validation, almost all authorization information (user groups) is stored in LDAP.

Four distinct systems requiring authentication are regularly used by employees:

Internal web services All internal web services are protected by single SSO server. Only single-factor authentication with the LDAP username and password is required.

External services usually have separate authentication schemes.

SSH connections to servers the LDAP username and password are used.

Workstations No centralized authentication or authorization (for example, Active Directory) is in use. People use local accounts.

This thesis concentrates on improving the first item, SSO for web services. Simultaneously, SSH authentication is being migrated to stronger authentication, thus reducing the attack surface available with only usernames and passwords.

In a survey conducted before deploying the improved SSO service, 15% of the employees admitted they reuse the same password in external systems (see appendix C). This problem is well documented in the literature

[11, 29, 48]. No technical measure can reliably prevent password reuse. Additionally, in the same survey, only 57% of the employees claimed they create a completely new password every time they are required to change it[2].

On an average week, 171 employees use the SSO service at least once. Similarly, on average, 160 employees sign in to Google Apps each week. Before this thesis, the Google Apps login was separate from the Futurice SSO service. The SSO authentication was valid for 12 hours, and Google Apps authentication for either the current session (until the browser was closed) or for 30 days (when the user selected "Remember this computer").

All employees have a laptop and a mobile phone. Additionally, a growing portion, currently approximately 30%[3], have a tablet. Usually, the devices are purchased, installed and configured by the centralized IT department. However, this is not always the case. As noted, the employees have credit cards and thus a convenient way to buy equipment when certain constraints are met. In that case, the IT department does not necessarily even know about the new device. Additionally, users retain full administrator rights on their devices. Furthermore, laptops and mobile devices are not centrally managed. Thus, automatically provisioning applications or modifying the settings of the devices is not possible. Using work laptops on free time is permitted. Indeed, 50% claimed to use the laptop for private purposes regularly (appendix C).

All laptops are leased for two years, thus limiting the oldest device and operating system to 24 months. When changing the laptop, the OS is usually reinstalled and not migrated from the previous device. The user data is of course migrated.

People update their operating systems and browsers regularly. Over 66% of visits to the SSO service were with a browser released during the last two months (see table B.1). For 6 months of observation (2013-04-01 – 2013-09-30), CSP was supported by at least 96% of the visits and HSTS by 54% (see table B.2). For the last month of the period, CSP was at 97% and HSTS at 65.5%. The difference is explained by the growing usage of Chrome. Migration to Google Apps during the observation period most probably contributed to the shift. Detailed statistics of the browsers are available in appendix B.

No registration or authentication is required for connecting to the office network. For authentication, technologies such as 802.1X[44] or captive portals could be used in the future.

[2]Normally, once per year at Futurice.

[3]Extracted from internal inventory at 2013-10-30.

3.2 Implemented data sources

In addition to the authentication service, multiple other sources were used to obtain user activity data. The criteria for the data sources was as follows. First, the data source must have an API or other interface for obtaining the data automatically. Second, all the data must be linked to a specific user. Finally, only data sources providing information about a relatively large portion of the users were considered.

Table 3.1 lists the implemented data sources. Data sources marked with * are further explained below.

Table 3.1: Details of implemented user activity data sources

Source	IP	Timestamp	Entries per day (average)
Backup server*	✓	✓	400
Electronic doors *		✓	300
Google docs *	✓	✓	78
Google logins [1]	✓	✓	70
SSH activity *	✓	✓	355
SSH logins[2]	✓	✓	88
Timestamps of sent emails		✓	416
VCS commits		?[3]	100
VCS pushes		✓	100
VPN connections*	✓	✓	154
Web activity*	✓	✓	2500
Yammer messages[4]		✓	9

[1] Google login information will not be updated after the new SSO service is introduced. This serves only as a training and validation data.

[2] Including login and logout times.

[3] Spoofing commit timestamps is easy, because data is recorded by the client.

[4] Private social network service. See `yammer.com` for more information. Referenced at 2013-04-14.

Backup server. Workstations maintain a constant connection to the backup server without any user interaction. The connection is open even when the user is not signed in. Therefore, an open connection is only an indication of the location (IP address) of the device at any given time. Additionally, the connection is automatically or manually closed for various reasons. First, when the laptop is running on battery power, connection is automatically closed when certain threshold on battery level is reached.

Second, people pause the backups while performing resource-intensive tasks. Third, when connected over slow or expensive connection, backups are often paused. Fourth, some networks have restrictive firewalls blocking the connection.

Google Docs offers two different ways to obtain user activity. First, an audit log [36] is available. The audit log includes entries for some of view and edit events. No logic for filtering is documented, and could not be easily deduced. Second, Drive SDK provides API [35] for downloading changes. Also, upload times for the majority of the files can be downloaded using the same API.

SSH activity log is based on pseudo terminal device (PTY) access times. Each interactive SSH connection is connected to PTY, represented by a device file in Linux. Every time user interacts with the connection – sends keystrokes – PTY's access time is updated. Same mechanism is used by standard Unix tools[4] to determine users' idle times. Access time is not refreshed for non-interactive actions, for example, when a program prints output to the user, or for protocol packets. Each server checks access times for all active PTYs every five minutes and stores the timestamp of the last activity for each user.

VPN connection entries include device name, remote IP address, connection and disconnection times, and additionally, information about currently active sessions. VPN device names are connected to usernames and should only be used by a single device. However, no remote device information – for example, MAC address, device model or serial number – is available to the server. Each user may have more than one VPN configuration, used in different devices. No activity information is available. VPN is not automatically disconnected on inactivity, as long as the client responds to keep-alive packets.

Electronic doors access data is only available for Helsinki office. Opening the door for others is relatively common. Therefore, the absence of a recent entry for a user in the electronic doors log does not necessarily imply that the user is not at the office. Furthermore, use of the key is not required for leaving the office.

Keystroke timing data includes millisecond-resolution timing for entering the username and password. The data is collected using a client-side JavaScript code. Therefore, spoofing is straightforward, assuming the attacker knows the correct timing. Keystroke timing is hard to imitate, and identifies users with reasonable accuracy [7, 37, 50, 57]. However, if person's writing rhythm is known, it is possible to learn to imitate others [92]. As

[4]See *bsd-finger*, or *who* from GNU coreutils.

simple JavaScript code can record timing data, it is hardly a secret. An attacker can obtain some timing data by asking user to write something on a page they control. However, people regularly write their passwords, which are often a complex series of characters. Consequently, they probably have different keystroke timing for that specific sequence of characters. Therefore, inferring the timing for their password from random text might not be possible. Inferring the timing from the network traffic is possible. If the keystroke timing is known, characters can be deduced [90]. Thus, as there is no need to send realtime updates for keystrokes, timing data is sent at once with username and password.

For further observations and analysis, see chapter 5.

3.3 Potential other data sources

The following data sources were evaluated but not implemented for various reasons. The purpose of this list is to show examples of possible data sources for future consideration.

Creation times of calendar entries obviously indicates user action. However, after briefly evaluating the data, implementation was left for later date. Large amount of filtering is required, as for example delegated calendars and automatically accepted invites create erroneous entries.

Credit card usage could be used to determine when user was working. However, delay in obtaining the data is prohibitively high. Entries from the previous month are available at the end of the month. Furthermore, only the date is available.

Face recognition with security cameras High-quality recordings from security cameras are available. However, as only a single photograph per employee is easily available, initial testing using Haar cascades [55] and Eigenfaces [93] produced unsatisfactory results.

Hour reporting entries At Futurice, only the total number of working hours per day is collected. In some companies, hour reporting includes check-in and check-out times for both working days and breaks such as lunches. In that case, hour reporting data would be remarkably more useful.

Legacy network drive Old Samba based network drive was almost entirely replaced by Google Drive. Additionally, to use the network drive,

either device has to be at the office, or VPN has to be connected, which already provides considerable amount of data.

Salesforce [5] A reasonably good API is available, but the current usage is relatively low. Therefore, implementation was left for later date.

Skype Even though Skype is widely used at Futurice, no support was implemented, as no stable API exists. Additionally, people tend to use Skype even when they are not working.

[5]Customer relationship management (CRM) service. See www.salesforce.com. Referenced at 2014-05-04.

Chapter 4

Implementation

The logic of the system was implemented as a finite state machine. Certain state transition decisions are based on the data analysis, which will be detailed in chapter 5. Figure 4.1 illustrates machine states and decisions.

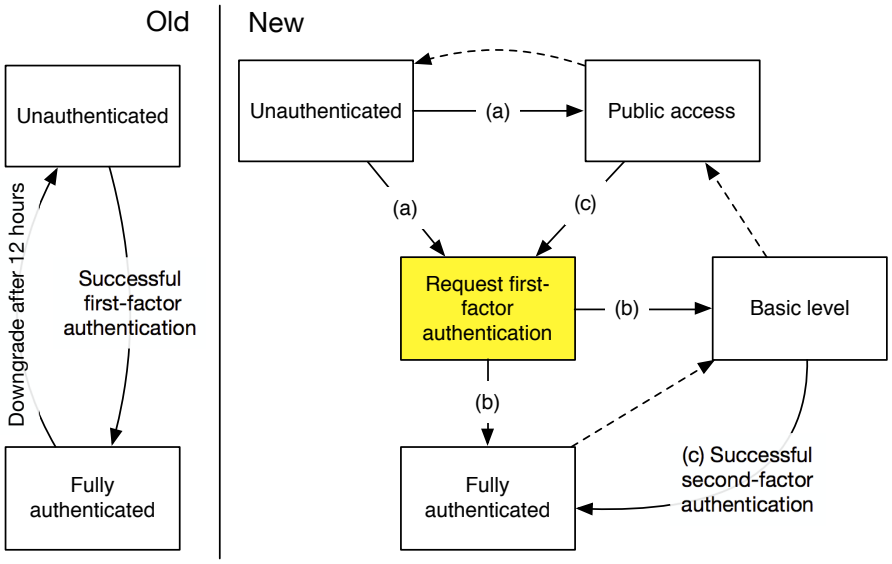

Figure 4.1: Authentication state machine. (a) and (b): automatic authentication decisions. (a): if the browser or device is already known, some access can be granted without any authentication. (b): if enough indicators support the legitimacy of the user, no second-factor authentication is required. (c): state transition when higher access level is required. The dashed lines denote automatic downgrades. Multiple downgrades may happen consecutively.

In addition to the state transitions in figure 4.1, the authentication state

may be downgraded in certain cases, which are listed in table 4.1.

Table 4.1: Authentication state downgrade mechanisms

Case	Action
Password has been changed	first-factor authentication
Password expired	first-factor authentication[1]
Phone number changed	second-factor authentication.
Reconfigured second factor	First factor might be required.

[1] This effectively signs out the user until the password is changed.

4.1 System architecture

Figure 4.2 describes the high-level components and changes made. The separation to open services and high-security services only available on LAN or over VPN was removed. Additionally, the Google Apps authentication is provided by the new login system. Obviously, the VPN service is still available even though it is not required to access internal services.

Internally, the authentication service is divided into the frontend and backend servers. The frontend server is the only component users can access directly. It handles the following actions:

- Rendering the pages for the user
- Validating the username and password from LDAP
- Validating two-factor authentication tokens
- Signing authentication tokens

The backend server is responsible for the following actions:

- Retrieving and storing decision data from various services
- Making authentication decisions

Figure 4.3 details the components for both the frontend and backend services. Each box represents a separate part of the system. As system is expected to change in the future, the colored components were designed to use well-defined, unified APIs. Thus, adding new functionality, including authentication providers and data sources, is easy.

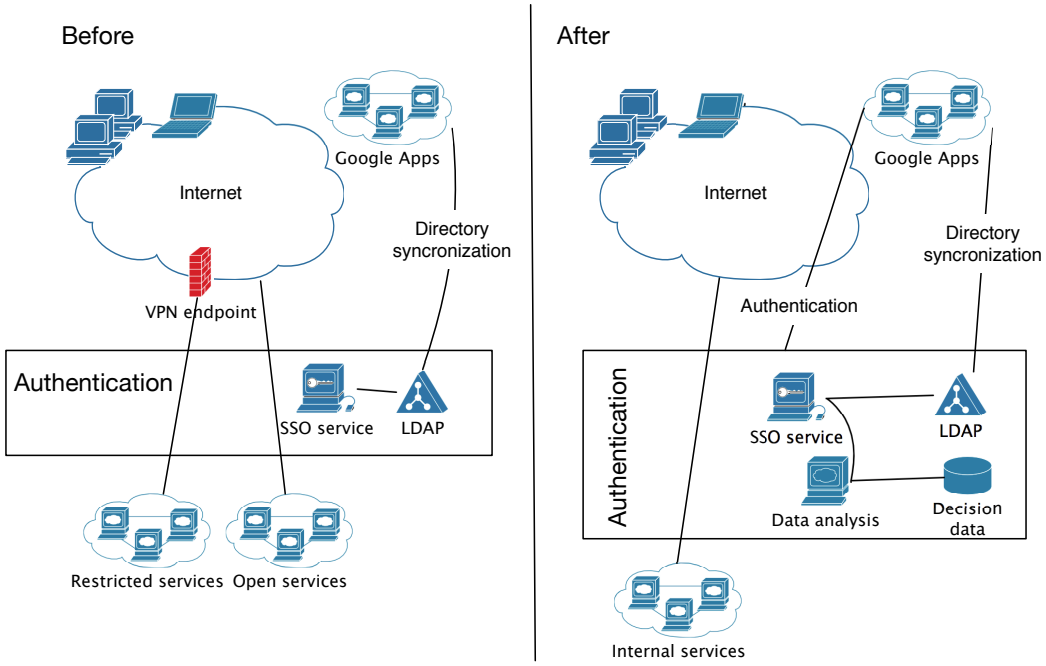

Figure 4.2: High-level overview of the system components before and after the implementation. All connections are over HTTPS.

All communication between the components is performed over HTTPS. Dashed lines are database connections. In case of performance issues, deploying the components to separate servers is only a matter of changing the addresses in configuration files. Additionally, duplicating the components for fault-tolerance is straightforward. The only component that is not trivially duplicated is the decision database. However, many database solutions offer built-in high-availability support.

This relatively simple architecture is designed to achieve the seven security related properties introduced in section 2.5 as well as is reasonably possible. Unless the attacker is a trusted insider – system administrator at Futurice – the typical attack scenario is a compromised frontend server. Two distinct levels of compromise are considered: read-only – i.e. ability to eavesdrop the traffic to and from the legitimate users – and read-write access. Protecting against a malicious trusted insider is not considered in this thesis. Table 4.2 evaluates the security properties in case of a security breach.

These properties could be improved further by building a more complex architecture. For instance, migrating the authentication token signing to a separate server – or to the backend server – could improve accountability and

Table 4.2: Security evaluation in case of a security breach with read-only or read-write access

	Read-only	Full read-write access
Confidentiality	Possible to protect against passive eavesdropping, assuming an earlier, secure connection has been made.	Attacker is able to obtain all credentials users send and all authentication tokens the system delivers to the users. Attacker can not obtain decision data or user credentials[1]
Integrity	Data integrity is preserved.	Integrity of the LDAP data is preserved[2]. Obviously, the attacker can modify some of the data sent to the backend, including the IP address of current user. The integrity of the data collected from another sources is not compromised.
Availability	If the backend server is not available, the frontend server could still authenticate users, falling back to requiring two-factor authentication. Additionally, duplicating both the frontend and backend servers for higher availability and performance is straightforward. The LDAP server is already duplicated on multiple locations.	
Authenticity	Compromised frontend server compromises the authenticity of the server, as no secure TLS co-processor or similar setup is available to protect the encryption keys[3]. Separate server or gateway could be deployed mitigate the compromise of encryption keys.	
Accountability and non-repudiation	Assuming an earlier, secure connection, accountability and non-repudiation of both user and server side data is protected.	Decision data collected from other services is accounted for. Accountability of user actions is compromised, as the attacker could generate authentication tokens for any user. Even if the architecture of the service would prevent the attacker from generating arbitrary authentication tokens, she could still use any token assigned to legitimate users.
Reliability [4]	Reliability is preserved.	Attacker can disturb the reliability of the user experience. Decision data collection still functions properly.

[1] LDAP setup only allows validating username and password, not retrieving password entries. The attacker is able to deduce some of the decision data by querying the backend server. This could be further improved with browser plugins, see section 4.5.

[2] However, the attacker can generate authentication tokens allowing access to the LDAP management service, thus compromising the integrity of the data on LDAP. This could be prevented with additional authentication.

[3] Depends on the level of compromise – obtaining encryption keys requires superuser (root) access. Thus, vulnerability in the authentication system does not necessarily compromise the keys.

[4] It is important to note the reliability – not to be confused with availability – is reduced, compared to the old authentication system. Users can no longer know when additional authentication is required.

non-repudiation, among other properties. However, this would compromise availability in case of issues with the backend server. Furthermore, securing authentication token signing requires moving all other components – except rendering the pages for the user – to a separate server. The only benefits of moving only token signing from the front-end server would be securing the signing keys and preventing token generation for users who did not sign in during the compromise. However, neither of the advantages outweighs the disadvantages. In any case, if the login service is compromised, all signing and encryption keys must be regenerated and redeployed. Also, as the majority of users sign in daily, limiting the damage to only those who did is not valuable.

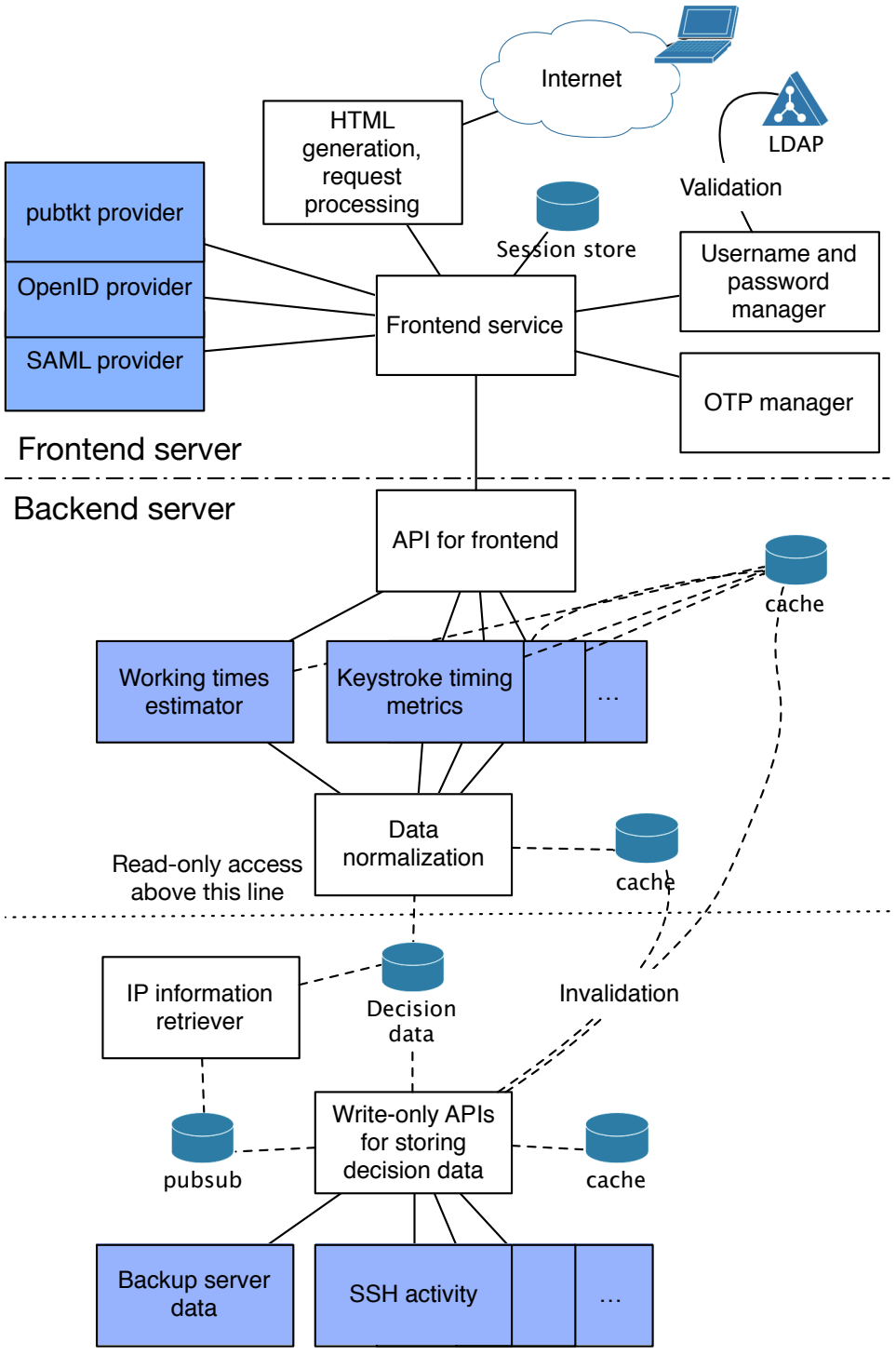

Figure 4.3: Overview of system components.

4.2 Data collection

Data collection components are detailed in figure 4.3 above. Each data collection component is executed independently on the various servers. Majority of the data is not real-time, for multiple reasons. First, third-party services impose ratelimits for retrieving the data. Second, available computing and IO capacity restricts the polling interval. This is the case with sent email timestamps – each update requires synchronizing all emails to a local server. Third, not all the data sources are reliable. For example, with the electronic doors, even though all entries are reliably collected, those are only available intermittently. Except for sent emails, the majority of the data is always available within 15 minutes.

Each component transmits the data to the backend server through write-only APIs. In effect, the data collection components are not able to retrieve any already stored decision data. Three different APIs were implemented:

Deduplicating API for small-scale datasources. The API automatically deduplicates input, i.e. only adds unique entries. This makes implementing the data collection straightforward as the collection component does not have to store state.

Add-only API does not perform any checks against duplicate entries for higher performance. This is primarily used for ephemeral data. For example, recent activity entries – by their nature, appearing only once – are stored using this API.

Time-range APIs for various components with time ranges. For example, for the VPN, the currently active connections are available. When the API receives the data, it automatically tracks open sessions, and only stores start and end timestamps.

Each API normalizes all timestamps to UTC, and stores users timezone information when available.

4.3 Ratelimiting

As the authentication service is a gatekeeper for every service it protects, it is also a single point of failure. In other words, the availability of the SSO also affects the availability of other services. Of course, users possessing a valid authentication token could continue their activities.

The frontend server maintains per-IP ratelimit counters with the following mechanism. For each request, multiple counters are incremented. For IPv4 addresses, counter for /24 (C-class, for example, 123.45.56.*) is used. For IPv6 addresses, netmask /48 (a recommended end site network size [68]) is used. The counters do not track single IP addresses, as changing the address within a local network is usually effortless. This is especially true with IPv6. Storing the counters consumes resources – memory – on the server. These counters are maintained for multiple time intervals to prevent both rapid bursts of requests, and an abnormal amount of requests over a longer period of time.

In addition to mitigating denial of service (DoS) attacks, it is important to prevent or mitigate brute-force attacks against user credentials. For this, {username, authentication method} counters are used. These counters were adjusted based on the strength of the credentials, detailed in table 4.3. It is important to note that the attacker must obtain the correct password before attacking any other authentication method. Therefore, the system could raise an alarm when the ratelimit for any second-factor method is reached. For passwords, this is not necessarily meaningful.

Table 4.3: Ratelimits for different authentication methods

Method	Strength	Ratelimit	Search space exhaustion[1]
Password	50^{10} - [2]	300/1min [3]	619M years
Authenticator	$0.9997596 * 10^6$ - [4]	5/1min & 200/24h	13.7 years
SMS	10^8 - [5]	5/1min & 200/24h	1370 years
Emergency codes	36^{20} - [6]	300/1min [3]	$8.4 * 10^{22}$ years

[1] Time required for exhausting the search space with imposed ratelimits.
[2] Underestimated. In theory, this is at least $(26+26+10+20)^{10} = 82^{10}$, but in practice, at least some assumptions can be made to reduce this. Furthermore, dictionary attacks may reduce this significantly.
[3] Arbitrarily high to mitigate potential DoS attacks.
[4] Each code is valid for 90 seconds. The distribution of the numbers is not perfectly uniform. This is corrected with the multiplier [65]. Each code is independent.
[5] The code is valid for 15 minutes after it is requested and sent. Each code is random and does not depend on the previous code.
[6] Emergency codes are randomly generated. The user can not modify the codes.

All counters – all time intervals for each IP address – are checked for each request. If the maximum value for any of the counters is exceeded, the user will see an error page, explaining the problem, with contact details for the IT team. Ratelimit counters were implemented with an in-memory

database called Redis [80]. The time complexity for each required operation – atomically fetching and incrementing the per-key values – is $O(1)$ [81], i.e., each operation require a constant time, not depending on the number of entries. A complete ratelimit check consumes less than 0.1ms. Sign-out requests by authenticated users are ratelimited separately.

Based on a quick experiment, a single counter entry consumes approximately 120 bytes of memory, including the database overhead for the various data structures. With a moderate 512MB memory limit, approximately 4.4 million counters (1.1 million IP networks) could be stored. For reference, 16.5 million C-class IPv4 networks exist. In case of running out of memory, the database evicts random keys.

4.4 Second-factor authentication methods

One-time passwords (OTP) were selected as the primary second-factor authentication. See section 2.1 for more information about the alternatives.

After evaluating the different service providers and applications, we selected Google Authenticator based on following observations:

Price Google Authenticator is free. Alternatives such as Authy [5] and Duo Security [20] cost approximately $5 000–10 000 per year.

Performance Alternatives require sending at least one request to the service provider. On average, this takes approximately 500ms, which certainly hinders the user experience. Both the client and server side components of Google Authenticator function offline. Validating OTPs with a Python library consumes approximately 0.05ms[1].

User experience Google Authenticator works with a large number of other services. This means users do not have to install separate application for this service. Also, with our own server implementation, more information can be given to the user: for example, the code is not accepted because the mobile phone's clock is off, or because the entered code is generated with a configuration that is already revoked.

Financial stability There are open source implementations for both mobile phone applications and for server side libraries. Using Google Authenticator does not depend on Google.

[1] High performance of the library allowed enhancing the user experience by implementing additional validations. See below for more information.

Security No third-party services are required. The attack surface is significantly smaller.

Despite the name, Google Authenticator works with arbitrary services without any interaction with Google. The application generates six-digit one-time passwords (OTP) based on the shared secret and either a timestamp or counter. For this implementation, timestamp based codes are used. The algorithms for generating the OTP are public [65, 66]. The Authenticator application is configured by scanning a QR code from the browser. The QR code contains a special URL with the name, shared secret and configuration information.

As generating the codes is virtually free, we implemented a few user experience improvements:

1. The OTP provided by the user is checked against the current, previous and next timestamps, and if a match is found, the code is accepted. This permits ±30 seconds offset to the clock synchronization. A small offset is relatively common, occurring in approximately 20% of sign-in attempts.

2. If no match is found, all codes for next and previous 15 minutes (that is, 60 codes in total) are checked. If a match is found, an error message detailing the user clock offset is shown. This occurs rarely, with only approximately 2% of sign-in attempts. If the code is in future, it is added to the list of used codes, to prevent a potential eavesdropper from using it at the correct time.

3. Finally, the same steps are executed for previous ten Authenticator configurations, if available. If any of those matches, a relevant error message is shown, and the user can opt to SMS authentication instead. Old configurations were encountered only in approximately 1% of the sign-in attempts.

In the worst case, 660 code checks are executed. This consumes approximately $0.05ms * 660 = 33ms$, which is perfectly reasonable delay to the login process. If necessary, this process can be distributed to multiple computing nodes. Furthermore, in case of performance issues, both steps 2. and 3. can be removed temporarily without affecting legitimate sign-in attempts. This reduces the maximum processing time required to $0.15ms$.

The attacker could deduce some information using this timing information. For example, the time required to check incorrect codes increases after new configuration is generated (due to step 3., until the user generates her

tenth configuration). This information could be used for social engineering. Of course, timing information is only available if the attacker has stolen valid session cookies or password for the user.

If Authenticator is not available or it is not configured, an SMS acts as a backup mechanism. If SMS authentication is requested, an 8-digit OTP is automatically sent to each phone number the user has. The codes are valid for 15 minutes. Some popular phones show the contents of the SMS on the lock screen, even if a PIN or similar authentication is required. A small-scale study of the most popular phones at the time of writing gave following results:

iOS 6/7 : displays a large amount of text, but only the first four lines. By default, the SMS contents are shown even if locking mechanism is enabled.

Android 4.x : if any authentication is configured, no contents are shown. With no locking mechanism configured, the whole message scrolls once. This could be changed by installing a third-party lock-screen widget. Additionally, some manufacturers could deliver such widgets by default.

Android 2.x : depending on the length of phone number, text equal to the width of 9-18 capital M characters is visible[2]. Line breaks are stripped.

Windows Phone 8 : text equal to the width of 18 capital M characters is briefly visible.

To limit the impact of this, a prefix message ("Your OTP is below. This was requested from [IP address] ([country code]).") and four line breaks were added to each text message. Obviously, this does not protect against phone theft if no lock mechanism is configured.

In addition to OTPs, the system supports emergency codes in case of a lost or unavailable mobile phone. The emergency codes are longer, static codes that could be used as a substitute for the OTP. The emergency codes should be printed, and stored securely. In the future, the codes could be printed to the back side of ID badges. This potentially exposes the codes, but on the other hand, it effectively discourages storing the emergency codes on the computer.

[2]M is often used to measure widths when using non-monospace fonts.

4.5 Browser plugins

Using browser plugins to improve the security and user experience was evaluated. No plugins were implemented in the scope of this thesis. This section details the potential advantages of plugins and the reasons not to implement them. The basic plugin features and advantages are as follows:

Certificate validation: the plugin would validate the server certificate fingerprint against a static whitelist. This effectively prevents all attacks that rely on user ignorance. This could be done also for other internal services.

Relogin prompts: warn the user before the authentication tokens expire, and prompt for re-login without redirecting the browser to the login service.

Additional encryption layer: encrypt the submitted data (for example, usernames and passwords) with a static public key. This limits the effects of compromised TLS. Also, this could hinder generic spyware that records the contents of the submitted forms. Furthermore, the frontend server could pass the encrypted data to a separate secure server for decryption and validation. This would prevent attacker with full access to the front-end server from obtaining any of the credentials.

Authentication for the browser: the plugin could authenticate the user browser with a private key. Simply eavesdropping the traffic would not enable the attacker to impersonate the browser. Cookies do allow this. In theory, standard client certificates could be used for this, but in practice, support in existing clients is inadequate.

Background data collection: web pages can only execute JavaScript code when the page is open on the browser. Plugins are running always when the browser is open. This could be used to collect for example location and accelerometer readings continuously to detect irregularities in user activity.

Even though the advantages are remarkable, the disadvantages are even greater. Two major disadvantages are the difficulties with the deployment and the amount of time required to implement and maintain the plugins.

For deployment, Google Apps provides a mechanism to deploy plugins to Chrome browsers. However, less than 50% of the users are using Chrome (see appendix B). Additionally, not everyone using Chrome has enabled synchronization with their Google Apps account. Furthermore, popular mobile

browsers do not currently support plugins. Moreover, some customer organizations do not allow installing non-whitelisted plugins. Thus, mandating plugin installation is not reasonable. As the plugin would not provide direct advantages for the user, the users have no clear incentive to install the plugin. Of course, the plugin would make the browser more trustworthy, thus reducing the number of required authentications.

To provide reasonable support, at least three separate plugins must be created: Chrome (46%), Safari (30%) and Firefox (18%). All popular browsers provide different APIs, thus requiring separate implementations for the majority of the functionality. Especially Firefox is famous for modifying its plugin APIs between releases, requiring constant updates for the plugins. Moreover, at the time of evaluation, the plugin API for Chrome is relatively restricted, making implementing the desired functionality hard or impossible. Due to the working culture and environment at Futurice, mandating a specific browser is not an option.

Based on these reasons – unreasonable amount of time required for implementation, constant need for maintenance, and difficulties with deployment – plugins were not implemented. In environments with a strictly controlled workstation setup, a browser plugin could and probably even should be implemented.

Chapter 5

Data analysis and evaluation

This chapter describes the data analysis and results of the different feature extraction steps. After describing an overall view on the authentication decisions, each feature is discussed separately. The next chapter presents and evaluates the overall results.

All the benchmarks are executed in a virtual machine running Linux 3.2.0 with a single 2.4GHz Intel Core 2 Duo processor and 1GB of memory. The benchmarks were implemented with Python 2.7.3, numpy 1.8.1 and scipy 0.13.3.

The data analysis has been separated to multiple different features. Decisions are formalized as a simple decision tree, illustrated in figure 5.1. This decision tree is used to update the state machine illustrated in figure 4.1.

For each feature, false negative rate (FNR) and false positive rate (FPR) are calculated. A false negative occurs when legitimate user must reauthenticate. In this case, in figure 5.1, the decision process leads to a state marked with "(1)". A false positive occurs when the feature does not reject illegitimate access, i.e., the decision process does not end to a state marked with "(1)". The whole system fails when all features produce false positive, i.e., the decision process for illegitimate access finishes to the state marked with "(2)".

Instead of this decision tree, algorithms such as Support Vector Machines (SVM) should be able to perform this task more efficiently. However, multiple issues exist in this approach. First, no labeled negative data – datasets of failed sign-in attempts that should have been rejected – exists. This hinders using any supervised algorithm. Second, being able to reason and adjust authentication decisions was deemed important. Extracting a specific reason – for example, whether additional authentication is required because device fingerprints do not match – from a SVM is not practical. Third, adjusting and training a SVM is a complicated task that is not easily performed by a

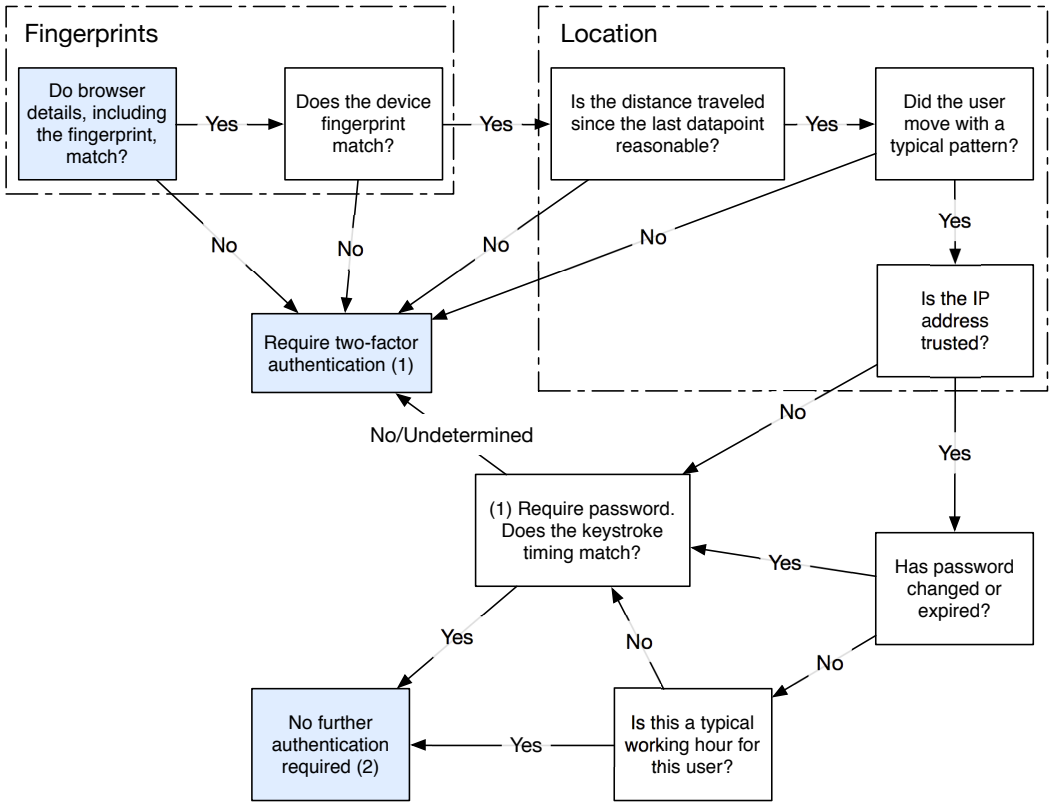

Figure 5.1: Authentication decision tree. Colored boxes depict the start and end states. Legitimate authentication attempts that go to a state marked with "(1)" are false negatives. Illegitimate attempts that finish to the state marked with "(2)" are false positives.

developer who is not familiar with the specifics of the system. The maintenance will be handled by developers who did not participate in developing the service. Fourth, adding or removing features should be easy. With SVM, this requires retraining and validating all results. Finally, validating the security implications of the decisions made by a SVM is rather difficult.

Furthermore, a simpler solution is easier to distribute to multiple servers, if needed. Based on these observations, the authentication decisions were not implemented with SVM or other robust machine learning algorithm. Implementing this should be investigated separately.

5.1 Browser authenticity

Browsers are identified with persistent cookies. However, stealing the cookies from backups, from intercepted traffic or with malware is entirely possible. To mitigate this, the browser and device authenticity is verified with several different mechanisms: user agent strings, JavaScript fingerprints and device fingerprints (section 5.4).

5.1.1 User-agent strings

Browsers transmit a user-agent string in the headers of every request[1]. User-agent strings are relatively stable, but neither unique nor secret identifier for the browser. The user-agent string changes on every software update, illustrated in the following examples (emphasis added):

*Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/**29.0.1547.76** Safari/537.36*

*Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/**30.0.1599.101** Safari/537.36*

Other browsers follow a similar pattern and provide similar information. We allow only the browser version (emphasized part in the above examples) to change, and only increasing the version is allowed. If any other information changes, or the version string decreases, the authenticity check fails.

During the year 2013, only three occurrences of downgraded browser versions were discovered, out of approximately 47000 logins by 286 users. In all three cases, the users had restored backups. Requiring reauthentication after the computer is restored from a backup is desirable, as it provides some protection against stolen backups.

The user-agent string comparison consumes virtually no resources: the memory requirements are restricted to the length of the user-agent strings and parsing the strings is a quick operation. The execution requires less than one millisecond.

5.1.2 JavaScript fingerprinting

As the name implies, in JavaScript fingerprinting, the browser executes JavaScript code that collects identifying information and sends it to the server. JavaScript fingerprinting was studied by Electronic Frontier Foundation (EFF) in their Panopticlick project [22]. The information JavaScript

[1] RFC 2616 specifies "User-Agent" as optional field [28]. In practice, all browsers send it.

collects could include a list of installed plugins, fonts, screen resolution, timezone and so on. In 2010, based on the data collected on the Panopticlick website, the fingerprint from a random browser carried at least 18.1 bits of identifying information [21]. Similar data collection was implemented for this thesis. The JavaScript snippet collects the timezone, screen resolution, and list of the plugins and file types which the browser accepts (mimetypes). The data is processed with the algorithm presented by Eckersley [21]. The entries are sorted, concatenated and compared with previous entries using a standard Python method, *difflib.SequenceMatcher* [78]. This string comparison algorithm outputs the ratio of identical blocks in the two sequences.

In this service, the problem is opposite from the EFF study. In general, people want to stay anonymous and minimize the identifiable features[2]. In contrast, with this authentication service, users do not want to be anonymous – after all, they want to authenticate themselves with minimal effort. The attacker wants to imitate another user as well as possible.

For tracking users for advertising, a large number of false positives is not an issue. However, in authentication, minimizing false positives – i.e., chances that an attacker successfully impersonates another user – is more important than minimizing false negatives.

Figure 5.2 shows the histogram for different similarity values. The results for different thresholds are introduced in table 5.1. For a stricter validation, the threshold value 0.96 produced good results. This typically allowed removing, installing or updating a single plugin. The values are calculated for all browsers that visited the service, including tablets and smartphones[3]. For this feature, two different thresholds were used: if the fingerprint similarity was below 0.85, two-factor authentication was required. If the similarity was below 0.96, at least a password was required. Values above 0.96 were accepted.

The threshold value 0.85 produced significantly higher FPR than what Eckersley achieved with general Internet population (3.3% vs. 0.86%). Our dataset is less diverse, as devices and operating system versions are more standardized. This means that the FPR should be remarkably lower when compared to a random device and browser. Of course, determined attacker tries to imitate the fingerprint as well as possible, thus increasing the chances of producing a false positive.

The worst case time requirement for the *SequenceMatcher* is $O(n^3)$, where n is the length of the strings. For matching or almost matching strings, the

[2]In practice, majority of the general population are not aware of this and the implications of reduced privacy.

[3]Eckersley found that tablet and smartphone browsers have few customization options, and therefore fingerprinting is not very effective.

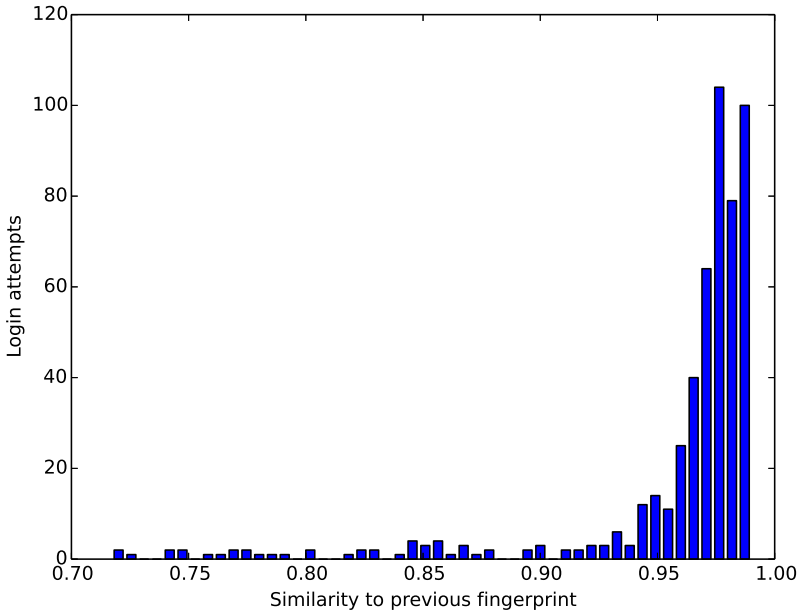

Figure 5.2: Browser fingerprint similarities for consecutive sign-in attempts, obtained with *SequenceMatcher*. Perfect matches ($n = 4067$) and values below 0.7 ($n = 8$) not shown for clarity. Values below 0.7 were for changing the device language. Majority of plugin description messages are localized, and thus the fingerprint changes greatly when locale is changed.

time requirement is considerably lower. In practice, with the real fingerprint sequences, the average execution time was approximately 9 milliseconds. The 95th percentile was almost 32ms. The longest legitimate browser fingerprint was approximately 10 000 characters. Restricting the maximum length of the fingerprint mitigates DoS attacks. The comparison does not require a significant amount of memory.

As the comparison time depends on the similarity of the strings, the attacker could use this information to guess the correct fingerprint. This attack vector could be eliminated by delaying the results until a constant (or almost constant) execution time is achieved. In practice, the attacker must eliminate all other latency sources, which should be either difficult or impossible. This should be investigated further.

Table 5.1: Acceptance thresholds for *SequenceMatcher*.

Similarity ratio threshold	FPR[1]	FNR[1]
0.85 [2]	3.3%	0.77%
0.92 [3]	1.1%	1.1%
0.96	0.13%	2.5%

[1] False positive occurs when two different browsers are not rejected. This was obtained by cross-checking all browser instances. False negative is recorded when consecutive access with the same browser is rejected, as the fingerprint changed too much.
[2] Used by Eckersley [21]. Their dataset resulted in 0.86% FPR and 0.56% FNR.
[3] Threshold for equal error rate.

5.2 Location

The location of the user attempting to access the services is pivotal to detecting illegitimate access. However, only the IP address of the user is available[4], and no accurate mechanism to obtain the physical location of the IP address exists. For example, with mobile networks, the physical location of the IP address is not necessarily constrained to any specific area.

As the IP address of the user changes relatively often (on average, 2.4 IP addresses per day per user), it is not practical to connect sessions to a single IP address. A closer inspection revealed a few reasons for the changing IP addresses. First, users change access networks at the office (for example, between LAN, VPN and Wi-Fi). Second, certain customers have multiple load-balancing gateways. There is no reliable way to detect this. Third, people actually move between locations and check in from home in the morning or in the evening.

Three different databases were evaluated for determining the physical location of the IP address: WHOIS, MaxMind's free GeoLite2 database [63] and MaxMind's commercial Precision service [62] (see section 2.4). The GeoLite2 database is less accurate, as MaxMind has deliberately added an unknown amount of noise to reduce the quality of the free product. Precision is the most accurate database MaxMind offers.

For evaluating the accuracy of the databases, a validation dataset was collected. For more information about the data collection methodology and the accuracy of the dataset, see appendix E.

[4]Except when the user is at the Futurice office.

For each datapoint on the validation dataset ($n = 397$), WHOIS, GeoLite2 and Precision data was obtained. For WHOIS data, a simple parse for extracting physical addresses from the data was implemented (for a single example of WHOIS data, see appendix G). After extracting physical addresses for IP networks, WGS84 coordinates were obtained from Google's reverse geocoding service [34]. For each datapoint, the error between the location geocoding database returned and the validation dataset location was calculated.

The errors do not seem to be normally distributed. Therefore, for evaluating the statistical significance of the accuracy difference between the geocoding databases, the Wilcoxon matched-pair signed-rank test [97] is used.

The Precision database reported correct country for 100% of the validation data entries. The WHOIS data did not improve the accuracy over results from Precision, except with small business offices. With those, the WHOIS data was accurate within 100 meters, while the accuracy of Precision data was limited to a correct city. This was observed only with 18 (4.5%) datapoints in the validation set. The difference is statistically significant ($z = 39.0$, $p = 0.0428$). This number may be higher for Futurice employees, as customer offices may have more accurate WHOIS records than the majority of the validation dataset entries. However, obtaining the WHOIS and reverse geocoding information is an expensive[5] and slow process. Therefore, the WHOIS data extraction was not employed in the actual implementation.

Finally, Precision and GeoLite2 were compared. The free GeoLite2 database does not provide accuracy information. For a fair comparison, accuracy information from the Precision database was not employed. GeoLite2 provided information for all the 397 entries while the Precision did not return results for 38 addresses (9.6%). However, the average error for the results which only GeoLite2 returned was 810km. This is clearly unusably high. Thus, only the 359 entries with data from both databases were compared.

The mean error for both databases is provided in table 5.2. Furthermore, figure 5.3 illustrates the distribution of the errors. GeoLite2 was more accurate for 22 addresses (5.5%), while Precision results were more accurate for 105 addresses (26.5%). The higher accuracy of the Precision database is statistically significant ($Z = 3080.0$, $p = 7.7 * 10^{-13}$). Based on these results, Precision was chosen for the geocoding process.

Obtaining the geocoding information consumes a considerable amount of time. The median response time for Precision was $166ms$. Of course, each IP address is fetched only once, and the response is cached. Furthermore, the information can be fetched immediately when the IP address is first seen.

[5]Quote for geocoding license from Google was over $10 000.

Table 5.2: Geocoding errors against the validation dataset

Data source	Mean error (km)
Precision	35.9
Precision & WHOIS	35.8
GeoLite2	102.0

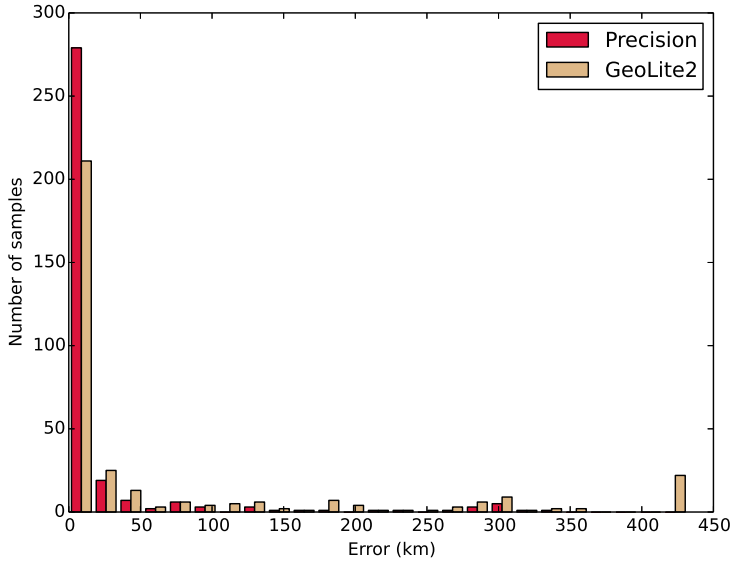

Figure 5.3: Histogram of Precision and GeoLite2 errors

If this is insufficient, the GeoLite2 offline database could be used to obtain coarse location virtually instantly (the median query time is approximately 1ms).

The Precision database reports the location accuracy (the radius of the circle where the IP address should be located). Figure 5.4 visualizes the location accuracy for entries where the reported accuracies were less than 100km. Three (0.75%) entries with error >100km are not shown for clarity. Based on this data, it was concluded that the Precision database is accurate enough for determining the user location. The accuracy information can be used to reliably ignore inaccurate entries.

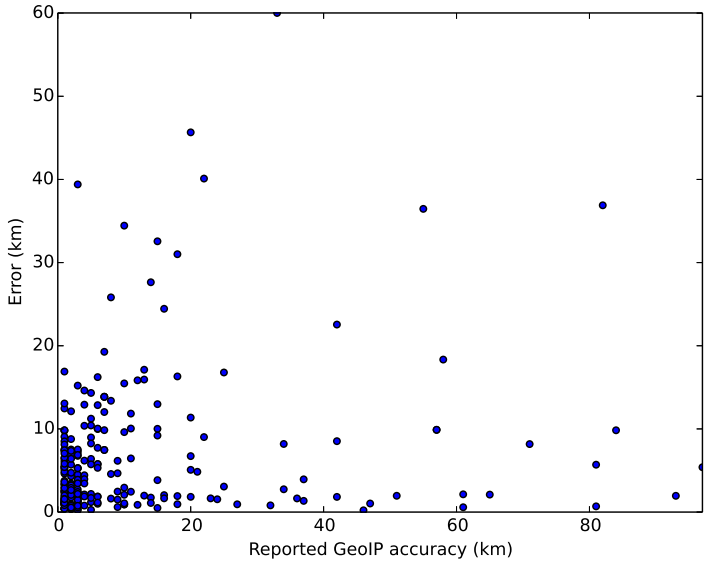

Figure 5.4: Correlation between the accuracy reported by the MaxMind Precision database and the validation dataset.

5.2.1 Anomaly detection

The goal of the anomaly detection is to detect unfeasible moving patterns in users' physical locations. For example, suddenly moving to another country is highly susceptible. This feature does not reject atypical patterns or unexpected locations, as long as the traveled distance is reasonable.

For evaluating anomaly detection for locations, a sample of previous user activity was obtained. The sample included entries over six weeks of user activity between 2014-03-01 and 2014-04-17. During this time, 272 users signed in from 2556 unique IP addresses. In total, 42278 transitions from one IP address to another were recorded.

The following heuristics were set for the traveling: if the distance traveled is below 1000 km, flying no faster than 800km/h (on average) is assumed. For distances in the range of 10–100 km, driving (or public transport) with an average speed no more than 60km/h is allowed. For the rest of the entries, 25km/h is the maximum movement speed. Figure 5.5 illustrates the number of entries with no data (discarded IPs) and number of false negatives for different location data accuracies.

Based on the results from the validation data and the results from the experiment, no accuracy threshold for the MaxMind data was specified.

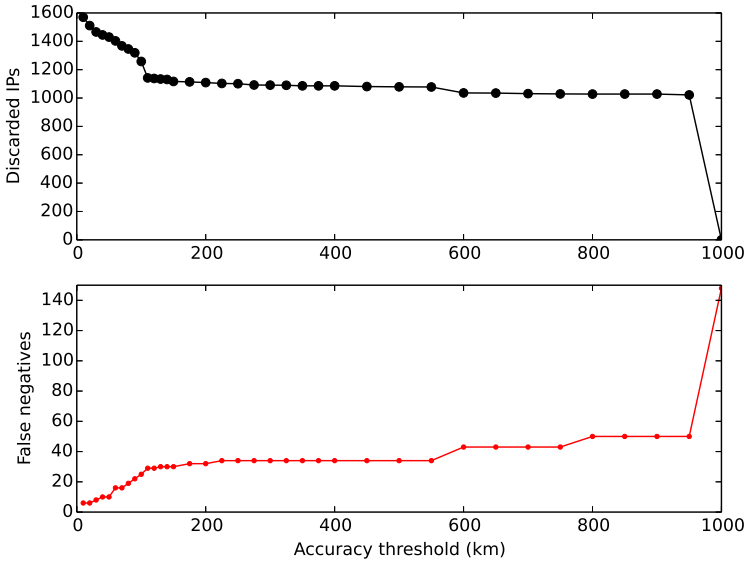

Figure 5.5: Travel simulations with different accuracy thresholds.

1000km seemed to be the minimum accuracy MaxMind provides. This resulted in 0.33% false negatives, i.e., cases where an anomaly is detected because of incorrect location data. The number of false positives is unknown. For a small number of IP addresses, there is no location data (0.16% in this sample). An attacker could potentially abuse this by using an IP address with no geolocation information. However, this is mitigated with typical location patterns (see section 5.2.2) and with IP reputation (see section 5.3).

As the anomaly detection only compares the distance to the previous datapoint, the whole process consumes less than a millisecond, assuming geolocation information for both IP addresses is available.

5.2.2 Typical location patterns

This features tracks the typical location patterns for each user. If the user travels to unexpected location, additional authentication is required.

A quick analysis indicated people typically leave their normal location to visit another country, and then return to the normal location. People rarely visit multiple foreign countries on the same trip. This is clearly a pattern that should be taken into account. Furthermore, during the first quarter of the 2013, approximately 75% of the users did not travel to any foreign

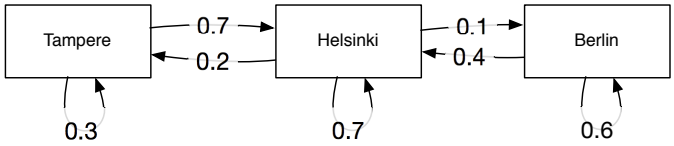

Figure 5.6: Demonstration of Markov chains

country[6]. 16% visited a single country and the rest (9%) were seen in more than two countries.

To model the user movement, the Markov chains were used. Figure 5.6 illustrates a simple Markov chain. In each state, there is a certain probability of either staying in the same state, or moving to one another state. Often, the probabilities in Markov chains depend only on the current state.

For each user, Markov chain was calculated for country-level locations. City-level modeling was tested, but it did not perform well enough. Based on historical data – the years 2009-2013 – a 6-months rolling window was used for training the Markov chains. Each day is counted only once, i.e., the Markov chain probabilities are adjusted only once per day per country. The threshold for accepting the transition was set to 5%. For example, if the user travels to a foreign country for a few days per month, it is considered a typical pattern. On the other hand, occasional travelers are challenged with two-factor prompts, which is the desired result. For the first quarter of 2014, this threshold caused approximately 20% of false negatives. The manual inspection indicated these are the cases where it is reasonable to ask for authentication: users traveling to a country they have not visited before. When returning to the originating country – a normal movement pattern – false negatives were 0%. For all user activity, the false negatives were at 0.3%, as the majority of the users' activities are within a single country.

The potential for false positives is high, but not easily measurable. However, as the IP address reputation system (see section 5.3) accepts a new IP address which another employee has used earlier, it is important to track the current country for each user. Furthermore, requiring two-factor authentication each time the user moves to another country is an insufficient solution, as a small but significant number of people travel regularly. The false positives could be reduced by detecting time-based patterns, for example, whether a specific user travels on specific dates of the month or week. However, the data did not directly indicate this to be the case.

[6]Or more specifically, the user did not sign in or access any tracked services from a foreign country.

The Markov chains are precalculated in the background. A complete calculation consumes several seconds, as fetching a large amount of data is a time-consuming operation. However, the models can be updated and only occasionally completely recalculated. Checking the current location is a quick operation (<1ms), as the server can keep all the necessary information in the memory at all times. The average model for a single user, with the associated data structures, consumes approximately one kilobyte of memory (or disk space).

5.3 Reputation of the IP address

This feature evaluates the legitimacy of the access based on the IP addresses. If the user tries to access the service from IP address which does not have adequate reputation, additional authentication is required. The reputation of the IP address does not consider the actual location, i.e., if two IP addresses with adequate reputation are located in different countries, simultaneous access from both addresses is allowed. This scenario is detected by anomaly detection (section 5.2.1) and location pattern analysis (section 5.2.2).

The reputation of each IP address declines over the time. That is, the older the previous legitimate access was, the less trusted the IP address is. Aborted sign-in attempts – either an invalid password or no response to the second-factor prompt – reduce the reputation, and activity by authenticated users increases the reputation. This greatly limits the number of addresses the attacker could use.

To determine the proper threshold, false negatives – i.e., cases where legitimate attempt was rejected – for different time thresholds were counted. Figure 5.7 illustrates the number of false negatives for different validity windows. The validity window is the maximum time since the previous legitimate access. The series marked with "3h" also consider the time of the day. For example, if the earlier access has been at 12:00, the IP address is only considered valid for 10:30-13:30. This limitation reduces the number of false positives – i.e., cases where illegitimate access is not rejected. For instance, if the user never works from the office during weekends, there is no reason to accept sign-in without reauthentication.

As the number of false negatives per user is relatively high even with longer expiration times, it is not practical to maintain only the per-user reputation. In practice, employees often visit the same locations. To reduce the number of false negatives, the windows of 2 days for global records – addresses other users have used – and 21 days for per-user records were selected.

This results in the false negative rate of 3.7%, while at least theoretically minimizing false positives. The number of false positives could be further reduced by automatically deciding per-user windows for both global and per-user reputation records. That is, at least for some users, lower windows may produce similar results.

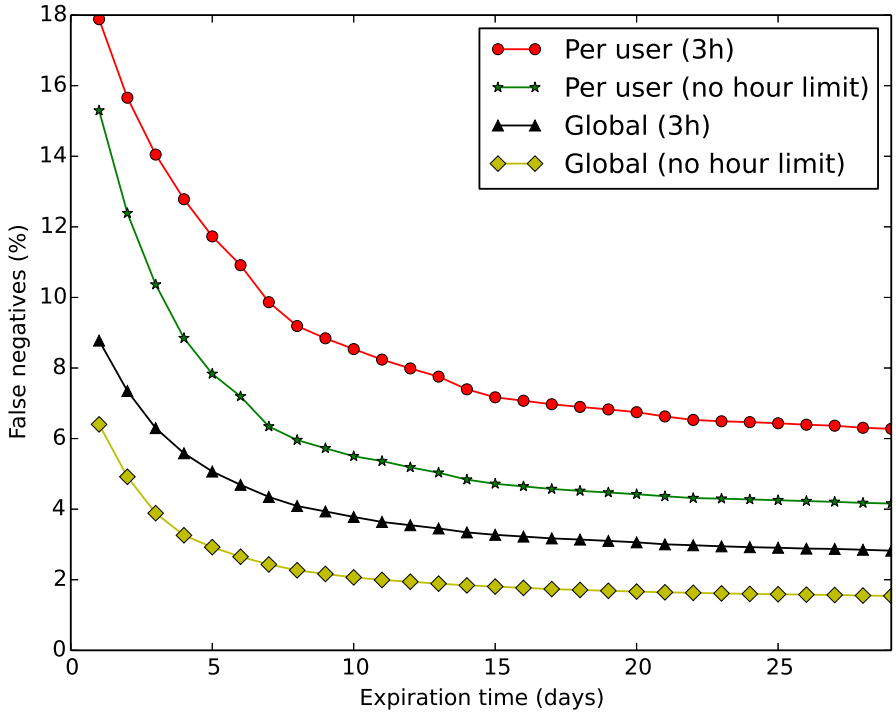

Figure 5.7: False negatives on IP reputation with different expiration times. In "Global", reputation is calculated without considering users or groups. "Per user" calculates IP reputation only for a single user. In the series marked with "(3h)", only entries within $\pm 1.5h$ from the current time of day are accepted.

The accuracy of the IP reputation was further improved with a few heuristics. When signing out, the IP was removed from the per-user database. Furthermore, if no other users accessed the service from the same IP address, it is removed from the global entries as well. If the brute-forcing ratelimits were exceeded (see section 4.3), the IP address was automatically removed from the reputation databases.

The reputation checks consume less than one millisecond of time. All the reputation information is stored in an in-memory database for a quick access. Information for one IP address consumes approximately 220 bytes of the memory. Thus, one megabyte holds over 4500 IP addresses. In practice, all the information older than 21 days can be discarded from the memory. For the first quarter of the 2014, at any time, approximately 22 000 reputation datapoints were stored. This required less than five megabytes of memory.

5.4 Device fingerprints

In addition to identifying the browser, assessing the authenticity of the device is useful. Even though browsers do not have a mechanism for obtaining device fingerprints[7], three device fingerprints were implemented:

Clock offset is measured with a simple JavaScript code. The functionality is presented in figure 5.8.

Device uptime is deduced from TCP timestamps [49]. The uptime indicates the time since the last reboot.

TCP flags and variables are relatively unique to each operating system and major version of the operating system.

5.4.1 Clock offsets

Clocks tend to drift linearly, unless synchronized[8] or set by the user. Based on this observation, two distinct types of devices were identified.

First, 9% of the devices seemed to synchronize the clocks either rarely or never. For these devices, the clock drifted linearly, and thus a linear model was fitted by minimizing the square sum of errors. The model is then used to predict the next offset. As measuring the uptime is inaccurate, our implementation allows $\pm 15\%$ error in the value. This resulted in 2% of false positives, i.e., cases where another device seen by the service was accepted.

Second, the rest of the devices seemed to synchronize their clocks regularly. For these devices, after enough data was collected, Gaussian distribution was assumed. The threshold for accepting the clock offset was set at the 99th percentile. That is, 1% of the past data was rejected as false positives.

[7]Flash, Silverlight and Java browser plugins expose some information, such as CPU speed and amount of memory, from the device and operating system, but executing any of these typically requires a separate approval from the user.

[8]Synchronizing the clock is typically handled with Network Time Protocol (NTP) [64]. Full NTP implementations also slowly adjust clock frequency to minimize drifting.

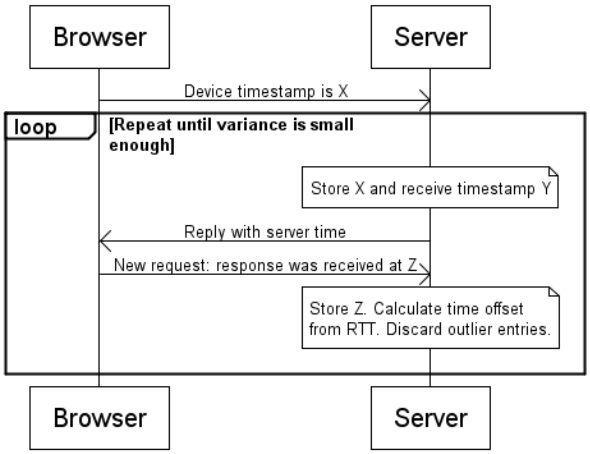

Figure 5.8: Overview of time offset calculation

5.4.2 Device uptime

As the device uptimes should increase linearly, again, linear regression was applied. As the uptime counters wrap around regularly, and devices are rebooted, linear regression uses only previous 30 entries for estimating the current value. We experimented with other values, but they produced less accurate results. With this model, 21% of the devices were considered as unpredictable (a typical example of unpredicatable device uptime data is demonstrated in figure 5.9). Unpredictable records are not explained by operating systems, network devices or remote hardware. The reason remains unclear.

For the rest of the devices, the optimal acceptance thresholds for each device were automatically determined with the following process:

1. Set threshold $r = 0.1$. Repeat steps 2-7 with $r = r + 0.1, r \geq 5$.
2. Set index $i = 0$. Repeat steps 3-7 with $i = i+1, i < entries.length-31$
3. Execute linear regression for $i..i + 30$ entries
4. Use fitted linear regression to predict $i + 31$
5. Calculate error percentage between the predicted and actual values
6. If the error is $< r$, entry is accepted. If the error is $\geq r$, entry is rejected.

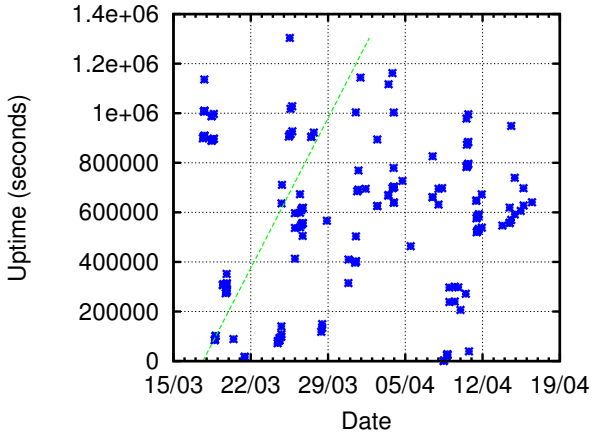

Figure 5.9: Example of irregular uptime records. The blue markers are recorded uptime guesses. The dashed green line illustrates the connection between scales of the two axis.

Consecutive rejected entries are considered as undetermined[9]. For each threshold, the number of accepted, rejected and undetermined entries is collected.

7. Calculate cost with $\frac{u*A+k*B}{a*C}$, where u is undetermined entries, k is rejected entries, a is accepted entries and A, B and C are weights.

Appropriate values for weights were obtained with cross-validating different values. For each device, the threshold that minimizes the cost function was selected. Weights $A = 1.3$, $B = 3.1$ and $C = 1.8$ produced FNR of 1.3% and FPR was at 0.7%.

5.4.3 TCP flags

The third and last device fingerprint feature is the TCP flags. All detected – identified from multiple different devices behind the same IP address – NAT and proxy servers were excluded, as NAT and VPN devices radically change the TCP flags. Simple operating system detection with the nmap operating system database [72] (more information available from [58]) provided FNR of 4.2% of the identifications, i.e., cases where operating system change was incorrectly detected.

[9]For example, after reboot, the model does not immediately adjust to the new uptime records.

5.4.4 Decision process

The device fingerprinting accepted the device with any of the following conditions:

- No undetermined features: two accepted features
- A single undetermined feature: a single accepted feature

Two or more undetermined features resulted in an undetermined device fingerprint. All other conditions caused the rejection of the device. This combination reduced false negatives to 0.3% and false positives to 0.4%. 17% were undetermined.

The linear regression time complexity is $O(n)$, where n is the number of entries. 30 entries required less than 0.2ms of processing time. In total, device fingerprint validation requires less than 3ms, including database overhead.

5.5 Working times

Working times are one defining character of user behavior. This features models the typical working hours of the user, and requests reauthentication if a deviation from normal pattern is detected.

Figure 5.10 details entries for one participant, the author of this thesis. Even the raw, unprocessed and non-filtered data clearly exhibits a few patterns:

- The user is probably not working on Wednesdays ("Logins" and "Doors")
- The computer is left running overnight and during weekends ("VPN and backup")
- Heavy amount of activity on evenings and weekends ("SSH and web activity")[10]
- Workday usually starts between 09 and 10.

Automatically extracting patterns like these from the data should be possible. However, not everyone is working with the same schedule. Figure 5.11 visualizes three common cases of user behavior. First, approximately 30% of the users mainly work regular hours, with only a small amount of activity outside of typical pattern. Second, 40% of the users work irregular hours,

[10]Due to writing this thesis.

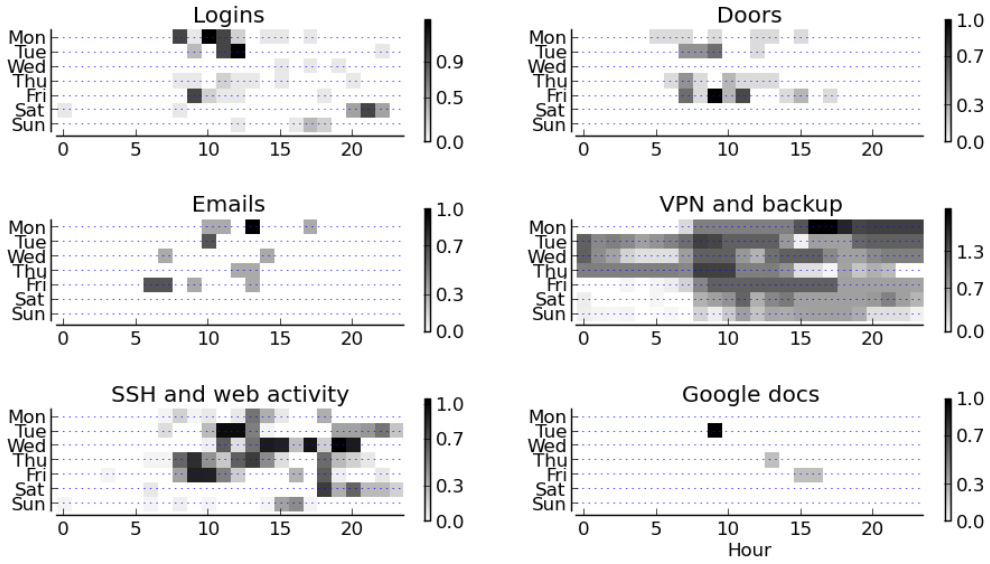

Figure 5.10: Working times entries from different sources for the author. All times in UTC. Darker color represents higher activity. Entries between 2013-10-10 and 2013-11-10.

with the main activity concentrated on relatively regular times. Third, a small number of employees work irregular hours. Finally, for some of users, virtually no data exists. For example, customers often sign in very irregularly, or do not use any service that collects activity data.

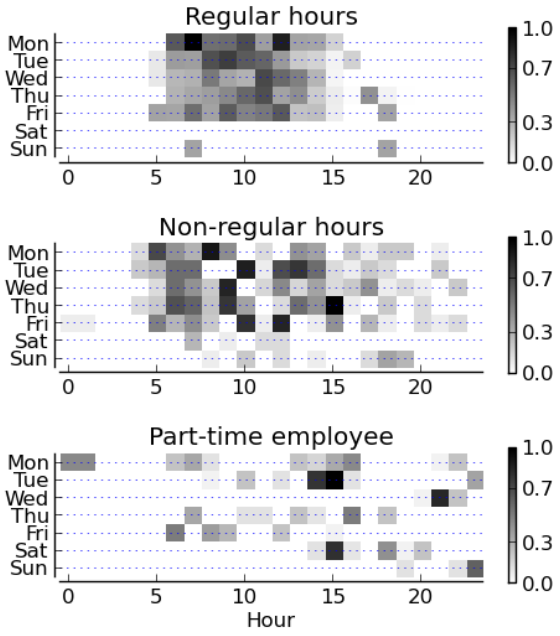

Figure 5.11: Examples of recorded working times patterns. All times in UTC. Normalized scale. Entries between 2013-10-10 and 2013-11-10.

5.5.1 git commits

Information extracted from the version control system provides important data, as it is the main system many developers use. Only support for git [31] was implemented. However, the same principles hold for many common version control systems.

The data is divided into two parts. First, git pushes are server-side records for user connections to the server. This includes push, pull and clone requests. Second, git commits are client-side records for changesets, subsequently pushed to the server. With git, the common usage pattern is to make multiple small local commits and push a larger amount of changes to the central server at once. This pattern produces a large number of client-side generated commits and a single push record on the server side at the end of the activity.

With git pushes, the information is recorded by the server and is therefore relatively trustworthy. Users are signing in using dedicated SSH public keys. The data was filtered as follows. First, a series with time differences (in seconds) between actions per key was created. Then, fast Fourier trans-

formation (FFT) was used to find regularities in the series. After discussions and closer look into the audit logs, it was concluded that the SSH public keys with clear patterns were used by various automated systems, namely deployment and continuous integration servers (CI). The keys are connected to a specific user, even when it is used by an automated system. Furthermore, some of the public keys were simultaneously used by the CI server and user.

The data was filtered to avoid storing incorrect activity entries by automated systems. Filtering overly aggressively was deemed preferable over storing incorrect data. After experimenting, the following actions were found to perform well for filtering automated entries:

1. Sort all entries into buckets based on the time difference (in seconds) to the previous entry.
2. If the number of buckets divided by number of entries is smaller than 0.05, all entries for the key is discarded. This automatically discards keys that are mainly used for regularly executed tasks.
3. Calculate the average time difference between the connections (in seconds).
4. Discard all entries in buckets with difference $\leq avg * 0.8$.

0.1% of the automated entries remained after filtering. This was deemed satisfactory accuracy. Approximately 4% of legitimate entries were removed.

git commits are created by the client. Thus, there is a large number of inconsistencies, including incorrectly set clocks, improperly configured timezones, and non-standard author names. Of course, fabricating local commit data is straightforward. On the other hand, pushing commits to the server is authenticated separately. Therefore, a potential attacker should not have a straightforward method to create or alter the records. Additionally, the server does not allow modification of commits already pushed to the server.

For filtering git commits, a cursory approach was taken. First, all entries with no valid email address or username as an author were dropped. Second, entries with no commit message or with frequently occurring commit message were discarded. Finally, entries with frequently occurring {username, minutes of commit timestamp} or {username, seconds of commit timestamp} pairs were removed, as automated commits seemed to run automatically at specific times, but not necessarily regularly. For example, an automatic commit might run 15 minutes past full hour, whenever there were changes to the files.

After filtering, both the commits and push data were included to the decision database.

5.5.2 Modeling and results

For modeling the working times, a validation dataset was collected. The dataset consists of self-reported working hours for one week from 30 randomly selected employees. For detailed information about the methodology, see appendix F.

Initial attempts to fit the data to the working times recorded in the validation dataset did not give good enough results. There is a large discrepancy between user actions and what was reported as working times. For example, almost everyone had at least some work-related activity after what they considered as the end of their working day. Discouraging work-related actions after the workday might be desirable. However, that is not a task the authentication system should perform in this case. Therefore, instead of modeling the working times, only user activity was tracked. For users with no earlier data – new employees and those returning from long holidays – user data from other employees from the same location is used.

For each user, activity from each data source was normalized. For each data source, the number of entries per day is counted. Then, the weight for the entry is calculated by dividing by the number of entries. When user activity is encountered, the sum of the weights for entries within N minutes are counted. That is, for example for $-30 \leq N \leq 30$ and timestamp 08:05, all weights between 07:35-08:35 are counted. Furthermore, only the entries from the same day of the week are considered. Intuitively, in addition to typical working times, this takes typical working days into account. This approach does not function perfectly for more complex working patterns – for example, working every second Monday – but such patterns were rare. Finally, only the entries from the past C days were considered. The probability obtained from counting the weights in proximity of the current timestamp is finally compared with the threshold t.

If the user is challenged with the authentication prompt – the activity is occurring outside of the typical working times – the working times feature is undetermined for a certain amount of time. Therefore, if the user accesses a service during the weekend, the authentication prompt is shown only once. Of course, other features may trigger an authentication prompt later on. The data indicated this period of undetermination should be four hours. In practice, activities outside typical working hours are often one-off operations, or responding to urgent service calls.

Values $c = 63$ (approximately two months), $-41 < N < 94$ and $t = 0.28$ provided reasonable results. This resulted in 1.3% of false negatives, i.e., cases where the legitimate user had to reauthenticate. The number of false positives – i.e., cases where the feature does not reject illegitimate access – is

difficult to measure. However, intuitively – and based on manual inspection – the times when the activity is allowed is greatly reduced. For example, if all activity ends at 17:00, the probability for valid working time swiftly drops below the threshold t (within 20 minutes). Similarly, access is accepted approximately 15 minutes before the start of the typical working day. Based on these values, a coarse estimate for false positives would be 2.4%. The results could be further improved by adjusting the thresholds separately for each user.

The working times validation requires approximately 2ms of processing time if the per-user models have been fully precalculated. Calculating the model for one user consumes several seconds, as fetching a large amount of data is a relatively slow operation. The memory requirements are modest – approximately 20kB/user.

5.6 Keystroke timing

Keystroke timing models the timing patterns users exhibit when entering their passwords. This is identifying feature [7, 37, 50, 57], even though with enough data and proper system, impersonation is possible [92].

We implemented keystroke timing using Dynamic time warping (DTW), which introduced in the section. After that, our implementation is described.

5.6.1 Dynamic time warping

Dynamic time warping (DTW) is an algorithm for comparing the similarity of two time series. Intuitively, DTW tries to fit two time series together with non-linear transformation. Warping time series together has many applications in speech, video and graphics recognition. Warping minimizes the cost function, which is usually either Euclidian distance or square-sum of distances. [67]

This process is illustrated in figure 5.12.

A basic, non-optimized DTW is relatively simple:

```
function dtw(a: [0-n], b: [0-m]):
  -> initialize d as array with dimensions n and m
  -> initialize all values of d with infinity
  d[0][0] = 0
  for i = 0...n:
    for j = 0...m:
      d[i][j] = distance(a[i], b[j]) + minimum(
          d[i-1][j],
```

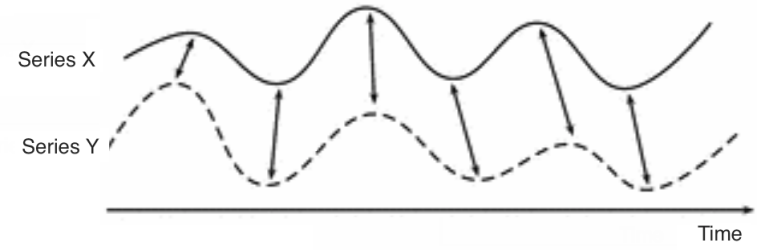

Figure 5.12: Illustration of DTW. Image from [67]. Arrows indicate aligned points from the series.

```
9                d[i][j-1],
10               d[i-1][j-1])
11   return d[n][m] # return the last value
```

distance returns the distance between the datapoints a and b. This could be for example Euclidian distance, i.e., $(a[x]-b[x])^2+(a[y]-b[y])^2$. *minimum* returns the smallest of the arguments. The algorithm calculates the distance between every pair of the input arrays. For simplicity, this non-optimized version calculates each distance twice. The actual implementations avoid this.

5.6.2 Analysis and results

If the user uses a password manager to enter the password, no meaningful keystroke timing data is available. However, using a password manager is a weak indicator too: if the user has almost always used a password manager – which applies to approximately 10% of the users – manually entering the password is clearly suspicious. On the other hand, imitating password entry by the password manager is straightforward.

The keystroke timing data consists of a millisecond-resolution series of timestamps for each keypress. The classification is relatively simple: the training data consists of previous sign-in attempts by the same user, and the input is the time series for current login. Obviously, only attempts for the same user entering the same password are considered.

The data extraction and analysis were implemented using DTW (see previous section). The algorithm runs pairwise DTW for the new record against all known good records. These records are sign-ins that were not interrupted, i.e., the authentication was finished, and no reason to doubt the legitimacy exists.

Unfortunately, the type of the device greatly affects the typing speed and patterns. For instance, writing the password on the mobile device differs greatly from full-sized keyboards. It seems that the type of the physical keyboard does not matter [92].

Experimenting with a relatively large dataset (ten entries for the same password from 100 users, [56]) confirmed that the cost function should be a squared sum of distances, as that minimized the false positives. With a sample dataset, after filtering erroneous records, the equal error rate (EER) 6.2% was achieved while verifying with leave-one-out cross-validation. EER is the point where FPR and FNR are equal. Of course, the cost of relatively low EER was a large number – approximately one third – of users discarded due to inconsistent writing patterns.

As keystroke timing only provides additional validation, a relatively high error rate (6.2% of false positives) is not an issue. Therefore, no attempt to resort to more sophisticated algorithms was made.

The process requires $O(n^2)$ time and memory, where n is the number of entries. Fortunately, passwords tend to be short [29]. For 100-character strings, a single execution of the DTW consumes less than 2ms. Pairwise calculation against ten previous entries is approximately 14 milliseconds in total[11].

[11]However, in practice, multiple calculations can run simultaneously on multicore processors.

Chapter 6

Results and evaluation

The overall performance of the decision process was measured by replaying earlier user activities and applying the decision process described in figure 5.1 to them. The replayed activity includes all the data the authentication service can use during normal operations. This includes sign-in attempts, authentication token requests and activity data from other services.

The performance was evaluated with three metrics: security (minimizing false positives, i.e., the likelihood that an attacker with a stolen session cookies would not be asked to reauthenticate), effectiveness (minimizing false negatives, i.e. the number of authentication prompts legitimate users encounter) and usability (minimizing the amount of time spent on the authentication process). Also, the response times – i.e., the time users spent waiting for pageloads – are presented. For qualitative analysis, a short user survey was performed two months after the deployment of the new authentication service. For more information, see appendix D.

A summary of the accuracy for the different features is available in table 6.1. The contents of the table are further evaluated in the following sections.

Table 6.1: Summary of the accuracy for user activity features. Undet. is the percentage of cases where the feature is undetermined, i.e., no decision was made. Performance is the 95th percentile of the processing time that the feature required.

Feature	False negatives	False positives	Undet.	Performance
Browser fingerprints	0.3%	2.4%	0%	10ms
Traveling anomalies	0.05%	?	42%	<1ms
IP reputation	3.7%	?	0%	<1ms
Device fingerprint	0.3%	0.4%	17%	3ms
Working times	1.3%	4.8%[1]	0%	2ms
Location pattern	0.3%	?	0%	<1ms
Total (naive)[2]	6.07%	0.03%[3]		18ms
Total (real)[4]	0.7%	?[5]		18ms

[1] Estimated.

[2] Based on assuming independent features, which is not the case. False negatives are accumulated, as all the features must pass. False positives are multiplied, as a single rejected feature rejects the authentication attempt.

[3] Includes undetermined entries.

[4] Based on replaying the earlier user activity, which represents the real performance of the system.

[5] As the features are not independent, the number of potential false positives differs from the theoretical one.

6.1 Security goal

The security goal is to minimize false positives, i.e., cases where the system accepted illegitimate access without requiring additional authentication. It is important to note that the attacker must first obtain cookies from a legitimate user.

Rejection by any feature described in table 6.1 results in an additional authentication prompt. Thus, the combined false positive rate is $\prod_{i=0}^{i \leq n} fp_i$, where fp_i is percentage of false positives for each feature. In theory, this is 0.03%, including undetermined entries – in effect, undetermined features are false positives.

However, the features are not independent. For example, the location patterns and IP reputation are often correlated: when the user travels to a new country, they usually use a new IP address as well. Furthermore, the number of false positives is unknown for several of the features. Even if false positives would be 100% for a certain undetermined feature, it would not degrade the security measurement. Of course, features with 100% of false

positives and more than 0% of false negatives have negative impact to the usability goals, and should be discarded. Some guidelines for the number of false positives can be presented:

IP reputation If the attacker is at the same location with a legitimate user – for example, at an Internet cafe – IP reputation will not affect the attacker. However, it still restricts the access to a specific time interval. Assuming the attacker attempts to access the service from a random IP address within the countries in which Futurice operates – Switzerland, Germany, Great Britain and Finland – in theory, the false positives would be 0.0002%[1]. Of course, for a dedicated attacker, this is not the case.

Location pattern Every IP address in the country user is currently located in is a potential false positive for this feature. However, at the same time, for a typical user, this excludes 95% or more of the IP addresses[2]. Of course, this only affects attackers that are abroad. Furthermore, it is not possible to reliably detect VPNs and proxies.

Traveling anomalies are connected to the IP reputation and location patterns. The attacker could potentially choose an IP address with no geocoding information. For the Precision database, the number of such IP addresses is not known. For GeoLite2, there are 4.4 million such addresses (0.13%).

When cross-checking all the entries, not a single false positive was detected. We evaluated this by executing the whole decision process for every authentication attempt by other users. The obvious failing check – session cookies – was not considered in this. In each time, the system required additional authentication, as at least one of the features rejected the authentication attempt. This result does not necessarily reflect the real-world performance, as no deliberate attempts to impersonate others were made.

6.2 Usability goals

The results of the usability goals are summarized in table 6.2. This comparison is biased against the new system, as Google Apps authentication was

[1]According to GeoLite2 database, there is approximately 280M IP addresses assigned to these countries. During the first quarter of 2014, at any time, no more than 500 IP addresses had high enough reputation.

[2]According to MaxMind GeoLite2 database, there is 120M IP addresses assigned to Germany. This is 3.4% of all available IP addresses.

Table 6.2: Summary of usability goals. Effectiveness is measured as number of interruptions per day, on average. Usability is defined as the time spent on the authentication process per day per user, on average.

Measurement	Old service	New service	Difference
Effectiveness	0.660	0.067	-89.8%
Usability	10.7s [1]	2.5s	-76.6%

[1] Due to inadequate data collection, no meaningful time measurements are available for 7.2% of the sign-in attempts.

migrated to the same SSO service. This means that more people than before use the authentication service in different situations.

Neither effectiveness nor usability measurements include time spent on signing in to workstations and to other services that are not covered by the single sign-on service.

With the old system, users performed authentication once per 1.5 days, on average. With the new system, entering the password – i.e., first-factor authentication – was performed once per 12.9 days. One-time passwords – i.e., second-factor authentication – was requested once per 16.9 days.

The measured time spent on the authentication process only includes the time user spent on the authentication service. We measure this because it is the only automatically measurable metric. What really matters is the user satisfaction with the two-factor authentication, and the total time users lose due to interruptions. Automatically measuring the length of the interruption – i.e., how long it takes before the user continue their earlier activities as if no interruption occurred – is difficult or impossible. In the survey (see appendix D), 96% of the respondents said they are happy with the new service. No comparison data for the old service exists.

With both services, the first-factor authentication consumed approximately the same amount of time (16.2 seconds). The second-factor authentication with the SMS required $23s$ (standard deviation $12.8s$) and with the Authenticator, $18s$ ($\pm 9.8s$) was needed. Authentication with the Authenticator requires less time. The difference is statistically significant (assuming normally distributed data, T-test, $t = 4.038$, $p = 0.000061$, $df = 588.7$). Time spent on the second-factor authentication did not change significantly on subsequent sign-in attempts. This is expected, as over 50% of the employees were already familiar with the Authenticator, and virtually everyone had used SMS authentication with other internal tools. A full two-factor authentication requires approximately 40 seconds, 150% increase over the old

system.

The average time spent on configuring the second-factor authentication was 120.1s (standard deviation 73.7s). However, this should not be taken into consideration when calculating the efficiency of the new system, as the configuration is a one-off operation and its costs are distributed over a long period of time. This is not an accurate measurement for multiple reasons. First, some users tried to configure the Authenticator app multiple times. Not all of these attempts were combined properly in the timing data, i.e., a small part of the consecutive configuration attempts were counted as separate sessions. Second, the time spent on reading the instructions from other sites and installing the application to the mobile device was not measured. Third, a few users had problems that required help from the IT team. This was not measured separately.

Table 6.1 summarizes the effectiveness – the number of false negatives, i.e., unnecessary authentication challenges legitimate users encounter – of the each feature. The significant difference between naive and real false negatives (6.07% and 0.7%, respectively) is due to dependencies between the features. For example, often when the IP reputation failed, the location patterns and working times failed as well.

The number of sessions requiring more than one sign-in attempt in the first-factor authentication was 7% for old service and 7.5% for the new service. Any small variance might be explained by confusion between regular passwords and one-time passwords. Additionally, not requiring the password daily may interrupt the routine and lead to more mistakes on entering the password[3]. For comparison, with SSH, 9.8% of the connections required more than one attempt before the correct credentials were provided[4].

During the second-factor authentication, 6.4% of the sessions required more than one attempt before the correct one-time code was entered. With Authenticator, in 6.9% of the attempts the correct code was not provided in the first attempt. Any variance in the error rate is probably explained by the confusion with multiple Authenticator codes[5].

[3]Anecdotal evidence supports this. No robust data about this has been collected.

[4]These figures only include sessions where the correct credentials were provided. This excludes potential brute-force attacks from skewing the results.

[5]Some users configured the Authenticator for Google Apps, which creates entry with label "firstname.lastname@futurice.com". Even though authentication service prompted for a code with different label, multiple users told they first tried to use the incorrect code.

6.3 Response times

As the decision process adds overhead to the authentication, measuring the response times is important. Nielsen specifies that for response times below 100ms, users feel that the system is reacting instantaneously [71, chapter 5]. For 1000ms, users notice the delay, but it does not break the workflow. The observed response time can be divided to three separate components: the time the server consumes on processing the request and response, network overhead (including DNS resolution and TCP overhead) and rendering the page on the browser (including parsing the HTML, and executing JavaScript code). We measured the network latency and rendering times experienced by the users with the navigation timing API [95].

Assuming that the navigation timing data which the browsers provide is accurate, the 100ms threshold is not achievable. The median for the whole pageload – from the start of the request to having the rendered page visible on the browser – was 637ms. The overhead from the decision process is approximately 3% of the total (20ms). The time spent on each part of the pageload is detailed in table 6.3. For the majority of the requests, network overhead consumed approximately 60% of the time, and rendering required 30%. Approximately 75% of the requests were processed and rendered to the user in less than 1000ms.

Enabling Perfect Forward Secrecy (see section 2.3.1) increased network overhead by 16%. This is considerable but necessary expense. Furthermore, newer, more responsive page layouts and icons increased the page rendering times by 60%. Whether better visual appearance[6] and browser compatibility are worth the increased page rendering times is not clear. The impact of this could be limited by automatically rendering a low-resource version for slower devices.

[6]This is obviously very subjective measurement. However, newer versions removed apparent visual errors.

Table 6.3: Summary of response times. All values in milliseconds.

	Network	Server	Rendering[1]	Total
Average	473 (59.9%)	81 (10.3%)	256 (29.8%)	790
10th percentile	8 (6.5%)	31 (25%)	85 (68.5%)	124 [2]
Median	385 (60.3%)	57 (8.9%)	197 (30.8%)	639
90th percentile	1077 (61.1%)	155 (8.8%)	531 (30.1%)	1763 [3]

[1] On some devices, rendering time includes loading resources, such as images, style sheets and fonts.

[2] Workstation connected to Futurice's Helsinki office LAN typically achieves this performance, as the latency to the authentication server is below 1ms.

[3] This is a realistic value for mobile devices. Even though the page is relatively lightweight, page rendering is slow in low-end devices.

Chapter 7

Discussion

Evaluating the security of the whole system proved to be difficult. In a different environment, this could be handled in a more robust way. For instance, data from earlier intrusion attempts and succeeded intrusions could be used to verify the effectiveness of the authentication decisions. Some of this data could be collected by an intrusion detection system (IDS). For this case study, no such data is available. In addition to attack data, commercial or internal penetration testing could provide some level of confidence. This was not studied within this thesis. Finally, code reviews and formal security reviews could be used to validate conformance with specifications, but do not necessarily prove real world performance.

As an alternative approach, in this thesis, each model was evaluated separately, as the evaluation of a single feature – for example, predicting employees' working times – can be executed reliably. As the logic for combining the metrics is straightforward, deducing the effectiveness of the system from the separate evaluations is plausible.

One open question is the potential for side-channel attacks. For example, the time required for browser fingerprint comparison greatly depends on the similarity of the fingerprints. Theoretically, this information could be used for deducing the correct fingerprint. If each feature is executed sequentially, the total time the server requires for the decision process may leak additional information. Furthermore, the recently used data is often retrieved from the cache. Again, this may leak information to the attacker. This could be mitigated for example by adding delays to implement constant-time processing. Investigating the practicality of using timing as a side-channel was left for further research.

The operating environment restricts the data collection. Web browsers provide only a restricted interface to some of the environmental sensors the device may have. Furthermore, the sensors can be accessed only when the

web page is open. Compared to native applications, web browsers are fairly open environments, as users can easily inspect and modify the JavaScript code. Also, development tools integrated into the browsers allow manually setting environmental sensor data. With native applications, forging the data is usually significantly harder, but obviously possible.

In addition to cameras and microphones, which were excluded for privacy reasons, accelerometer is one interesting data source that was not implemented in this thesis, due to time constraints[1]. If the device has an accelerometer, the data should be available with a simple JavaScript API [9]. Current browsers do not require user approval before allowing access to the data. This is different from the location, camera and microphone data. All modern smartphones, tablets and Apple laptops, among other devices, have an accelerometer. At least with mobile phones, it is possible to identify users when they pick up the phone [14]. Furthermore, imperfections in the sensors reliably identify the devices [18]. Even if identification is not possible, the data could be used to deduce some facts: for example, whether the device is on the table or in a vehicle. This data might provide additional information that helps to identify the user, device and environment.

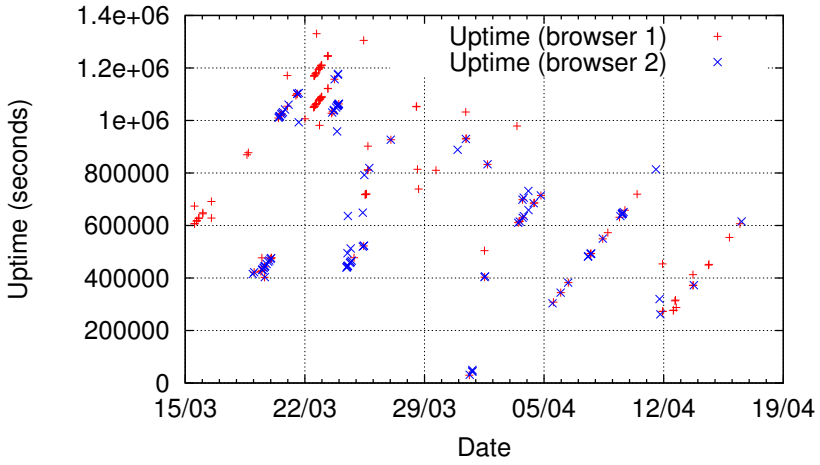

Figure 7.1: Uptime records for two browsers running on the same computer

Some users, especially developers often use multiple browsers, or multiple browser profiles. Cookies or other session identifiers are not shared between these. The client device uptime records could be used to identify browsers

[1]Additionally, reading the accelerometer data caused serious performance degradation on every tested device. The reason for this is unclear.

running on the same computer. Furthermore, uptime records could be used to detect cookies shared between multiple devices by detecting anomalies in the observed uptime records over a longer period of time. This information increases or reduces the trustworthiness of other browsers, i.e., when the user successfully signs in on one browser, in some cases there is no need to require full two-factor authentication on other browsers that are running on the same computer. See figure 7.1 for an example of a relatively clear dataset from two browsers running on the same computer. The preliminary inspection of the data suggested this should be further investigated. Multiple browsers were correctly detected to belong to a single computer in 11 instances using only the uptime data. All these cases were manually validated. The accuracy of the analysis could be improved with clock offset data and operating system fingerprints.

Two-factor authentication introduces potential denial of service: if the user loses their mobile phone and cannot access the emergency codes, it is not possible to sign in without the intervention of the IT department. This could be avoided in various ways. For example, other employees can help a user to bypass the second-factor prompt. The process could go as follows: when the user realizes that they cannot access the second-factor codes, they need to meet with more than one other employee simultaneously. The user enters her username and password, selects one-time recovery, and two – or more, depending on the situation – others enter their passwords and second-factor codes to sign the user in. Requiring more than one full authentications is crucial to prevent trivial escalations. If the user must enter their password first, two malicious employees cannot impersonate her by abusing the recovery method. Requiring at least two other employees for recovering the access is not unrealistic, as employees often move as a team.

Another possible approach is to allow bypassing the second-factor authentication by entering enough personal information. The service could automatically ask details such as emails sent, email labels, projects with committed code, calendar events, last sign-ins, last holidays, ID badge and key serial numbers, device serial numbers etc. As long as the user provides diverse enough answers – for example, answers that can not be found from a stolen mobile phone – this information could be used to override the two-factor prompt. The difficult problem is to find enough data sources that are secret but which user can still remember. For example, all calendars are automatically shared between all employees and thus are not sufficiently secret.

In a larger environment, the data processing and performance could be improved by predicting which users are most probably signing in within a short timeframe, and pre-calculating the models for those users. This, and

caching in general, could lead to side-channel attacks. For example, by measuring response times with different data, the attacker could deduce which users have their data cached. Furthermore, the response times could leak information about the amount of the data.

As a large amount of activity data from various services is collected, at least some anomaly detection for usage patterns should be possible. This was left for further research. The anomaly detection could provide invaluable information for detecting attackers who have gained access to computer with open session. As a trivial example, some people always open the intranet frontpage before opening any other service. When deviation from a typical pattern is detected, the system could challenge the user with an additional authentication prompt.

One interesting arrangement would be to have a very restrictive firewall, and a single sign-on server accessible from the Internet. After a successful login, the firewall rules for the remote IP address are automatically added, to allow accessing other services. However, opening the firewall from an unknown IP address poses security problems too [16]. On the positive side, even if the IP address is shared between multiple devices and users, the arrangement greatly restricts the IP addresses that could access the services. This was not implemented, as various customer networks route traffic to different services through separate gateways. In practice, the connection to the authentication service may be routed through a different gateway than the traffic to other services. In this case, the user would not be able to access the services even after signing in successfully.

Chapter 8

Conclusions

This thesis detailed the background, design, implementation and evaluation of an intelligent two-factor authentication service. The service was implemented at Futurice, a medium-sized software consulting company.

Traditional web authentication systems either prompt for the username and password at regular intervals (e.g., once per day), or offer a choice to save the browser and avoid entering the credentials in the future. At Futurice, the old service naively requested the username and password once every 12 hours, and thus interrupting users almost every day.

Furthermore, the password alone does not provide a satisfactory level of security. For example, if the attacker eavesdrops the password, she can sign in from anywhere. Traditionally, companies have restricted the access to their internal services from the Internet, requiring either connection from the office LAN or over VPN. This provides a second-factor component to the authentication process. At Futurice, restricting access to LAN or VPN is not practical due to a large number of employees working from customer premises.

To improve the security, two-factor authentication with one-time passwords (OTPs) was implemented in this thesis. OTP effectively eliminates the risks related to the eavesdropped credentials. Unfortunately, two-factor authentication also increases the time required for the authentication process. Entering the username and password consumes approximately 16 seconds on average (including multiple sign-in attempts, when necessary). The same metric for the OTP is approximately 25 seconds. When combined, this is a 150% increase to the time required for the authentication process.

Fortunately, the data collected from various services confirmed that users follow relatively strict routines. The intelligent authentication service implemented in this thesis challenges the user only when the earlier routines are not followed. To avoid compromising security, multiple independent data

sources were used. Browsers – and thus, users – are primarily identified with cookies, which is the standard method in the web. If the attacker is able to obtain the cookies, she must also impersonate the device and browser properly. Furthermore, the attacker must be in the correct location at the correct time.

Our authentication service reduced the number of interruptions users encountered by 89.8%. Furthermore, even though two-factor authentication consumes more time, our system reduced the required time by 76.6%. On average, users spent approximately two minutes on configuring the two-factor authentication. With our service, user saves this amount of time in less than a month. The assumptions and data analysis were validated with various surveys and validation datasets, which are described in appendices B–F.

The survey executed approximately two months after we deployed the new service indicated that a large majority of the users were satisfied with the new service (96%, $n = 100$). 70% thought that the new service is more secure than the old one[1], and the rest (30%) had no opinion. 20% indicated the credentials were required too often[2]. For detailed results and methodology, see appendix D.

Deploying two-factor authentication with one-time passwords is not necessarily a large inconvenience. At least technically inclined people understand the need for such a change with rather minimal communication.

One interesting question is whether the more complicated and less predictable sign-in process causes users to abandon tasks they were about to perform. For example, if performing some action is optional or just interesting, having to enter the OTP from the mobile phone may discourage the user from continuing.

Another open question is how the authentication service detailed in this thesis changed the user behavior. On the post-deployment survey (appendix D), 20% said that they started using two-factor authentication on other services as well, and 12% claimed they check in for work-related tasks on their free time less often. Further, 12% said they use random computers less often. These factors could be found from the data. Furthermore, user behavior could have changed in unexpected ways. This was left for further research.

With a sufficient amount of labeled data, the decision process could incorporate machine learning, as opposed to using heuristics. Optimally, the system could process arbitrary data and automatically deduce important portions, patterns and combinations. In theory, with enough training, neu-

[1]However, users' perception of security does not necessarily reflect the facts. Often, more difficult systems are perceived as more secure. See for example [94].

[2]This was affected by a bug that caused unnecessary credential prompts for 10% of the users.

ral networks could do this. In practice, it requires an impractical amount of data. However, more intelligent systems would learn the more complex and hidden user behaviors. As a far-fetched example, weather and seasons probably affect working times and behavior.

The same service could be deployed to other environments. Some parts of the data collection are highly specialized (for example, electronic door access data collection) and thus require customization. Similarly, thresholds used in this thesis may not apply to other environments. Tuning the decision process could be automated at least partly. This too was left for further research.

Bibliography

[1] Adams, A., and Sasse, M. A. Users are not the enemy. *Communications of ACM 42*, 12 (Dec. 1999), 40–46.

[2] Adelsbach, A., Gajek, S., and Schwenk, J. Visual spoofing of SSL protected web sites and effective countermeasures. In *Information Security Practice and Experience*. Springer, 2005, pp. 204–216.

[3] Amazon Web Services. AWS multi-factor authentication. Product page available at `http://aws.amazon.com/mfa/`. Referenced at 2013-10-26.

[4] Apple. iOS: Understanding Control Center. Official support article for iOS control center. Available at `http://support.apple.com/kb/HT5858`. Referenced at 2014-04-13.

[5] Authy. One-time password service. Product website available at `https://www.authy.com/`. Referenced at 2013-12-09.

[6] Barth, A. HTTP State Management Mechanism. RFC 6265 (Proposed Standard), Apr. 2011.

[7] Bergadano, F., Gunetti, D., and Picardi, C. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security 5*, 4 (Nov. 2002), 367–397.

[8] Bernat, V. SSL/TLS & Perfect Forward Secrecy. Blog post available at `http://vincent.bernat.im/en/blog/2011-ssl-perfect-forward-secrecy.html`. Referenced at 2014-03-09.

[9] Block, S., and Popescu, A. DeviceOrientation Event Specification. W3C documentation for DeviceOrientation JavaScript API. Available at `http://www.w3.org/TR/orientation-event/`. Referenced at 2014-04-14.

[10] Brooks, R., and Deng, J. Lies and the lying liars that tell them: A fair and balanced look at tls. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research* (New York, NY, USA, 2010), CSIIRW '10, ACM, pp. 59:1–59:3.

[11] Bumett, M. 10,000 top passwords, 2011. Blog post available at `https://xato.net/passwords/more-top-worst-passwords/`. Referenced at 2013-10-29.

[12] Cantor, S., Kemp, J., Philpott, R., and Maler, E. Assertions and protocols for the OASIS security assertion markup language (SAML) v2.0, 2005. SAML v2.0 standard, 15 March 2005. Available at `http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`. Referenced at 2013-12-10.

[13] Clark, J., and van Oorschot, P. SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. In *2013 IEEE Symposium on Security and Privacy (SP)* (May 2013), pp. 511–525.

[14] Conti, M., Zachia-Zlatea, I., and Crispo, B. Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security* (New York, NY, USA, 2011), ASIACCS '11, ACM, pp. 249–259.

[15] Daigle, L. WHOIS Protocol Specification. RFC 3912 (Draft Standard), Sept. 2004.

[16] deGraaf, R., Aycock, J., and Jacobson, M. Improved port knocking with strong authentication. In *Computer Security Applications Conference, 21st Annual* (Dec. 2005).

[17] Deveria, A. Can I use... HSTS. Available at `http://caniuse.com/#feat=stricttransportsecurity`. Referenced at 2013-10-15.

[18] Dey, S., Roy, N., Xu, W., Choudhury, R. R., and Nelakuditi, S. AccelPrint: Imperfections of accelerometers make smartphones trackable. In *Proceedings of the 20th Annual Network and Distributed System Security Symposium (Feb 2014), NDSS* (2014), vol. 14.

[19] Diffie, W., Van Oorschot, P. C., and Wiener, M. J. Authentication and authenticated key exchanges. *Designs, Codes and cryptography 2*, 2 (1992), 107–125.

[20] Two-Factor Authentication Made Easy - Duo Security. Two-factor authentication service provider. Website available at `https://www.duosecurity.com/`. Referenced at 2013-12-09.

[21] Eckersley, P. How unique is your web browser? In *Privacy Enhancing Technologies*, M. Atallah and N. Hopper, Eds., vol. 6205 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010, pp. 1–18.

[22] EFF. Panopticlick - how unique - and trackable - is your browser? Research project website, available at `https://panopticlick.eff.org/`. Referenced at 2014-04-11.

[23] European Commission. Progress on EU data protection reform now irreversible following European Parliament vote. European Commission MEMO/14/186, released at 2014-03-12.

[24] European Parliament and the Council of the European Union. Proposal for a regulation of the European parliament and of the council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (general data protection regulation). Version COM(2012) 11 final. Available at `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2012:0011:FIN:EN:PDF`. Referenced at 2014-04-10.

[25] European Parliament and the Council of the European Union. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Union L 281* (1995), 0031–0050.

[26] Facebook login. Login service documentation available at `https://developers.facebook.com/docs/concepts/login/` . Referenced at 2013-11-20.

[27] Fette, I., and Melnikov, A. The WebSocket Protocol. RFC 6455 (Proposed Standard), Dec. 2011.

[28] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999.

[29] Florencio, D., and Herley, C. A large-scale study of web password habits. In *Proceedings of the 16th International Conference on*

World Wide Web (New York, NY, USA, 2007), WWW '07, ACM, pp. 657–666.

[30] FUTURICE. Futuricen 2013 culture audit -dokumentti. Culture audit of Futurice for Great Place to Work survey. Available at `http://blog.futurice.com/wp-content/uploads/2013/10/Futurice_CultureAudit_GPtW2013.pdf`. Referenced at 2013-11-01.

[31] GIT. Git. Official home page of distributed version control system Git. Available at `http://git-scm.com/`. Referenced at 2014-03-15.

[32] GOOGLE. About 2-step verification. Help page available at `https://support.google.com/accounts/answer/180744?hl=en`. Referenced at 2013-10-18.

[33] GOOGLE. google-authenticator. Google Authenticator code repository. Available at `https://code.google.com/p/google-authenticator/`. Referenced at 2013-10-10.

[34] GOOGLE. The Google Geocoding API. API documentation for geocoding API. Available at `https://developers.google.com/maps/documentation/geocoding/`. Referenced at 2014-04-01.

[35] GOOGLE. Google Drive SDK: API Reference, 2013. Documentation available at `https://developers.google.com/drive/v2/`. Referenced at 2014-03-20.

[36] GOOGLE. Reports API: Docs activity - access event names, 2013. Documentation available at `https://developers.google.com/admin-sdk/reports/`. Referenced at 2014-03-20.

[37] HAIDER, S., ABBAS, A., AND ZAIDI, A. K. A multi-technique approach for user identification through keystroke dynamics. In *IEEE International Conference on Systems, Man, and Cybernetics, 2000* (2000), vol. 2, IEEE, pp. 1336–1341.

[38] HARDT, D. The OAuth 2.0 Authorization Framework. RFC 6749 (Proposed Standard), Oct. 2012.

[39] HAYASHI, E., RIVA, O., STRAUSS, K., BRUSH, A. J. B., AND SCHECHTER, S. Goldilocks and the two mobile devices: going beyond all-or-nothing access to a device's applications. In *Proceedings of the Eighth Symposium on Usable Privacy and Security* (New York, NY, USA, 2012), SOUPS '12, ACM, pp. 2:1–2:11.

[40] Henkilötietolaki (personal data act) 523/1999, 1999. Unofficial English translation available at `http://www.finlex.fi/fi/laki/kaannokset/1999/en19990523.pdf`.

[41] Hickson, I. Web storage, 2013. Documentation available at `http://www.w3.org/TR/2013/REC-webstorage-20130730/`. Referenced at 2013-10-27.

[42] Hodges, J., Jackson, C., and Barth, A. HTTP Strict Transport Security (HSTS). RFC 6797 (Proposed Standard), Nov. 2012.

[43] ICANN. Whois data reminder policy. Policy available at `http://www.icann.org/en/resources/registrars/consensus-policies/wdrp`. Referenced at 2014-04-06.

[44] IEEE. IEEE standard for local and metropolitan area networks - port-based network access control. *IEEE Std 802.1X-2010 (Revision of IEEE Std 802.1X-2004)* (2010), C1–205.

[45] ISACA. ISACA glossary of terms, 2012. Information Systems Audit and Control Association (ISACA). Available at `http://www.isaca.org/Knowledge-Center/Documents/Glossary/glossary.pdf`. Referenced at 2014-01-20.

[46] ISO. ISO/IEC 27000:2014 information technology - security techniques - information security management systems - overview and vocabulary. Tech. rep., ISO, 2014.

[47] Israel, S. A., Irvine, J. M., Cheng, A., Wiederhold, M. D., and Wiederhold, B. K. ECG to identify individuals. *Pattern recognition 38*, 1 (2005), 133–142.

[48] Ives, B., Walsh, K. R., and Schneider, H. The domino effect of password reuse. *Communications of the ACM 47*, 4 (Apr. 2004), 75–78.

[49] Jacobson, V., Braden, R., and Borman, D. TCP Extensions for High Performance. RFC 1323 (Proposed Standard), May 1992.

[50] Jiang, C.-H., Shieh, S., and Liu, J.-C. Keystroke statistical learning model for web authentication. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security* (New York, NY, USA, 2007), ASIACCS '07, ACM, pp. 359–361.

[51] Kasper, M. mod_auth_pubtkt: a pragmatic web single sign-on (SSO) solution, Jun. 2012. Software website at `https://neon1.net/mod_auth_pubtkt/index.html` . Referenced at 2014-03-08.

[52] Knecht, T. Frequent update request. Withdrawn policy proposal for RIPE. Available at `http://www.ripe.net/ripe/policies/proposals/2010-09`. Referenced at 2014-04-03.

[53] Laki yhteistoiminnasta (act on co-operation within undertakings) 334/2007, 2004. Unofficial English translation available at `http://www.finlex.fi/en/laki/kaannokset/2007/en20070334.pdf`.

[54] Laki yksityisyyden suojasta työelämässä (act on the protection of privacy in working life) 759/2004, 2004. Unofficial English translation available at `http://www.finlex.fi/en/laki/kaannokset/2004/en20040759.pdf`.

[55] Lienhart, R., and Maydt, J. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on* (2002), vol. 1, IEEE, pp. I–900.

[56] Loy, C. C., Lai, W. K., and Lim, C. P. Keystroke100 dataset. Dataset and description available at `http://www.eecs.qmul.ac.uk/~ccloy/downloads_keystroke100.html`. Referenced at 2013-10-25.

[57] Loy, C. C., Lai, W. K., and Lim, C. P. Keystroke patterns classification using the ARTMAP-FD neural network. In *Intelligent Information Hiding and Multimedia Signal Processing, 2007. IIHMSP 2007. Third International Conference on* (Nov 2007), vol. 1, IEEE, pp. 61–64.

[58] Lyon, G. F. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning.* Insecure, USA, 2009.

[59] Marlinspike, M. More tricks for defeating SSL in practice. *Black Hat USA* (2009). Video available at `http://www.youtube.com/watch?v=ibF36Yyeehw`. Slides available at `http://www.blackhat.com/presentations/bh-usa-09/MARLINSPIKE/BHUSA09-Marlinspike-DefeatSSL-SLIDES.pdf`. Referenced at 2013-10-21.

[60] Marlinspike, M. Moxie Marlinspike - software - sslstrip, 2011. Website of sslstrip software available at `http://www.thoughtcrime.org/software/sslstrip/`. Referenced at 2014-03-30.

[61] MaxMind. GeoIP2 databases and web services. Product page for geocoding databases and service. Available at `http://www.maxmind.com/en/geolocation_landing`. Referenced at 2014-04-11.

[62] MaxMind. GeoIP2 Omni web service. Product page for MaxMind's GeoIP2 Omni web service. Available at `http://www.maxmind.com/en/web_services_omni`. Referenced at 2014-04-05.

[63] MaxMind developer site. GeoLite2 free downloadable databases. Product page for free geoIP databases. Available at `http://dev.maxmind.com/geoip/geoip2/geolite2/`. Referenced at 2014-04-05.

[64] Mills, D., Martin, J., Burbank, J., and Kasch, W. Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905 (Proposed Standard), June 2010.

[65] M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., and Ranen, O. HOTP: An HMAC-Based One-Time Password Algorithm. RFC 4226 (Informational), Dec. 2005.

[66] M'Raihi, D., Machani, S., Pei, M., and Rydell, J. TOTP: Time-Based One-Time Password Algorithm. RFC 6238 (Informational), May 2011.

[67] Müller, M. Dynamic time warping. *Information Retrieval for Music and Motion* (2007), 69–84.

[68] Narten, T., Huston, G., and Roberts, L. IPv6 Address Assignment to End Sites. RFC 6177 (Best Current Practice), Mar. 2011.

[69] Needleman, M. The shibboleth authentication/authorization system. *Serials Review 30*, 3 (2004), 252–253.

[70] Neuman, C., Yu, T., Hartman, S., and Raeburn, K. The Kerberos Network Authentication Service (V5). RFC 4120 (Proposed Standard), July 2005.

[71] Nielsen, J. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[72] Nmap. Nmap OS fingerprint 2nd generation DB. Operating system detection database. Available at `https://svn.nmap.org/nmap/nmap-os-db`. Referenced at 2014-04-24.

[73] O'Leary, J. Getting started with login verification, May 2013. Blog post available at `https://blog.twitter.com/2013/getting-started-with-login-verification`. Referenced at 2013-10-18.

[74] OpenSSL. OpenSSL Security Advisory [07 Apr 2014] - TLS heartbeat read overrun (CVE-2014-0160). Security advisory for Heartbleed vulnerability. Available at `https://www.openssl.org/news/secadv_20140407.txt`. Referenced at 2014-04-09.

[75] Pashalidis, A., and Mitchell, C. Impostor: a single sign-on system for use from untrusted devices. In *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE* (Nov.-3 Dec. 2004), vol. 4, pp. 2191 – 2195 Vol.4.

[76] Pashalidis, A., and Mitchell, C. Using EMV cards for single sign-on. In *Public Key Infrastructure*, S. Katsikas, S. Gritzalis, and J. Lopez, Eds., vol. 3093 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 205–217.

[77] Popescu, A. Geolocation API specification. Available at `http://www.w3.org/TR/2013/REC-geolocation-API-20131024/`. Referenced at 2013-11-01.

[78] Ratcliff, J. W., and Metzener, D. E. Pattern-matching - the gestalt approach. *Dr. Dobbś Journal 13*, 7 (1988), 46.

[79] Recordon, D., and Reed, D. OpenID 2.0: a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management* (New York, NY, USA, 2006), DIM '06, ACM, pp. 11–16.

[80] Redis - key-value store. Software website available at `http://redis.io/`. Referenced at 2013-12-09.

[81] Redis command reference. Available at `http://redis.io/commands`. Referenced at 2014-04-01.

[82] Rescorla, E. HTTP Over TLS. RFC 2818 (Informational), May 2000.

[83] Riva, O., Qin, C., Strauss, K., and Lymberopoulos, D. Progressive authentication: deciding when to authenticate on mobile phones. In *21st USENIX Security Symposium* (August 2012).

[84] Samar, V. Single sign-on using cookies for web applications. In *IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 1999. (WET ICE '99) Proceedings* (1999), pp. 158–163.

[85] Schneier, B. Two-factor authentication: too little, too late. *Communications of ACM 48*, 4 (Apr. 2005), 136.

[86] RSA SecurID - Two-Factor Authentication, Security Token - EMC. Product website available at `http://www.emc.com/security/rsa-securid.htm`. Referenced at 2014-04-11.

[87] Sermersheim, J. Lightweight Directory Access Protocol (LDAP): The Protocol. RFC 4511 (Proposed Standard), June 2006.

[88] Silveira, V. Protecting your LinkedIn account with two-step verification, May 2013. Blog post available at `http://blog.linkedin.com/2013/05/31/protecting-your-linkedin-account-with-two-step-verification/`. Referenced at 2013-10-18.

[89] Song, A. Introducing login approvals, May 2011. Press release available at `https://www.facebook.com/note.php?note_id=10150172618258920`. Referenced at 2014-01-10.

[90] Song, D. X., Wagner, D., and Tian, X. Timing analysis of keystrokes and timing attacks on SSH. In *10th USENIX Security Symposium* (2001), vol. 2, p. 3.

[91] Statistics Finland. Official statistics of Finland: Education, 2012. ISSN 1796-0479. Available through StatFin service, `http://www.stat.fi/tup/statfin/index_en.html`. Referenced at 2013-12-12.

[92] Tey, C. M., Gupta, P., and GAO, D. I can be you: Questioning the use of keystroke dynamics as biometrics. The 20th Annual Network & Distributed System Security Symposium (NDSS 2013).

[93] Turk, M., and Pentland, A. Eigenfaces for recognition. *Journal of cognitive neuroscience 3*, 1 (1991), 71–86.

[94] Uzun, E., Karvonen, K., and Asokan, N. Usability analysis of secure pairing methods. In *Financial Cryptography and Data Security*. Springer, 2007, pp. 307–324.

[95] Wang, Z. Navigation timing, Dec. 2012. Reference available at `http://www.w3.org/TR/navigation-timing/`. Referenced at 2013-10-07.

[96] Whitten, A., and Tygar, J. Usability of security: A case study. Tech. rep., DTIC Document, Dec. 1998. CMU-CS-98-155 / ADA361032.

[97] Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics 1*, 6 (1945), 80–83.

[98] Williams, J., and Wichers, D. OWASP top 10 – 2013. *OWASP Foundation, June* (2013). Available at `http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf`. Referenced at 2013-10-18.

[99] Wu, S.-T., and Chieu, B.-C. A user friendly remote authentication scheme with smart cards. *Computers & Security 22*, 6 (2003), 547 – 550.

[100] Zhou, Y., and Evans, D. Why aren't HTTP-only cookies more widely deployed. *Proceedings of 4th Web 2.0 Security and Privacy Workshop 2* (2010). Available at `http://w2spconf.com/2010/papers/p25.pdf`. Referenced at 2013-10-18.

Appendix A

CSP policy

```
1 Content-Security-Policy: \
2         default-src https://login.futurice.com; \
3         frame-src 'none'; object-src 'none'; \
4         font-src 'none'; connect-src 'none'; \
5         media-src 'none'; \
6         report-uri https://login.futurice.com/csp_report
```

"\" represents non-breaking lines. The actual HTTP header does not contain line breaks.

default-src https://login.futurice.com : default-src tag controls the default setting, including all other content-types. More specific options override this. CSP also supports 'self', but manually defining the *https* and the hostname further restricts general man-in-the-middle attacks.

frame-src 'none' : prevents loading HTML frame elements from any host.

object-src 'none' : prevents loading HTML <object> content from any host. This includes Flash.

font-src 'none' : disables external fonts. External fonts are potential security issue, both for exploiting font handling code, and for deceiving the user by providing font file with invalid/altered characters.

connect-src 'none' : *connect-src* controls where the browser is allowed to connect, including XHR (AJAX), WebSockets and EventSource.

media-src 'none' : *media-src* is for video and audio content.

report-uri : Information about CSP policy violations are automatically posted to *report-uri*.

Appendix B

Browser versions

All sources referenced at 2013-10-22. Browser support for HSTS and CSP retrieved from `caniuse.com`.

Table B.1: Visits by server version (2013-09-01 – 2013-09-30)

Browser version	Visits	% of visits	CSP	HSTS	Release date
Chrome 29.0	1390	41.7%	✓	✓	August, 2013
Safari 6.0	1010	30.3%	✓		July, 2012
Firefox 23.0	487	14.6%	✓	✓	August, 2013
Chrome 28.0	148	4.4%	✓	✓	July, 2013
Firefox 24.0	106	3.2%	✓	✓	September, 2013
Safari 7.0	43	1.3%	✓		Beta release
Chrome 30.0	36	1.1%	✓	✓	October, 2013 [1,2]
Internet Explorer 9.0	30	0.9%			March, 2011
Chrome 31.0	17	0.5%	✓	✓	Beta release [2]
Internet Explorer 10.0	12	0.4%			August, 2012
Others	55	1.6%			-

[1] `http://googlechromereleases.blogspot.fi/2013/10/stable-channel-update.html`

[2] Users are using beta channel, e.g unreleased versions.

Table B.2: Visits by server version (2013-04-01 – 2013-09-30)

Browser version	Visits	% of visits	CSP	HSTS	Release date
Safari 6.0	6955	40.9%	✓		July, 2012
Chrome 26.0	2199	12.9%	✓	✓	April, 2013 [1]
Chrome 27.0	2132	12.6%	✓	✓	May, 2013 [2]
Chrome 28.0	1929	11.4%	✓	✓	July, 2013 [3]
Chrome 29.0	1802	10.6%	✓	✓	August, 2013 [4]
Firefox 23.0	840	4.9%	✓	✓	August, 2013 [5]
Chrome 25.0	266	1.6%	✓	✓	February, 2013 [6]
Internet Explorer 9.0	158	0.9%			March, 2011
Firefox 24.0	121	0.7%	✓	✓	September, 2013 [7]
Safari 7.0	119	0.7%	✓		Beta release
Others	465	2.7%	?	?	-

[1] `http://googlechromereleases.blogspot.fi/2013/04/stable-channel-update.html`

[2] `http://googlechromereleases.blogspot.fi/2013/05/stable-channel-release.html`

[3] `http://googlechromereleases.blogspot.fi/2013/07/stable-channel-update.html`

[4] `http://googlechromereleases.blogspot.fi/2013/08/stable-channel-update.html`

[5] `https://www.mozilla.org/en-US/firefox/23.0/releasenotes/`

[6] `http://googlechromereleases.blogspot.fi/2013/02/stable-channel-update_21.html`

[7] `https://www.mozilla.org/en-US/firefox/24.0/releasenotes/`

Appendix C

Pre-deployment survey

Time of the study: this survey was executed before implementing or deploying the new authentication service. Answers were collected between 2013-11-29 and 2013-12-09.

Purpose of the survey: The goal was to understand opinions and knowledge related to two-factor authentication. Furthermore, understanding how well users manage their passwords was important.

Procedure: the survey was implemented as a web-based survey. Participants answered the survey on their own time with their own device. The survey was anonymous. There is no mechanism to detect or remove duplicate answers, or answers by people who are not employees.

Participants: the link to the web survey was sent with email to all employees ($n = 197$)[1] on 2013-11-29. Additionally, the link was distributed in various internal instant messaging chats. Approximately 50% ($n = 106$) of the population answered the survey.

Confidence: 106 participants represent the whole population with error of $\pm 7\%$ (95% confidence level). As participation was voluntarily, this may introduce bias to the sample. For instance, people with no opinion may pass participating.

[1] This is the number of unique email addresses on the internal mailing lists at the time of sending the survey. It is not exact representation of the number of employees at that time.

C.1 Questions

The survey had two pages. Footnotes were not included in the survey. No detailed background information – age, sex, length of the employment, role at the company – were collected.

1 Are you using two-factor authentication? *... excluding online banks.* Yes/No *

2 Do you regularly use your work laptop on your free time? Yes/no[2]

3 Do you allow others use your work laptop with your account? If you only allow using a separate guest account, answer no. Yes/On exceptional cases/Never

4 Are you using your Futurice password on any external services or device? *Futurice Google Apps does not count*[3]*. For example, do you use the same password for external services or on the device at home?* Yes/No

5 When you have to change your password at Futurice, do you create a completely new password? *Answer no, if you only make a minimal change to your previous password.* Yes/No

If user answered Yes to first question (*), next page showed following questions:

A.1 Are you using two-factor authentication voluntarily? Yes/No

A.2 Is it easy to use? Yes/No

A.3 Is it required too often? Yes/No

For No, second page showed following questions:

B.1 Why not? I don't know what is two-factor authentication / It's too difficult / I don't think it makes a difference / I don't have anything to protect / It is not available

The last question on the second page for both choices was

A&B What about using it at Futurice? *Assuming you only need to enter it when using a new device, or on similar exceptional cases.* Yes, for everyone / Yes, as opt-in feature / No

[2]Company policy explicitly allows using work laptop on free time
[3]Password is automatically synced from LDAP to Google Apps

C.2 Results

106 users submitted answers to the survey.

People might have answered "No" to question **6**, as no "Indifferent" or similar neutral option was available. When this was realized, additional option was not added, as significant number of answers were already collected.

For the first 31 answers, question **5** did not specify "at Futurice". However, as majority of the services do not mandate password changes, misunderstanding the question should not skew the results. Indeed, distribution of answers is equal before and after modifying the question.

Table C.1: Survey answers: general questions

Question	Yes	No	Remarks
Are you using two-factor authentication?	56%	44%	
Do you regularly use your work laptop on our free time?	53%	47%	
Do you allow others use your work laptop with your account?	2%	68%	30% answered "On exceptional cases"
Are you using your Futurice password on any external service or device?	15%	85%	
When you have to change your password at Futurice, do you create a completely new password?	57%	43%	

Table C.2: Survey answers: users using two-factor authentication

Question	Yes	No	Remarks
Are you using two-factor authentication voluntarily?	83%	17%	
Is it easy to use?	87%	13%	
Is it required too often?	15%	85%	
What about using it at Futurice?	96%	4%	33% answered "Yes, as opt-in feature"

Table C.3: Survey answers: users not using two-factor authentication

Question	Yes	No	Remarks
What about using it at Futurice?	85%	15%	56% answered "Yes, as opt-in feature"

Table C.4: Survey answers: Why you don't use two-factor authentication?

Answer	Percentage
I don't know what is two-factor authentication	9%
It's too difficult	35%
I don't think it makes a difference	11%
I don't have anything to protect	7%
It is not available	27%
Other	11%

Appendix D

Post-deployment survey

Time of the study: this survey was executed approximately two months after deploying the new authentication service.

Purpose of the survey: The goal was to collect feedback about the deployment process and to measure user satisfaction.

Procedure: the survey was implemented as a web-based survey. Participants answered the survey on their own time with their own device, with no supervision. The survey was anonymous. There is no mechanism to detect or remove duplicate answers, or answers by people who are not in fact employees.

Participants: the link to the web survey was sent with email to all employees ($n = 218$)[1]. Additionally, the link was distributed in various internal instant messaging chats. Approximately 50% ($n = 100$) of the population answered the survey.

Confidence: 100 participants represent the whole population with error of $\pm 7.5\%$ (95% confidence level). As participation was voluntarily, this may introduce bias to the sample. For instance, people with no opinion may pass participating.

Questions and answers are detailed in table D.1. All questions were optional. However, only a single person did not answer every question. Thus questions 4 and 5 have 99 answers.

[1]This is the number of unique email addresses on the internal mailing lists at the time of sending the survey. It is not exact representation of the number of employees at that time.

Table D.1: Survey questions and answers

Question		Answers
1. Are you happy with the change?	Yes	68
	No	4
	Indifferent	26
	What change?	2
2. During a typical week, how much time you spend on the login service?	Less than before, which is good	44
	More than before, which is bad	21
	About the same	12
	Not sure	23
3. What is your opinion about the security of the new login service?	It is better than before	70
	Worse than before	0
	I have no opinion about this	30
4. With the new system, how often you need to enter password/one-time password?	Too often	20
	I'm happy with it	70
	Too rarely	2
	I have no opinion about this	7
5. Did you understand why the change was necessary?	Yes	74
	No	16
	Yes, after asking for better explanation	9
6. Did you change your behaviour with the new system? [1]	I don't think anything changed.	58
	I took two-factor authentication into use in other services too.	21
	I use random computers for work-related things less often.	12
	I check emails/other work related things less often on my free time.	13
	I started using password manager.	3
	Other[2]	8

[1] "Choose all that apply"

[2] Freetext field. No noteworthy answers were collected.

Appendix E

Geocoding validation dataset

Time of the study: the dataset was collected between 2013-03-10 and 2013-04-30.

Purpose of the survey: the dataset was collected to evaluate the accuracy of different geocoding databases.

Procedure: a simple web service was built to enable automatic data collection. When the user entered the page, a brief introduction to what will be collected was shown, and the user could either close the page or proceed with sharing their geographic location using HTML5 geolocation API[77]. The source of each datapoint – i.e., from where the user followed the link to the page – was not recorded[1]. Participant demographics were not collected.

Participants: The link was sent to all Futurice employees ($n = 208$) [2], shared on Twitter and Facebook, and with multiple crowdsourcing services[3]. On crowdsourcing services, \$0.05–0.25 was paid for each data entry. Only one entry per user was allowed.

Confidence: 397 random datapoints represent the whole internet with error of $\pm 5\%$ (95% confidence level). The sample is not uniform over the whole population.

[1] This is in fact unintentional. Unfortunately, the bug was discovered only after the collection.

[2] This is the number of unique email addresses on the internal mailing lists at the time of sending the survey. It is not exact representation of the number of employees at that time.

[3] microWorkers (microworkers.com), rapidWorkers (rapidworkers.com) and ShortTask (shorttask.com) were used.

After removing duplicate and invalid entries, the dataset consists of 397 IP-geographic coordinate pairs. Each datapoint contains following information, with examples:

IP address: 82.130.48.39

Latitude and longitude: 60.1906261,24.8352919

Accuracy: 40.0m[4]

The data does not necessarily represent Futurice employees very well. Especially with crowdsourcing services user demographics – including income and residential area. Furthermore, when employees are traveling abroad, they are likely staying in hotels and working from customer premises. This is presumably different from the users of the crowdsourcing services. This was not investigated further. Similar accuracy for IP addresses employees use was assumed.

The geolocation API reports the accuracy of the location. Histogram of accuracies is visualized in figure E.1. All entries with accuracy >500 meters were discarded, on the basis that either Wi-Fi or GPS location should report more accurate locations. If the browser obtained the geographic location from geocoding database, it does not provide meaningful information for evaluating the accuracy of similar databases.

At least 80%[5] of the datapoints are from devices without GPS – laptops or desktop computers – the accuracy requirement effectively removes all areas with no Wi-Fi access points. It is unclear whether this affects the accuracy of the dataset.

[4]Self-reported value. The radius of the circle where user should be. The center of the circle is at reported coordinates.

[5]Estimated from user-agent strings and screen resolutions.

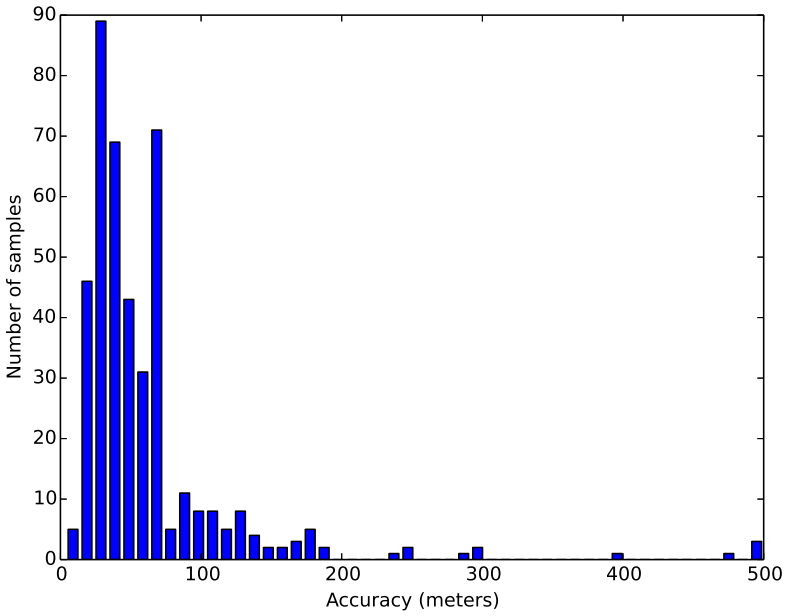

Figure E.1: Reported accuracy of location validation dataset.

Appendix F

Validation dataset for employee working times

Time of the study: the dataset was collected during November 2013.

Purpose of the survey: the dataset was collected to correlation between perceived working times and collected activity data.

Procedure: participants were instructed to write down working times (the beginning and end of the workday), times of lunch breaks and other breaks over a period of seven consecutive days. As a general guideline, 15 minutes resolution was deemed sufficient. 26 participants sent daily report emails. The rest sent a single report with all the entries. Each participant recorded their entries independently. After recording the working times, each participant completed a web-based survey.

Participants: 30 employees were randomly chosen. The selection criteria for participants was at least two successful logins to Futurice SSO between 2013-11-04 and 2013-11-17. As virtually no data exists for users who do not use Futurice services at least occasionally, the selection criteria was imposed to obtain more useful and accurate validation dataset. In total, 178 employees matched this criteria. Three employees declined to participate on the basis of too busy schedules. Another three employees did not respond to request to participate. In both cases, new participants were randomly selected.

Confidence: 30 participants represent the whole population with error of $\pm 14\%$ (90% confidence level). Six reselected participants potentially increase the error.

Working times were recorded independently, and people might interpret instructions differently. For instance, in the web-based survey sent after the data collection, only 24 (80%) claimed it was always clear what is considered as working times. People might act differently when they consciously monitor their working and break times. In the same survey, 19 participants (63%) estimated their reports were "very accurate". The rest described their reports more accurate than official hour markings. However, only 16 (53%) claimed to mark all working times to official hour reporting. Only two participants (7%) thought the participation might have affected their working days or official hour markings.

Appendix G

Example of WHOIS data

Example of WHOIS output for IP address from RIPE database. Unimportant parts and comments omitted.

```
1  inetnum:        83.145.221.208 - 83.145.221.223
2  netname:        FUTURICE-DSL
3  descr:          Futurice Oy, Helsinki
4  country:        FI
5  admin-c:        HN842-RIPE
6  tech-c:         NBL4-RIPE
7
8  role:           Nebula Internet Services
9  address:        Heikkilantie 2A
10 address:        00210 Helsinki
11 address:        Finland
12 ...
13 nic-hdl:        NBL4-RIPE
14
15 person:         Hanno Nevanlinna
16 address:        Futurice Oy
17 address:        Vattuniemenranta 2, 5 krs
18 address:        00210 Helsinki
19 address:        Finland
20 phone:          +358207479210
21 nic-hdl:        HN842-RIPE
```

Example of WHOIS output for domain **futurice.com**. Unimportant parts and comments omitted.

```
Domain Name: FUTURICE.COM
Registry Domain ID: 32355694_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.godaddy.com
Registrar URL: http://www.godaddy.com
Update Date: 2011-06-17 07:05:44
Creation Date: 2000-08-04 02:59:12
Registrar Registration Expiration Date: 2014-08-04 02:59:12
Registrar: GoDaddy.com, LLC
Registrar IANA ID: 146
Registrar Abuse Contact Email: abuse@godaddy.com
Registrar Abuse Contact Phone: +1.480-624-2505
Domain Status: clientTransferProhibited
Domain Status: clientUpdateProhibited
Domain Status: clientRenewProhibited
Domain Status: clientDeleteProhibited
Registry Registrant ID:
Registrant Name: Mats Malmsten
Registrant Organization: Futurice Oy
Registrant Street: Vattuniemenranta 2
Registrant City: Helsinki
Registrant State/Province:
Registrant Postal Code: 00210
Registrant Country: Finland
Registrant Phone: +358.207479210
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email: admin@futurice.com
Registry Admin ID:
Name Server: NS1.FUTURICE.COM
Name Server: NS2.FUTURICE.COM
DNSSEC: unsigned
Last update of WHOIS database: 2013-12-12T15:00:00Z
```