Trusted Execution on Commodity Devices

Use Case: Online User Authentication

Doctoral Dissertation by

Ijlal Loutfi

Submitted to the
Faculty of Mathematics and Natural Sciences at the University of Oslo
for the degree Philosophiae Doctor in Computer Science

Date of submission: 05. 04. 2019

Department of Informatics
University of Oslo
Norway

Oslo, April 2019

Abstract

Security-critical user applications routinely run within commodity devices, such as laptops and smartphones. The trusted computing base of these applications includes the devices' underlying system software (operating system, hypervisor and firmware), which is large, complex, and vulnerable to compromise. To mitigate this risk, Trusted Execution Environments, TEEs, offer a security primitive which protects the confidentiality and integrity of user applications' code and data, against a powerful adversary who controls the device's system software. While different realizations of TEEs are commonly deployed within today's devices, user applications do not make use of their services. Instead, TEEs are heavily used by few premium Service Providers, SPs, such as original equipment manufacturers. In fact, TEEs have been primarily designed to meet the requirements of such premium SPs, and have consequently underpriortized the needs of end-users and application developers. As a result, TEEs lack important features, such as open provisioning and trusted input channels.

The aim of this PhD project has been to investigate how available commodity TEEs can be used as primitives to build systems which meet the requirements of end-users and application developers. This work focuses on online user authentication, as it is a core component of numerous security-critical applications. As a starting point, we study the different modalities and protocols of user authentication. First, we conduct a usability study that explores the behaviour of IT professionals regarding password-based authentication. Second, we analyse the trust requirements of FIDO (Fast Identity Online), as an example of strong multi-factor-authentication protocols, and discuss its new distributed trust model. We then note that irrespective of the deployed authentication protocol, all authentication client applications are security-critical, and can benefit from TEEs. Therefore, we design, implement and evaluate TrustUI, a solution which enables secure end-to-end input and output channels from end-users to online service providers. TrustUI uses two TEE primitives which are system management mode, and personal security devices that are based on secure elements. Finally, we present SMMDecoy, a new architecture for detecting firmware keyloggers that can compromise the confidentiality of the keyboard's user interface. We use system management mode and Intel SGX as TEE primitives.

This thesis makes multiple contributions to the fields of user authentication and system security. The presented solutions can be generalized to other security-critical user applications, and can be practically deployed to today's off-the-shelf devices.

Acknowledgements

This thesis would not have been possible without the support and contribution of many people.

I would like to thank my supervisor, Professor Audun Jøsang. You have generously shared your knowledge throughout the years, and always kept your office door open. I benefited greatly from your research intuition to steer the direction of this thesis. I especially thank you for your trust and for the academic freedom you offered me throughout the entire PhD period.

Other professors and colleges at the department of Informatics have also been a great help. Thank you Professor Olaf Owe for your constant support and advice, especially during the last year of this PhD. For all the colleagues I have shared office space with, thank you for your great camaraderie and encouragement.

During this PhD, I have had the opportunity to complete an internship at HP security labs. Thank you David Plaquin for your excellent guidance. Your team has a very unique culture, and I am glad I got to experience it for myself.

Completing this PhD program meant leaving friends and family away. Thankfully, the distance was only physical.

Thank you dear uncles and cousins for never missing a birthday or a special occasion to call and express your support. Thank you dear grand-mother for your blessings and wisdom; you are wonderful.

I would like to express my deep gratitude and love to my parents and siblings.

To my mom, you have always believed in me and supported my dreams. Thank you for dedicating a great deal of your life to us, your kids, and to our education. I aspire to have your strength and loyalty.

To my dad, you have been unwavering in your support for me and my ambitions. Thank you for always sharing your insightful advice and emotional support whenever needed. I am humbled by your dedication and compassion.

To my brothers, I look fondly on the adventures and fun we shared as kids, and I look forward to creating more memories with you, and sharing more of your personal and professional successes. It's a privilege to be your sister.

To my little sister, you shine a light on all of us, your family. I simply love being your sister.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Commodity computing devices, such as laptops and smartphones, are an essential part of today's society. We routinely rely on them for both our professional and personal lives. Furthermore, many of the client applications they run are security-critical. Noteworthy examples are applications that support user authentication, online banking and e-government services.

This work focuses on online user authentication as a core component of such security-critical client applications. Authentication is indeed a fundamental part of today's digital experience. It allows us to ascertain our digital identities to various online service providers, and then gain access to their services. NIST (National Institute of Standards and Technology) formally defines authentication as "the verification of the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system" [88].

To achieve this goal, several authentication modalities have been developed over the years. Notable examples are passwords, verification codes through SMS, and fingerprint recognition. NIST refers to these modalities as factors, and classifies them into 3 categories: possession, knowledge and ownership [88]. From a historic perspective, online authentication protocols initially relied on passwords as a single factor. While passwords are to this day the most widely deployed modality, their weaknesses are also well recognized. Users routinely use weak passwords, reuse them across different services, and resort to insecure storage methods to remember them [17].

Therefore, two-factor authentication, 2FA, emerged as a natural progression. Before its adoption online, 2FA was successfully utilized by the banking sector for the authentication of its consumers to ATM machines. It used smart cards as a possession factor, and passwords/PINs as a knowledge factor. However, smart cards are not a suitable second factor for online user authentication applications. On the one hand, it is costly and infeasible for online service providers to distribute them to their large base of dynamic users. On the other hand, end-users would quickly become overwhelmed by the large number of smart cards they would need to manage. Therefore, online authentication found a more appropriate second factor, which is the phone: most users possess one, and they often keep it by their side. To authenticate, the online service provider sends a one-time-password to the user's mobile phone via SMS or a mobile application, which the user then enters into the login form along with the rest of

their credentials. As alternative authentication modalities, such as biometrics, became more mature, Multi-Factor Authentication (MFA) was proposed [69]. In 2005, the United States' Federal Financial Institutions Examination Council officially recommended its use for services requiring high assurance levels [90].

However, the successful adoption of these strong alternative authentication protocols was hindered by the lack of their interoperability: Every modality is implemented differently at the client side, and requires its own server-side technology [123]. Such a fragmented authentication ecosystem clearly overwhelms online service providers, and leaves them with the daunting task of navigating such a complexity, deploying the chosen authentication solution, and maintaining it at both the client and server sides.

Fast Identity Online, FIDO, is an industry standardization consortium which was formed in 2012, with the aim to solve the passwords problem and the MFA silos, through a set of device-centric protocols. FIDO proposes to split online user authentication into two phases: first, users authenticate themselves locally to a FIDO authenticator, using the modality of their choice. Then the authenticator authenticates itself on behalf of the user to the online service provider, using a standard challenge-response protocol [123] [36].

1.1.1 The Need For Trustworthy Platforms

Irrespective of the chosen authentication modality, all client applications for user authentication are security-sensitive. Therefore, application developers invest considerable effort to design and implement secure client applications that are free from software vulnerabilities, such as buffer overflows.

While such efforts are essential, they are insufficient. The Trusted Computing Base, TCB, of authentication applications is large and complex: It includes the platform's underlying system software, which when compromised, systematically threatens the security of all user-level applications. Within this context, system software is defined as to include the operating system, virtual machine manager and all the platforms' firmware embedded within, that run in CPU protection ring 0 and below. User-level software refers to third party-developed user applications running in CPU-ring 3 [56].

To put it differently, it matters little if a user chooses a perfectly strong unique password, when their operating system is infected with a keylogger leaking it to malicious third-parties [110]. Similarly, it matters little if a user relies on a passwordless MFA FIDO protocol, when firmware-resident malware is able to directly access the system's memory and steal its FIDO private keys. [92].

So why does the security of user authentication applications depend on the security of its underlying system software? The reason is the hierarchical architecture of commodity devices: privileged system software gets unrestricted access to all the resources of unprivileged user-level applications, because it controls its execution, memory, and access to the underlying hardware. In fact, CPUs originally implemented 4 protections rings (0-4), where ring 1 was intended for the kernel, ring 1 & 2 for the OS drivers, and ring 3 for user applications. Within this hierarchy, a process running in a specific ring has full access to code

and data of all processes running in the rings above. For performance reasons, OS vendors decided to run all OS modules, including its extensions and drivers in ring 0, user applications still run in ring 3, whereas ring 1 and 2 are not used. With the introduction of virtualization, it was necessary to create an even more privileged execution mode than the kernel ring, that is often referred to as ring -1. This design is used within both the Intel X86 and ARM architectures. Throughout this thesis, we adhere to the Intel terminology in order to maintain consistency.

Within such a hierarchical architecture, user-level applications have to trust that the million lines of code of the platforms' system software, which has been implemented by hundreds of developers over several decades, are completely free from vulnerabilities and backdoors. This is a strong trust assumption. Unfortunately, evidence shows that commodity system software is not secure and thereby not trustworthy. A non-exhaustive explanation of this assessment can be articulated as follows:

- Commodity operating systems:
  - The Large Code Base: Linux 3.6 was comprised of around 16 million Lines of Code (LOC) [83], while Windows 10 is estimated to be around 50 million LOC [49] . Such a code bloat can be attributed to the need to maintain backward compatibility, as well as running device drivers within the same privilege rings as the OS kernel, even though they do not need all the capabilities that the kernel level provides. This large code base makes the proper testing of operating systems and their formal verification daunting, if not impossible [26].

    As a result, it is estimated that for every 1000 lines of code of commodity monolithic operating systems, there exists between 16 to 75 bugs [95]. Furthermore, device drivers which make up around 75% of the OS's code base, contain 3 to 7 times higher rates of bugs than mature kernel code [15].
  - Unsafe languages: Programming languages such as C are commonly used for implementing commodity operating systems. However they are notoriously prone to security-critical errors such as buffer overflows, uninitialized pointers and integer overflows [64].
- Commodity firmware: As with operating systems, firmware is also prone to vulnerabilities caused by the use of unsafe languages, and the increased complexity of devices' functionalities. However, firmware malware is often more difficult to detect, because CPU-based defence mechanisms cannot monitor code residing within the execution environments of other components. Furthermore, recent years have witnessed an increasing ease of deploying firmware malware. This is especially true for devices such as the GPU (Graphical Professing Unit), which have become open to general-purpose computations [71]. Once a malware successfully infects a

platform's firmware, it can use its DMA (Direct Memory Access) capabilities to directly read and leak the system's memory [78].

Over the years, attackers have successfully exploited numerous system level vulnerabilities [22] [21] [23] [24], and there is no reason to believe that this trend will change any time soon.

1.1.2 The Need for Isolated Trustworthy Execution Environments

The hierarchical architecture of commodity platforms and the insecurity of their privileged system software, make it evident that such an execution environment is unfit for running security-sensitive applications. We name it the general-purpose or rich execution environment, REE [107].

A natural solution to this problem would be to design and deploy more secure commodity system software. While there are some ongoing projects working towards this goal, such as Qubes OS [4] and Coreboot [19], the reality is that currently deployed system software will not be displaced in the foreseeable future, due to two main reasons. First, there exists a large number of already deployed devices which have strong backward compatibility requirements, and have billions invested in their ecosystems [26]. Second, it is simply difficult to design and implement secure system software, in the face of a constantly changing threat environment, which has become clearly evident during the recent meltdown/spectre vulnerability disclosures [77].

An alternative solution is to run the security-sensitive subset of an application within a separate Trusted Execution Environments, TEE, that is isolated from the rich execution environment where the untrusted commodity system software runs. Such TEEs often have a limited set of resources and capabilities.

Generally speaking, isolation between the TEE and the rest of the platform can be enforced either in software or hardware. However, software solutions such as the ones based on hypervisors are undesirable, because they include untrusted system software in their TCB. Therefore, this thesis focuses on hardware-based trusted execution environments, because they provide stronger security guarantees: they can resist software-based attacks and even be resilient, to a certain degree, against physical tampering [30].

The idea of relying on hardware isolation to create a TEE with better security guarantees is not a new one. Over the years, different ways to realize hardware TEEs have been developed.

- Secure Elements (SE): These are special-purpose implementations of the traditional smart card. They are designed around a dedicated embedded micro-controller, a set of registers and system memory. Therefore, they can securely execute general-purpose applications, commonly referred to as applets, without interference from the host platform's untrusted system software. SEs can be implemented in two ways [86]:

1. A discrete hardware module outside the platform's main core. Such SEs are removable and can take two form factors: a Universal Integrated Circuit Card (UICC), commonly known as a SIM card, or a microSD card.
2. An embedded hardware module attached to the platform's motherboard or integrated with its NFC chip, and sharing some of the main system resources such as the memory buses. We refer to such implementations as Embedded Secure Elements, eSE [58].

- Processor Secure Environment: The platform's main processor core can implement several virtual cores which are logically isolated from each other, and can securely multiplex their execution within the same host [62]. Recent Intel X86 platforms implement one variant of this architecture, called Intel Security Guard Extensions (SGX). SGX allows ring- 3 applications to control their security by instantiating hardware TEEs, referred to as enclaves, within their own process address space [20]. Enclaves can then securely run the application's security-sensitive code and data, because they isolate it from all other untrusted software running within the paltform platform, which inludes the operating system, the virtual machine manager, all the platform's firmware, as well as the other unprivileged applications and enclaves. ARM platforms also implement their own variant of this configuration, named ARM TrustZone. When the new special Secure Monitor Call instruction (SMC) is executed, the processor securely transitions from the untrusted rich execution environment into the trusted one.

- System Management Mode (SMM): This is a highly privileged X86 CPU mode. SMM code is part of the BIOS that resides on the SPI flash memory. During the system boot-up, and before the operating system is loaded, the BIOS loads SMM into a hardware protected memory area called SMRAM, that is not addressable from any other CPU mode, including the kernel and VMX modes (Virtual Machine Extensions) [78].

- Trusted Platform Modules: TPMs are non-programmable security modules, often implemented as discrete micro-controller chips attached to the motherboard. They expose a number of APIs that can be called by the system software as well as user applications. These APIs securely implement services such as, cryptographic operations, integrity measurements, sealing and attestation [122].

Today's commodity platforms come already equipped with several of the above described hardware TEEs. For instance, a commodity laptop typically includes a TPM, an embedded secure element, a slot for one or more removable secure elements, a system management mode, and more recently, a CPU based-TEE such as Intel SGX. Similarly, a smart mobile phone typically includes a SIM card, an SD card, one or more embedded secure elements, and ARMTrust Zone [139].

1.2 Problem Statement

Hardware TEEs are abundantly deployed within today's commodity computing devices, and the idea of using them for hardening user-level applications security is already well-established. Therefore, it is reasonable to assume that security-sensitive user-level applications should be leveraging TEE functionalities to protect the integrity and confidentiality of their code and data. Unfortunately, this is not the case. User applications still rely on the default software-based protection mechanisms provided by the untrusted commodity system software. Commodity TEEs are instead extensively used by a handful of stakeholders: Original Equipment Manufacturers (OEMs), Mobile Network Operators (MNOs), and operating system vendors. These premium stakeholders use TEEs to protect their assets from being compromised by the platform's system software and end-users. For instance, MNOs use SIM cards to implement carrier locking, while OEMs use embedded secure elements and TPMs for Boot-loader locking and DRM (Digital Rights Management) [30].

The explanation behind this seemingly paradoxical situation, is the fact that commodity hardware TEEs have initially been designed to primarily satisfy the security requirements for those select few premium stakeholders. They have thus underprioritized the requirements for end-users and other application developers. For instance, embedded secure elements do not support user input trusted channels, because OEMs do not require end-user interaction to run their boot-loader locking and DRM use cases. Furthermore, most TEE's operate within a closed provisioning ecosystem, where no third-party code is allowed to run, without a prior business partnership with one of the premium stakeholders [8] [133].

1.3 Thesis Aim and Research Questions

There exist fundamental gaps between the security requirements for online user authentication developers and end users, and the functionalities offered by commodity hardware TEEs. Therefore, we define the main goal of this thesis as follows:

We aim to use available commodity hardware TEEs as primitives to build systems which meet the specific security requirements for online user authentication developers and end-users.

To achieve this goal, we define the following four research questions:

- **RQ1.** What are the failings of current password-based and strong online authentication solutions?

- **RQ2.** How can we leverage commodity hardware TEEs to provide **isolated execution and secure end-to-end input channels** between

online user authentication client-applications and end-users, even within a compromised platform?

- **RQ3.** How can we leverage commodity hardware TEEs to provide **secure end-to-end output channels** between online user authentication client-applications and end-users, even within compromised platforms?
- **RQ4.** How can we leverage commodity hardware TEEs to **detect compromise of commodity input channels**?

1.4 Approach and Research Method

In order to answer the research questions of the thesis, we use the following research approach:

1. We investigate the existing online user authentication modalities and protocols, study their limitations, and provide a set of their security requirements.
2. We study the currently deployed commodity TEEs, and provide a systematic analysis of the security properties they offer to end-user applications.
3. We perform a gap analysis study between the security requirements of online user authentication applications and the security properties provided by TEEs.
4. We design, implement and evaluate solutions which address a subset of the identified gaps:

    - Part I of TrustUI provides isolated execution for online user applications, as well trusted input channels between end-users and their service providers, even in the presence of a malicious system software.
    - Part II of TrustUI provides trusted end-to-end output channels between end-users and their online service providers, even in the presence of a malicious system software.
    - SMMdecoy addresses the challenge of detecting stealthy firmware-level malware which can compromise the confidentiality of the keyboard's user interface.

1.5 Structure of the Thesis

This work is written in the form of a cumulative thesis, compiling the results of four research papers.

Part I is comprised of 5 chapters: Chapter 1 introduces the motivation behind

this research, and defines its main research questions. Chapter 2 provides an overview of relevant background concepts. Chapter 3 presents the results of a gap analysis study between the security requirements for online user authentication applications, and the functionalities offered by currently deployed commodity hardware-TEEs. Chapter 4 provides a summary of the four research papers that are part of this thesis, as well as a brief overview of additional papers and filed patents that are related to this research area, but are not included in this thesis. Chapter 5 discusses how the main papers answer the initially outlined research questions and summarizes their research contributions. This thesis concludes with an overview of relevant future research directions.

The four main research papers are compiled and included in the second part of this thesis.

Chapter 2

Background

2.1 System Integrity and Integrity Targets

The integrity of a computing platform depends on the sum of the code and data that govern its behaviour. When end-users and other third-parties interact with a computing system, they rely on its integrity to trust it with their input, and to accept the output of its computations as valid. Therefore, being able to verify a system' integrity, especially a remote one, is an essential requirement for facilitating multilateral communication. Assessing a system's integrity involves a secure verifier which defines a desired integrity target, and a prover which provides evidence that it meets the said target, and that it is therefore trustworthy [109].

Over the years, numerous models for defining integrity targets have been proposed. Biba Strict is one such model which requires all executed code to be formally verified, and all input data to be of an equal integrity class or higher [54] [109]. However, such strict models are impractical to deploy, especially within commodity platforms. Therefore, more practical integrity models have been proposed, such as the Usable Mandatory Integrity Protection (UMIP) [74]. These alternative models aim to offer the same integrity guarantees as the classic ones, but make more assumptions in order to allow the system to accept non-trusted data with lower integrity classes. For instance, UMIP uses discretionary access control (DAC) permissions on binaries to label the integrity of processes in which they execute [109]. A more recent practical model is the one defined by the Trusted Computing Group (TCG), an industry standard group for trusted infrastructures. TCG requires the prover to exclusively execute code provided by a trusted distribution party, starting at the platform's boot-time. Examples include a boot-loader image that has been signed by a firmware provider that is trusted by the platform's OEM or device owner.

2.2 TPM 1.2

The Trusted Platform Module (TPM) is a TCG (Trusted Computing Group) specification for a security module. The goal of TPM is to establish trust in the platform's integrity, by proving that it satisfies the TCG's integrity target. To put it differently, TPM establishes that a platform exclusively executes code provided by a trusted distribution party, starting at the platform's boot-time.

TPM is often implemented as a discrete micro-controller chip attached to the motherboard. It can also be implemented as part of the platform's firmware. At

its core, TPM is a cryptographic co-processor which implements standardized cryptosystems, such as AES, RSA and SHA1, and uses a hardware random number generator for secure key generation. In TPM 1.2, each chip has a unique RSA identity key pair referred to as the endorsement key pair, EK, which is signed by a trusted certificate authority, which could be the manufacturer. The EK as well as its signed certificate are permanently embedded within the TPM at manufacturing time [6]. However, TPM is non-programmable and does not execute application-specific logic. It defines and exposes a set of APIs to the platform's operating system and applications. In the following sections, we explore TPM's architecture and the core functionalities it enables.

2.2.1 Protected Storage and The Root of Trust of Storage

TPM is equipped with a general purpose non-volatile RAM (NVRAM). It is used to securely store persistent security-sensitive data such as passwords and cryptographic keys. Because the amount of NVRAM is limited, sensitive data might need to be moved outside the trusted perimeter of the TPM. Once outside, this data is encrypted by the Storage Hierarchy key, SRK, which is always securely stored within the TPM's NVRAM. Unlike the endorsement key, the SRK is not embedded within the TPM at manufacturing time. It is generated after a TPM user takes ownership of the platform. SRK is used to protect all the keys which are subsequently created by the operating system and user-level applications. It is therefore referred to as the Root of Trust of Storage, RTS [122]

2.2.2 Platform Configuration Registers and Integrity Measurements

One of the most important components of the TPM is its bank of Platform Configuration Registers (PCR). A PCR is a 160 bit wide register that can hold a SHA-1 hash, which corresponds to a measurement of a piece of software or firmware present on the platform. It is not possible to write directly to a PCR. Furthermore, PCRs have only one single allowed operation, called extend. This operation computes the new value of a PCR as a SHA-1 hash of the concatenation of its old value and the most recently measured software. By definition, the extend operation is non-commutative, which means that PCR values reflect the order of the recorded events. PCRs are reset to an initial value (usually 0) at power-on [122].

Thanks to these properties, a PCR register can be used to record an ordered chain of events. For instance, when a platform is booting, the BIOS can compute the hash of the next piece of software to run, which is the boot-loader, extend the PCR register value, and finally hand over execution to the boot-loader. This "measure before execute" principle continues until the platform is booted. At the end, we get a trusted log of the software that the platform has booted. This is a requirement for assessing the system's integrity and thus its trustworthiness, according to the integrity target model defined by the TCG[52].

In order to establish trust in the final PCR measurement, it is essential to have trust in the first piece of code that the platform executes, because it cannot be measured by any prior software. This self-measured code is called the Core Root of Trust of Measurement, CRTM, and is often implemented as part of the host platform's motherboard, and not the TPM. While TCG recommends CRTM to be immutable, different platforms implement it differently(BIOS, UEFI) [6].

2.2.3 Remote Attestation and the the Root of Trust of Reporting

Integrity measurements are a first step towards assessing a platform's integrity. A subsequent step is to communicate the final PCR measurements to a third-party, capable of evaluating whether it matches one of the expected trustworthy values. This process is referred to as remote attestation. TPM measurements are signed with the Attestation Key, AIK, a short-lived RSA key, which itself is signed by the TPM's endorsement key. Therefore, the attestation key is the TPM's Root of Trust of Reporting, RTR.

Since the attestation signature is bound to the EK which can uniquely identify the platform, the remote attestation verification process can compromise the privacy of end-users. To mitigate this risk, a trusted third-party called the Privacy Certificate Authority (PrivacyCA) can be used instead. It vouches for the authenticity of TPM quotes, without revealing their identity to the verifiers. Alternatively, a zero-knowledge proof protocol called Direct Anonymous Attestation can also be used, without requiring any trusted third parties [9] [122].

2.2.4 TPM 1.2 Authorization

A hierarchy is a collection of entities that are related and managed as a group. Those entities include permanent objects (the hierarchy handles), primary objects at the root of a tree, and other objects, such as keys, within the tree branches.

TPM 1.2 has only one key hierarchy, represented by the owner authorization and the storage root key (SRK). There can only be one SRK, and therefore only one single hierarchy. The SRK is generated randomly, can not be reproduced once it has been erased, and it can not be swapped out of the TPM. Children-keys can be recursively created and wrapped by the SRK. This creates one deep key hierarchy that is under the control of one owner which is represented by an authorization value [115] [52].

2.2.5 TPM 2.0

The latest TPM 2.0 specifications introduce several manageability and security enhancements over TPM 1.2.

In fact, the single TPM 1.2 hierarchy architecture leads to considerable challenges with regards to how TPM platforms are managed after deployment. This is mainly because of the overlapping authorization domains between the TPM firmware, the TPM owner (e.g.: IT administrator who owns the platform

in which the TPM is embedded), and the TPM end-user. In order to overcome these management challenges, TPM 2.0 introduces a new architecture of multiple hierarchies [52]:

1. Standard storage hierarchy: It is under the control of the device owner, and essentially replicates the TPM 1.2 SRK hierarchy.
2. Platform hierarchy: It is exclusive used by the OEM's BIOS and system management mode.
3. Endorsement hierarchy or privacy hierarchy: It prevents using the TPM for attestation without the approval of the device's owner.
4. Null hierarchy: It uses the TPM only as a cryptographic co-processor, and is therefore open for all types of users.

Furthermore, TPM 2.0 offers greater agility regarding the cryptographic algorithms it supports, as opposed to TPM 1.2 which only supports the RSA and SHA-1 algorithms. Indeed, TPM 2.0 allows manufacturers to choose from a wide list of cryptographic algorithms which include elliptic-curve cryptography, RSA and SHA-2 [7].

2.3 Smart Cards and Secure Elements

A smart card is a tamper-resistant computer in the housing of a credit card. It provides the following core functionalities to its hosted applications [102]:

- A secure execution environment enabled by its isolated embedded micro-controller.
- Secure storage for protecting persistent data such as cryptographic keys, and authentication credentials.
- Hardware-based implementation of cryptographic algorithms.
- A stable security evaluation and certification process according to the Common Criteria (CC) [53].

When smart cards were first invented in the 1970s, they could only run one single application at a time. This changed in the mid-90s, when smart card operating systems, such as MULTOS [89] and JavaCard [14], were developed. They securely partition the card's physical resources and create multiple trust domains which can securely multiplex the execution of several applications within the same smart card. Smart Card operating systems also provide a unified high level API and development tools for application developers. These properties have enabled the rapid design and deployment of applications which leverage smart cards within different industries. One prominent success story took place in the 90s within the telecommunication industry, when GSM mobile networks (2G)

chose smart cards to implement their subscriber identity modules (SIM). Today, smart cards are ubiquitously used to host credentials, such as passwords and cryptographic keys, and have become a standard within the payment industry, as is the case with the EMV payment card standards by Visa, MasterCard and Europay [35]. These smart cards communicate with the host platform through standardized wired and wireless interfaces [51] [12] [62].

Secure elements are a special implementation of the traditional smart card, which focus on providing more tamper resistance and security guarantees to their host applications. It also adapts its design to meet the requirements for newer platforms, namely smartphones and IoT devices. SEs are often based on the JavaCard OS, and adhere to the global platform specification for loading and managing applications. Within commodity devices, SEs take one of the three following form factors [58]:

1. An embedded chip within the platform's main chipset (embedded SE),
2. A removable Universal Integrated Circuit Card (UICC), commonly known as a SIM card,
3. A removable micro Secure Digital Card, commonly abbreviated as SD card.

2.4 Intel Software Guard Extensions

Intel Software Guard Extensions (SGX) are recent security extensions for the X86 Intel processor family. They allow ring-3 applications to control their security by instantiating hardware TEEs, referred to as enclaves, within their process address space [20]. Enclaves can then securely run the application's security-sensitive code and data, because they isolate it from all other untrusted software running within the same platform. This includes the operating system, the virtual machine manager, the firmware, as well the other unprivileged applications and enclaves. These strong security guarantees are facilitated by two main architectural changes:

- Memory Encryption Engine, MEE: When the enclave's code or data leaves the trusted CPU perimeter and moves into the system memory, its security can be compromised by a malicious privileged system software, or a physical attacker with access to the memory bus. Therefore, the MEE encrypts all the enclaves' code and data leaving the CPU. It uses a hardware protected key which is regenerated upon every platform reboot [20].
- Hardware-Enforced Access Control Mechanisms: During address translation, the CPU performs extra access control checks for all memory read/write requests into the enclave. All illegal requests coming from outside the enclave result in a page fault. However, when the enclave's

code and data are within the CPU perimeters, they are kept in plain text. This makes them vulnerable to cache memory attacks. Therefore, the CPU flushes the enclave's logical core cached Transaction Lookaside buffer entries (TLB), before switching into the enclave mode [59].

2.4.1 Intel Remote Attestation

Remote attestation is an interactive protocol between 1) the attested platform, 2) the attesting remote service provider and 3) the Intel Attestation Service (IAS), an online service operated by Intel [20].

Each SGX hardware has a unique attestation key, which is only accessible by special Intel SGX architectural enclaves, such as the quoting enclave. During an attestation protocol, the quoting enclave first forms a structure referred to as a quote, composed of a hash of measurements reflecting the order and the content of code which has been initially loaded into the enclave by the operating system. Subsequently, the quoting enclave signs the quote structure using the attestation key, and forwards it to the remote service provider and the Intel Attestation Service. The IAS verifies that the signature has been generated by a genuine non-revoked Intel SGX hardware, while the service provider compares the measurement value to a reference one, so as to ensure that the enclave code has not been tampered with during the loading process [55][20] .

2.5 System Management Mode

System Management Mode, SMM, is a highly privileged X86 CPU mode. SMM code is part of the BIOS that resides on the SPI flash memory. During the system boot-up, and before the operating system is loaded, the BIOS loads SMM into a hardware protected memory area referred to as SMRAM, which is not addressable from any other CPU mode, including the kernel and hypervisor modes. All illegal requests are forwarded instead to the video memory by default. Therefore, SMRAM can be used for both secure storage and isolated execution [61] [78].

SMM implements several System Management Interrupt handlers (SMI), which primarily manage system functions, such as power and heat control. In order to execute SMI handlers, an SMM pin should be asserted. Before the system switches to SMM mode, the CPU state is securely saved into SMRAM, so that it can return to it upon exiting SMM. This makes SMM highly transparent to all privileged system software [55].

Recent academic research groups have been revisiting the purpose of SMM, and proposed to open it for several non-traditional use cases, such as debugging and system introspection. For instance, IOCheck and SMMDumper use SMM to reliably scan system memory and dump it for forensic analysis [105] [143]. HyperCheck uses SMM for hypervisor integrity verification. Aurora leverages SMM to augment Intel SGX enclaves with trusted network and time services[76]. While TrustLogin uses it to provide an end-to-end encrypted input channel from the keyboard to the Network Interface card [144].

2.6 Personal Security devices

Smart cards and secure elements provide important security primitives to the applications they host, such as secure storage for their sensitive data at rest, and isolated execution of their code. However, they provide no trustworthy input/output channels to the end user and cannot ”protect the owner from abuse of the smart card” [126]. Let us take the example of a smart card attached to a host PC platform, and which is used to securely sign messages/emails that the user sends from their PC. We consider the host platform to be untrustworthy and potentially compromised. While such a malicious platform is unable to access the private signing keys which are securely stored within the smart card, it can alter the messages being sent to the said smart card. For example, when the user types an email message saying ”N is a spy”, the malicious email application can alter the message to ”N in not a spy”. The smart card and the end user would then approve its signature unsuspectingly, as no trusted output channels are available for them to verify the content of the message being signed [126].

Personal Security Devices (PSD) were developed to mitigate such a class of vulnerabilities, which arise due to the lack of trusted input/output channels from end users to smart cards. We define PSDs as minimal embedded devices, which often adopt a portable form factor. They use an embedded secure element to execute security-sensitive operations outside the perimeter of the potentially compromised general-purpose platform, just as a traditional secure elements would. The SE is then augmented with a dedicated trusted output channel, such as a display screen. It is also desirable that the PSD has at least one dedicated trusted input channel, such as a keypad or a fingerprint reader. While PSDs have been around for many decades, such as the 3COM PalmPilot, they have been re-popularized recently thanks to the rise of user-centric and distributed architectures. They have also found a popular use case within the cryptocurrency community, where they are commonly referred to as wallets [73].

2.6.1 OffPAD

OffPAD is a personal security device which was developed within the University of Oslo as part of 2 research projects of 8 years. Its design is based on a secure element that is augmented with a dedicated keypad and screen. A fundamental problem is that most commercially available SEs cannot communicate with input/output devices. To solve this problem, OffPAD designed a customized dual-architecture which incorporates an ARM-Cortex M4 processor which acts as a proxy between the SE and the OffPAD’s I/O devices [132]. A summary of the OffPAD components is provided below:

- A Gemalto IDCore 10 Javacard-based secure element.
- An ARM-Cortex M4 processor-STM32f405.

- A secure Fingerprint reader, FPC1020Touch, which can store minutias securely using a secret key shared with the SE.
- A secure Display, e-ink 2.5 inch display.
- A 32 MB flash memory.
- A keypad.
- An NFC Transceiver, NPC 102A2EV.
- A microUsb USB ORG 2.0(High Speed).

OffPAD implements a proprietary low level software to manage the SE' communication with the peripherals. It also implements an open-source-friendly higher level software responsible for managing and loading the hosted application.

2.7 FIDO

Fast Identity Online, FIDO, is an industry standardization consortium which was formed in 2012, with the aim to solve the passwords problem, as well as the siloed MFA ecosystem, through a set of device-centric protocols. FIDO proposes to split online user authentication into two phases by using a client-side device as a proxy between end-users and their online service providers. First the end-user authenticates to the FIDO authenticator, which then authenticates on behalf of the user to the service prvider.

FIDO Authenticators: These are security modules that can be implemented as a discrete external hardware device, as an embedded hardware token or CPU-based TEE, or as a software process running within the host's kernel or user space. Every authenticator can implement one or more authentication modalities, such as fingerprint recognition, face recognition, passwords, or proofs of presence. Different manufacturers will provide different implementations and modalities, and it is up to service providers to accept and register authenticators that match their required assurance levels [36]. This can be achieved through a remote attestation protocol, where the authenticator sends a signed quote to the online service provider claiming to meet a specific security level for implementing a specific authentication modality. For instance, an authenticator would want to claim that it runs a fingerprint recognition function within Intel SGX.

Attestation Key Pair: This is an identity key pair which is generated and embedded into the FIDO authenticator at manufacturing time. It is shared between a class of authenticators, which often have the same manufacturer and security level. The Attestation key should be signed by a certificate authority which is trusted by online service providers[36].

FIDO Registration: Every time a user wants to register with a new service provider, the authenticator generates a new private/public key pair, referred

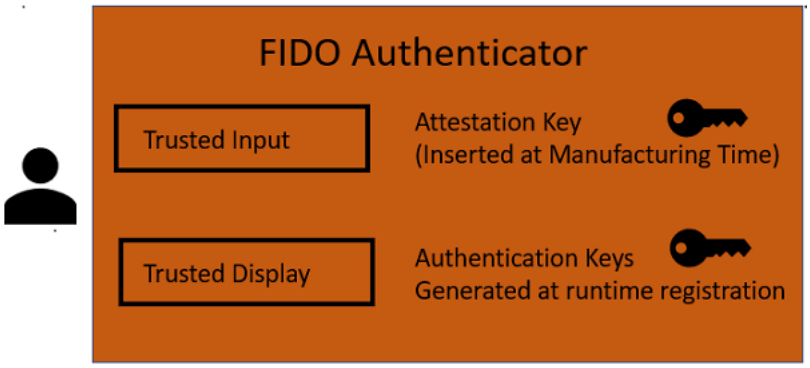

Figure 2.1: FIDO authenticator [36]

to as the authentication key pair. This key pair is unique for every unique combination of {end-user, service provider, authenticator}. The authenticator stores the authentication private key, signs it with the attestation key, and sends its corresponding signed public key along with a certificate to the service provider.

FIDO Authentication is a two-step protocol:

1. First mile authentication : Users authenticate themselves to the FIDO authenticator which they have already registered with the service provider.
2. Second mile authentication : The authenticator authenticates itself, on behalf of the user, to the online service provider, using a standard challenge-response protocol, where he uses the proper authentication key already registered with the service provider to sign the challenge [123] [36].

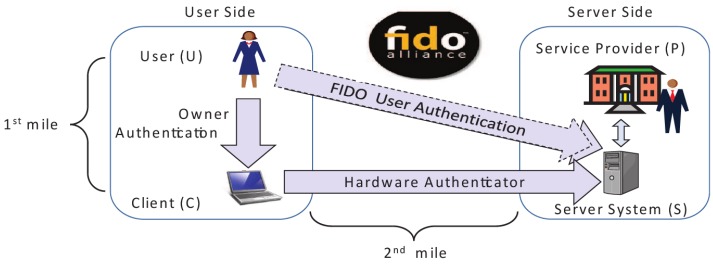

Figure 2.2: FIDO authentication [82]

Chapter 3

Commodity Hardware TEE Gaps

The ecosystem of commodity devices is made up of several stakeholders: device owners (end-users), Original Equipment Manufacturers (OEMs), Mobile Network Operators (MNOs), operating system vendors, and third party-application developers.

An initial literature review of available hardware TEEs reveals that third party-application developers do not currently use the strong security guarantees offered by these TEEs. This is due to the fact that commodity TEEs have been primarily designed to meet the requirements for OEMs, MNOs and OS developers, and have thus underprioritized the requirements for end-users and application developers [133].

In order to remedy this situation, we present in this chapter a gap analysis between the security functionalities that end-users and application developers require from commodity TEEs, and the functionalities offered by currently deployed TEEs. This chapter focuses on the TEEs surveyed in the background study of Chapter 2, and which are relevant to the remainder of this thesis.

The results of this analysis were fundamental in steering the direction of the rest of this research. We used the identified gaps as a guide to formulate the last 3 research questions of this research and choose relevant problems statement for the last 2 papers.

3.1 Security Requirements

In this first section, we answer the question of "what security services do online user authentication developers and end-users need from commodity hardware TEEs?"

3.1.1 Secure Storage

Online authentication applications need to ensure the confidentiality, integrity and freshness of their users' stored credentials at rest. The default mechanisms provided by commodity platforms rely on the operating system's file system permissions. Such solutions are vulnerable to a compromised operating system, as well as physical cold boot attacks[133] [119].

Therefore, online authentication requires the TEE to provide a secure storage mechanism which does not rely on the platform's system software. This requires the following three components [8]:

- Encrypting the credentials using cryptographic keys winch cannot be accessed by the platform's system software.
- A secure implementation of the relevant cryptographic algorithms.
- A secure non-volatile memory to persistently store monotonic counters across the platform's reboots.

3.1.2 Isolated Execution

Online user authentication applications need to preserve the confidentiality and integrity of their code and data at run time. A default mechanism to achieve this within commodity devices is to rely on the operating system's process-based isolation. However, this solution is vulnerable to a compromised operating system [119][133].

Therefore, online authentication applications need to run within a trusted isolated execution environment. The run-time isolation property can be implemented in various ways:

- Full hardware isolation: The code runs within a physically isolated processor, which does not share any context with the untrusted execution environment. Notable examples are co-processors, smart cards, and secure elements. Such solutions provide high protection guarantees against the host platform side channel attacks, by virtue of their complete isolation. However, they lack direct access to the system's memory.
- Multiplexed Logical Isolation: The security sensitive functions run within the same host commodity processor, and share its physical execution context. However, additional CPU-based hardware access control mechanisms are introduced in order to enforce strict logical isolation of resources between the secure execution environment and the untrusted one. A notable examples is Intel SGX, which allows applications to protect their sensitive code within a ring-3 enclave [86].

3.1.2.1 Non-Monolithic Trust Domain

Isolating the execution of the authentication application from the rest of the TEE is essential. However, when one TEE instance is used by multiple applications which belong to different non-trusting service providers, the TEE should provide adequate isolation between these applications.

3.1.3 Remote Attestation

Before granting users access to their services, online service providers need to ensure that the authentication credentials they receive are indeed coming from

trusted applications running within a particular trusted TEE. Remote attestation is the process which enables remote third-parties to verify the platform's state. When an online authentication application runs within the rich execution environment, its TCB includes the operating system, the platform's firmware and the application itself. Therefore, the remote service provider would need to evaluate the state of such a complex TCB, and decide whether to trust it [133]. This is a daunting task, and is referred to as the semantic gap [61].

Therefore, online authentication applications need a TEE with a small and stable TCB, in order to enable a meaningful remote attestation verification.

3.1.4 Open Secure Provisioning

Online authentication developers require an open provisioning policy which allows them to provision their code and data remotely into the TEE without having to enter into a business relationship with a centralized authority, such as MNOs, OEMs, or OS providers [8]. The authorization of authentication developers by device owners should be sufficient. Remote attestation can then be used to enable secure remote communication between the application developers and the TEE.

3.1.5 Trusted Path

Users routinely rely on their device's standard input interfaces, such as the keyboard, to enter their authentication credentials. This input is then processed by the platform's firmware and operating system, before reaching the target application or TEE, which then forwards it over the network to a remote service provider. The latter relies on this input to make various security decisions, such as granting access to end-users and approving their transactions. However, a malicious system software, keyboard firmware or browser can modify the credential or leak it to malicious third-parties. This compromises the confidentiality and integrity of the input.

Users also rely of the content rendered by the browser which is then displayed by the monitor to evaluate if they are interacting with the correct application, and select the input fields where they should type in their credentials. However, nothing stops a compromised malicious browser from injecting fake authentic-looking fields prompting the user to enter their credentials, in order to capture and leak them to malicious third-parties. As the rest of the web-page, such as the URL and the 'https' lock, remain unchanged, it is very difficult for users to know that the web-page has been tampered with [140] [11]. This type of malware is referred to as Man-in-The-Browser malware (MITB).

Indeed, the lack of trusted input and output channels compromises the trust between end-users and their service providers. Therefore, online authentication applications require trusted input and output paths to the TEE, which do not rely on the system software. These paths should ensure the integrity and confidentiality of the user's credentials. They should also authenticate the TEE to end-users.

3.2 TEE Gaps

In this section, we identify and discuss the security requirements which are not met by the currently deployed TEEs.

3.2.1 Lack of an Open Provisioning Ecosystem

Many of the TEEs deployed within commodity devices are not under the control of their device owners. For instance, OEMs control embedded secure elements and SMM, MNOs control SIM cards, and OS vendors control some embedded secure elements [133]. These premium service stakeholders exclusively use TEEs to run their own workloads, and prohibit application developers from provisioning and running theirs.

However, few exceptions have been documented, where large incumbent application developers entered into a business agreement with a premium stakeholder to allow them to provision their application into the commodity TEE. For instance, MNOs opened SIM cards for both the Vodafone payment app and the Norwegian "BankID på mobil". While OS vendors opened their embedded secure elements to GoogleWallet. Clearly, such business partnerships are out of reach for the other "normal " application developers [30].

3.2.2 Isolated Execution

TPMs are fixed-function modules, which are not suitable for general-purpose computations. They only offer a predefined set of APIs that the authentication application can call. While Intel SGX is a programmable TEE, its enclaves lack built-in support for system-level functions, such as networking and file system services [76]. Such a trust model might be sufficient for securing some server-side workloads, but not client-side authentication applications.

3.2.2.1 Non-Monolithic Trust Model

While SMM and removable SEs do provide a programmable isolated execution environment, they do not implement any isolation between applications running within the same TEE instance.

3.3 Lack of Trusted Input /Output Channels

When an OEM needs to securely communicate with a remote embedded controller, an end-to-end network encrypted channel is sufficient. Such a communication use case does not involve any interactions with end-users. Therefore, trusted input/output channels from end-users to their keyboard/display are not required. Clearly, this is not the case for online authentication applications.

Unfortunately, SMM is the only hardware TEE we surveyed which offers trusted input/output channels.

3.4 Summary of Gaps

We aggregate the results of this gap analysis into the following table, in order to provide a convenient visual summary of the identified TEE gaps.

	TPM	SIM	SD Card	Embedded SE	Intel SGX	SMM
Open Provisioning	Y	N	Y	N	Y	N
Isolated Execution	N	Y	Y	Y	Y* *No System Calls	Y
Secure Storage	Y	Y	Y	Y	Y	Y
Remote Attestation	Y	Y	Y	Y	Y	Y
Trusted Input	N	N	N	N	N	Y
Trusted Output	N	N	N	N	N	Y

Table 3.1: TEE gaps

Chapter 4

List of Research Papers

The research contributions of this thesis are presented as 4 research papers, which are briefly summarized in this chapter. The full text of the papers is presented in Part II of this thesis. Additionally, Section 4.5 presents a short overview of additional research contributions which are not part of this thesis. These are 2 patent applications and 2 research papers, to which the author of this thesis has contributed during her Ph.D. studies. The patents are not part of this thesis because they are still under review and contain commercially confidential material. While the papers represent preliminary work that is related to this thesis.

4.1 Paper I: Passwords Are Not Always Stronger on the Other Side of the Fence

Authors: Ijlal Loutfi, Audun Jøsang.
Publication: Proceedings for the 2015 NDSS Workshop on Usable Security, USEC'15. DOI: 10.14722/usec.2015.23005.

This paper presents the results of a usability study that explores the behaviour of IT professionals regarding passwords-based authentication [79].

Studying passwords is important because it is the most widely deployed authentication modality and is often used as a single factor to construct authentication protocols. Unfortunately, multiple studies and vulnerability disclosure reports show that end-users routinely use weak passwords, reuse them across different services, and resort to insecure storage methods to remember them.

Instead of acknowledging this situation as a natural consequence of the current situation where end-users need to manage a large number of passwords, IT professionals often attribute the fault to the "laziness and ignorance of end-users". Therefore, instead of researching and deploying more usable and stronger authentication protocols, the IT community's response often focuses on educating the supposedly ignorant end-users on how to better manage their passwords.

This paper challenges this assumption, and explores the hypothesis that "knowledge of good password habits is a necessary but not by itself a satisfactory requirement for a safe password behaviour". To achieve this goal, we conduct a usability survey exclusively targeting an audience of IT professionals, with good knowledge about security.

The survey focuses on 8 online service providers (Facebook, Gmail, LinkedIn, Twitter, Work/studies email, Bank account, online storage, online video games). For each service, 66 respondents reported their behaviour across the following password parameters: length, characters mix, memorability, reuse/uniqueness, usage of federated logins and usage of password managers. Overall, 2970 unique data points were collected.

The analysis of the data was completed in two iterations. In the first iteration, we performed a high-level statistical analysis in order to deduce patterns within the data set. In the second iteration, we measured the statistical association between the users' independent variables and their reported password behaviour. Therefore, we computed the Chi square test value of different combinations of pairs and evaluated its value against the null hypothesis. The significance value was set at 0.05. Finally, we computed the residual deviation for each combination of pairs, where we set the significance range at (-2,2) . This allowed us to identify the combination with the highest statistical significance.

The main result of this analysis was that we found no evidence of any statistically significant correlation between the perceived sensitivity of a service and the respondent's password behaviour. Other interesting findings were also reported. For instance most respondents do not use a mix of characters if it is not mandatory. This behaviour was more prevalent for social media services like Facebook and LinkedIn. While such services might not be highly sensitive to our respondents, many of them reported using them their federated login to authenticate to other service providers. Such behaviour creates long transitive chains of trust that are difficult to cognitively evaluate. We also found that only 69 of the reported passwords meet the requirements for what constitutes a strong password. This is a mere 12% of the data set.

This paper provides significant insights into the password habits of IT professionals. Although they possess enough cognitive knowledge to be fully aware of what constitutes an adequate password behaviour, they fail to materialize it into practical habits in many instances. It is hoped that the paper's results will confront IT professionals with their own password practices which fail to adhere to what they preach to end-users, and further motivate them to adopt stronger more usable forms of authentication [79].

4.2 Paper II: FIDO Trust Requirements

Authors: Ijlal Loutfi, Audun Jøsang.
Publication: Proceeding for the 20th Nordic Conference on Secure IT Systems 139–155, NordSec'15. DOI: 10.1000/182.

At the time of writing this paper, the first version of the FIDO protocols were just released. This paper was one of the first academic contributions to provide a systematic analysis of the protocols.

This paper analyses the trust requirements for FIDO and cross compares them to currently deployed authentication solutions. The two most important

results of this analysis are as follows:

- Eliminated Trust Requirements: Online service providers were traditionally liable to protecting users' credentials at the server side, e.g. passwords databases. However, the FIDO architecture frees service providers from such a liability. As a matter of fact, FIDO is defined as a device-centric protocol, where the secret credentials are stored at the client side, whereas the server only stores the public keys/certificates of the FIDO authenticators.

- Inherited Trust Requirements: The security of the FIDO protocols fundamentally depends on the authenticator's ability to preserve the confidentiality and integrity of its identity key, and to properly authenticate users locally to their authenticators. FIDO leaves the specific implementation of the authenticator to the discretion of third-party hardware manufacturers and software developers: A FIDO authenticator can be implemented as a software application running within the host platform's user or kernel space. It can also be implemented as a trusted application running within a hardware based TEE, or as a discrete external hardware module. Therefore, we conclude that the security of FIDO protocols depends on the security of the authenticator's host platform.

- New trust requirements: The FIDO consortium also acts as a certification authority which evaluates the security claims of FIDO authenticators, and evaluates the types of authentication modalities they implement before granting them a FIDO certificate which can be used during the attestation step that takes place during the registration phase. This certification process can breach the users' privacy, by enabling easier linkability of users' activities between colluding service providers. Therefore, FIDO introduces the FIDO consortium as a new trusted-third party.

Trust in FIDO authentication protocols is therefore distributed between end-users, authenticators and the the FIDO consortium. This is a new trust model for online user authentication, and should be reasoned about and modelled as such. We argue that describing FIDO as a device-centric protocol is a misrepresentation of such a complex and distributed trust model [82].

4.3 Paper III: TrustUI-EnablingTrusted User Input and Output Channels to Web-Applications in Untrusted Client Platforms

Authors: Ijlal Loutfi, Audun Jøsang.
Publication: Under Review at the 22nd Euromicro Conference on Digital System Design, DSD'19.

This paper presents the design, implementation and evaluation of TrustUI, a new solution architecture we propose for providing isolated execution, trusted

input channels and trusted output channels for online authentication client applications, even in the presence of a compromised platform. This paper focuses on the two following vulnerabilities against IO channels:

1. **Trusted Input Channel Attack**: When users need to communicate with their remote web service providers, they enter input data into their platforms through standard input interfaces, such as the keyboard. This input is then processed by the system software and the browser before being sent through the network to the web server. The latter relies on this input to make various security decisions, such as granting access to end-users and approving their transactions. However, a malicious browser, or keylogger which has hooked into the kernel, is able to leak the users' credentials to malicious third-parties [92].

2. **Trusted Output Attack**: Web users routinely rely on what the browser displays to them in order to make various decisions. For instance, they check the banking transaction amounts displayed on the screen before approving them for payment. However, nothing stops a compromised browser from redirecting the payment transaction to the attacker's account, then modifying the HTML rendering of the user's banking web page to still display the legitimate transaction. The end user would then unsuspectingly confirm the transaction, and the bank would process it as if it originated from a legitimate user. The compromised browser can also inject new authentic-looking fields prompting the user to enter sensitive information such as credit card details or authentication credentials. As the rest of the web-page, such as the URL and the 'https' lock, remain unchanged, it is very difficult for users to detect that the web page is tampered with [140] [11]. This type of malware is referred to as Man-in-The-Browser malware (MiTB).

Therefore, TrustUI presents trusted input and output channels, even in the presence of a malicious platform:

- **TrustUI-Trusted Output Solution**: Users cannot reliably detect whether the rendering of a web-page is altered, because the layout of a rogue page can look completely identical to a legitimate one. Furthermore, solutions which rely on out-of-band authentication modalities, such as SMS verification codes can also be defeated. This is due to two fundamental weaknesses: on the one hand, out-of-band modalities are subject to habituation: Users simply start confirming transactions out of habit without double-checking their details, or become inattentive to small transaction modifications such as similar bank accounts, or slightly rounded transaction amounts [3]. On the other hand, even when out-of-band verification is successfully carried out by the user, its effectiveness is limited to stopping attacks where only dynamic page values, such as transaction amounts, are modified. They are,

however, ineffective against attacks which inject new rogue web elements into the page, such as input fields prompting users to enter their passwords.

TrustUI argues that a web server, such as a bank, can increase its trust in the transaction it is processing if it gets a strong guarantee that it matches the screen displayed to the end-user. In a parallel world, web servers would have a way to access the users' screen before every transaction. In the absence of that, TrustUI proposes a two-step solution which gives similar guarantees. The first component is a screen capture engine which programmatically captures a bitmap representation of the rendered display directly from the GPU frame buffer, which can not be accessed by MiTB malware. The second component is an image analysis engine, which cross-compares the captured image to a reference one. This allows it to detect any rogue injected elements.

- **TrustUI-Trusted Input Solution**: In order to protect the confidentiality of user input from a compromised browser and operating system, TrustUI aims to build an end-to-end encrypted input channel between the end-users' input interface and their web server. A straightforward solution to this problem would be to encrypt all sensitive input at the level of the keyboard's subsystem, before it becomes accessible to the rest of the platform's system software and browser. However, this solution is impractical within out-of-the box commodity platforms, since the standard keyboards are not able to perform secure cryptographic operations and negotiate keys. Instead, TrustUI proposes to equip users with PSDs (Personal Security Devices) with dedicated input keypads. These PSDs can establish an end-to-end encrypted channel to a web enclave which TrustUI establishes within the application's address space. PSDs and the SGX enclave can then negotiate and establish an end-to-end encrypted channel. The end-user can use the PSD's dedicated keypad to enter their sensitive input, send it encrypted to the web enclave and then to its corresponding web server.

Finally, the paper experimentally evaluates the core functionalities of TrustUI, and reports its results:

1. Trusted input evaluation: We evaluate the trusted input solution, by using an Arduino Uno platform to implement and simulate the functionalities of a personal security device with a dedicated keyboard. The Arduino is used to send encrypted messages over a serial communication port to the application, which creates an enclave within its address space.

2. Trusted output evaluation: This experiment uses a mock-up login page where we mimic an MiTB attack which injects rogue input fields into a webpage. First, we programmatically capture the displayed image of the rogue login page using an open-source implementation of the Windows GDI screen capture API (Graphics Device interface). Second, we perform a pixel-by-pixel comparison of the bitmap representation of the reference logging page and the captured one. As these are different images, the user

gets a trigger that the webpage is malicious. While the delay introduced by the screen capture image is insignificant, the image analysis takes 4 seconds to complete.

4.4 Paper IV: SMMDecoy-Detecting GPU Keyloggers Using Security By Deception Techniques

Authors: Ijlal Loutfi.
Publication: Proceedings of the 5th International Conference on Information Systems Security and Privacy 580-587, ICISSP'19.
DOI: 10.5220/0007578505800587
Published as a short paper

This paper presents SMMDecoy, a solution for detecting firmware-based keyloggers which compromise the integrity and confidentiality of the users' keyboard input.

Studying the security of the keyboard interface is an important question, because millions of users routinely use it as their primary input interface, and rely on its security when they type in their security-critical input, such as authentication credentials and credit card details. Unfortunately, both the PS/2 and USB keyboards have open buffers that can be read by the platform's operating system and firmware. Keyloggers, are one class of malware which abuse this vulnerability. Once they infect a platform, keyloggers log the keyboard's activity, and leak it to malicious third-parties. Keyloggers can hook into the platform's user space API, kernel space API, or firmware.

This paper focuses on detecting the stealthiest variance of keyloggers, which is the one deployed within the firmware of the platform's IO devices, such as the GPU (Graphical Processing Unit).

In this paper, we propose SMMDecoy as a novel solution architecture which uses principles from the paradigm of security-by-deception to detect GPU keyloggers. It proposes to use a transparent mechanism which injects intentionally crafted noise that mimics authentication credentials, to the keyboard's buffer, and wait for any potential GPU keyloggers to sniff it. SMMDecoy would subsequently send a list of the injected decoy credentials to a remote third-party. Finally, the service provider triggers a detection alarm when it receives an authentication request using any of the decoy credentials. The paper assumes a strong adversary that can infect both the GPU and the kernel. Therefore, SMMDecoy should be deployed within the System Management Mode, SMM, in order to take advantage of its integrity and transparency guarantees. the paper also proposes to use Intel Software Guard Extension remote attestation capabilities to send decoy credentials securely over the internet [78].

4.5 Other Contributions

4.5.1 Patent Applications I & II

The author of this thesis spent a 6 months internship at the security labs of HP Inc. in Bristol, UK. She worked on trusted execution environments and how they can be used to extend the trust from the platform's firmware controlled by the OEM, to the operating system execution environment.
The results of this internship were 2 patent applications where the author is a co-inventor, alongside her internship advisor. Both patents are currently under a confidentiality agreement, and are under review at the US patents office.

Inventors: Ijlal, Loutfi and David, Plaquin.
Invention Owner: HP Inc.
Status: Under Review.

4.5.2 Paper V: Privacy Concerns of TPM 2.0

Authors: Ijlal Loutfi, Audun Jøsang.
Publication: Proceeding for the 15th European Conference on Information Warfare and Security p. 205-211, ECCWS'16

The integrity of a computing platform is defined by the sum of the code and data that govern its behaviour. The goal of the TPM (Trusted Platform Module) is to establish trust in the integrity of its host platform, by providing cryptographic evidence to verifying remote parties that the said platform only executes code provided by trusted distribution parties. Remote attestation is the TPM functionality which enables this integrity verification. At the time of writing this paper, TPM 2.0 specifications have just been released.

This paper focuses on studying the new TPM 2.0 remote attestation functionality and its privacy. This study uncovers two TPM architectural design decisions which could compromise the privacy of end-users and make them more vulnerable to tracking by the TPM manufacturers as well as law enforcement entities.

First, we discuss how TPM 2.0 specifications allow the platform's firmware to enable/disable the platform hierarchy without the user's consent. We call this vulnerability TOOPH (The Always-On Platform Hierarchy).

Second, we discuss the "re-certification" process, which is a new mechanism that maintains a cryptographically traceable link from the platform hierarchy controlled by the TPM manufacturer, to all the other hierarchies, even when their root keys been deleted and re-generated by the user. This defeats the isolation guarantees that TPM 2.0 provides between its 4 hierarchies. In order to mitigate the recertification vulnerability, the paper proposes to implement a user-level monitoring application, which notifies end-users about silent TPM actions that were not implemented to ask for the users' consent [52].

4.5.3 Paper VI: 1,2, Pause: Lets Start by Meaningfully Navigating the Current Online Authentication Solutions Space

Authors: Ijlal Loutfi, Audun Jøsang.
Publication: Proceedings for the 9th IFIP WG 11.11 International Conference on Trust Management, Volume 454 p. 165-176, IFIPTM'15.

This paper provides a framework of a well-motivated set of attributes, for categorizing and assessing online authentication solutions. This paper also uses this framework to analyse and compare two important online authentication protocols: LUCIDMAN as an example of a user-centric solution, and FIDO as an example of a device-centric one. Some of the framework attributes discussed in this paper are: the solution's security guarantees, its usability, and its deployability. The results of this research are anticipated to make the navigation of the online authentication solutions' space more systematic, and facilitate knowledge transfer between different stakeholders [80]

Chapter 5

Conclusion

This chapter returns to the research questions formulated in Section 1.2 and discusses them in connection with the contributions of this thesis. This chapter also suggests directions for future work.

5.1 Summary of Contributions

This research project started with one simple question: To what degree are currently deployed online authentication solutions secure? And if their security is insufficient, what measures can we take to increase their assurance levels? Answering these questions set us on a long and exciting journey: What started as research within the fields of security and the application layer, quickly turned into a study of the platform's underlying trusted execution environments.

In the first part of this research project, we focused on studying password-based authentication, as well as FIDO strong multi-factor authentication protocols. The results of the 2 first papers made it evident that irrespective of the chosen authentication modality, all client applications for user authentication are security-critical, and that their TCB includes the platform's underlying system software. Therefore, for the remainder of this research project, we focused on studying the security of commodity platforms, specifically Intel X86.

Several studies and vulnerability reports show that commodity system software is not secure and therefore non-trustworthy. To mitigate this risk, hardware-based trusted execution environments offer strong security guarantees to protect the confidentiality and integrity of user-level applications, even against a compromised system software. While commodity platforms are equipped with numerous commodity TEEs, authentication applications do not use their functionalities. In Chapter 3, we presented the results of a gap analysis study, where we map the security requirements for online authentication applications to the security features currently deployed by commodity TEEs. This analysis identified a number of gaps which clearly prevent user-level applications from using TEEs: 1) lack of open provisioning, 2) lack of trusted input channels, 3) lack of trusted output channels, and 4) lack of isolation within the same TEE instance.

The results of this gap analysis were fundamental in steering the research direction of the rest of this research. We used the identified gaps to formulate the 3 last research questions of this thesis.

In the rest of this conclusion chapter, we discuss the 4 research questions of this thesis individually:

RQ1. What are the failings of current password-based and strong online authentication solutions?

Formulating this question started with the realization that passwords are the most widely deployed user authentication method. While people have been predicting their demise with the rise of alternative more usable modalities such as biometrics, passwords are still well and alive, and are currently used by 99,9% of online service providers.

This first research queston is addressed in both paper I and paper II. In the first paper, we present the results of a usability study that explores the behaviour of IT professionals regarding password-based authentication. The aim of this study is to challenge the assumption that only ignorant users have poor password usage habits, and explore the hypothesis that "knowledge of good password habits is a necessary but not by itself a satisfactory requirement for a safe password behaviour". The analysis of the data set collected from IT professionals reveals no evidence of any statistically significant correlation between the perceived sensitivity of an online service and the respondent's password strength.

The results of paper 1 reflect the fact that password-based-authentication have inherent weaknesses , especially when used as a single factor to construct authentication protocols. Therefore, we decided to study the so-called strong multi-authentication-protocols, as reported in paper 2. This work focuses specifically on providing a systematic analysis of the trust requirement for the FIDO protocols. An interesting result of this study reveals that FIDO protocols introduce a new distributed trust model, where the security of the authenticator's underlying platform is a strong trust requirement which can be difficult to satisfy [79].

RQ2. How can we leverage commodity hardware TEEs to provide isolated execution and secure end-to-end input channels between online user authentication client-applications and end-users, even within a compromised platform?

In paper 3, we introduce the design, implementation and evaluation of TrustUI. This new solution architecture uses 2 TEEs, Intel SGX and a PSD (Personal Security Device) which is based on an embedded secure element. Intel SGX creates an enclave which enables the isolated execution of the user authentication application. Intel SGX and the PSD also establish an end-to-end encrypted channel between the end user and the web enclave. The user can then use the PSD's dedicated keypad to enter sensitive input, which will be sent securely to the web enclave and then to the web server. TrustUI provides strong security guarantees even against a strong adversary who controls the platform's underlying system software.

RQ3. How can we leverage commodity hardware TEEs to provide secure end-to-end output channels between online user authentication client-applications and end-users, even within compromised platforms

The second part of TrustUI which is presented in paper 3, introduces the design, implementation and evaluation of a new architecture solution which provides trusted output channels from end-users to their online service providers, even in the presence of a malicious browser which changes the rendering of the pages displayed to the user. TrustUI is a two-step solution: the first component is a screen capture engine which programmatically captures a bitmap representation of the rendered display directly from the GPU frame buffer, which is out of reach from Man-in-the-Browser malware. The second component is the image analysis engine, which cross-compares the captured image to a reference one. This allows it to detect any rogue injected elements. The two components can be implemented within a user-level TEE such as Intel SGX.

RQ4. How can we leverage commodity hardware TEEs to detect compromise of commodity input channels?

In paper 4, we introduce SMMDecoy as a novel solution for detecting firmware-based keylogger malware which can compromise the integrity and confidentiality of the keyboard's user interface.

SMMDecoy uses principles from the security-by-deception paradigm. The solution proposes to inject intentionally crafted noise, which mimics authentication credentials, into the keyboard buffer, and wait for any potential GPU keyloggers to sniff it. A remote server triggers a detection alarm when it receives an authentication request using any of the rogue credentials. SMMDecoy uses 2 TEEs: SMM and Intel SGX [78].

The Main Aim and Contribution of the Thesis

The research contributions of the 4 research papers are a direct answer to the main aim of this thesis, because they clearly show that we can indeed *"use available commodity hardware TEEs as primitives to build systems which meet the specific security requirements for online user authentication developers and end-users"*.

However, we do acknowledge that these TEE-based solutions have their own set of limitations. In fact, the security of applications which use TEEs still completely depends on the hardware implementation of these TEEs, and on the trustworthiness of their manufacturers. This is problematic, because hardware can include backdoors that are difficult to detect, since it is much harder to test and reverse engineer than software, especially by independant third-parties. Furthermore, CPU-based TEEs have so far been vulnerable to certain side-channel attacks, and no proper mitigations have been so far proposed by their manufactureres. Finally, it is interesting to note that TEEs introduce new distributed trust models. The answer to whether end-users want to consolidate

trust among one or few entities, or rather distribute it among many, will depend less on technical considerations, and more on geo-political ones.

5.2 Applicability of TrustUI and SMMDecoy Solutions

This research envisions a world where the average user can have access to reasonable security-guarantees when they use the devices and services which have come to be a fundamental part of today's modern life, such as e-banking, and e-government services. Therefore, this thesis focuses on commodity devices which are available to the majority of end-users, instead of customized high-end military or enterprise-grade devices. In fact, both TrustUI and SMMDecoy are solutions which can be practically deployed within today's off-the shelf X86 devices. This is possible because they do not require any custom modifications to the underlying commodity hardware. Furthermore, these solutions are not exclusively relevant to online user authentication. They are rather applicable to all security critical user-level applications running on X86 commodity devices, such as online banking solutions, crypto-currency wallets and applications processing health-related data.

Finally, solutions such as SMMDecoy and TrustUI which can offer stronger integrity and confidentiality guarantees to applications running on end-point devices, are an important pre-requisite for enabling the adoption of truly distributed user applications [85], as well as a secure realization of the edge computing paradigm [116].

5.3 Limitations

While this thesis provides several contributions to the fields of online user authentication and commodity hardware TEEs, its output and results are limited by various constraints. These are mainly due to the limited time-frame allocated for this research and the constantly evolving threat scene and technology solutions.

Therefore, the thesis and its solutions are limited to the Intel X86 platforms with a focus on the PC market. The thesis studies neither the mobile nor embedded platforms. This choice consequently excludes a number of ARM and AMD specific commodity TEEs such as ARM TrustZone.

Furthermore, the thesis only focuses on addressing a subset of the gaps identified in the gap analysis performed in Chapter 3, namely the lack of trusted input and output channels. However, the lack of open provisioning and a multiplexed execution environment are not covered.

Additionally, the solutions presented in papers III and IV include the design of several cryptographic protocols. The correctness and security of the latter has only been informally argued [121]. However, it would be beneficial to formally verify these protocols using automated tools such as Maude-NPA [33].

5.4 Outlook on Future Work

A natural continuation of this thesis is to extend its research into other computing platforms, namely ARM IoT devices. This is an important research direction because the embedded systems' market is going through a profound transformation. As a matter of fact, most embedded systems were initially built under the assumption that they will be used as standalone devices, and there were few ways for the outside world to access their execution environments. However, as the cost of computing technologies decreases, embedded systems are quickly becoming more resource-rich. For instance, numerous manufacturers augment their IoT devices with telecommunications hardware and TCP/IP software networking stacks, which allow them to connect to the Internet. Furthermore, many of these IoT devices are expected to run security-sensitive workloads, namely within the financial and health sectors. However, not every IoT device is secure enough to handle such security-critical applications. Hardware TEEs are a promising approach to mitigate this risk.

Another interesting extension of this work is to focus on the problem of remote trusted computation as a fundamental enabler for the cloud computing paradigm. As a matter of fact, cloud databases for instance, offer greater performance, elasticity and storage advantages over traditional on-premise solutions. However, customers worry that the confidentiality and integrity of their data will be compromised by a malicious cloud provider, especially within a multi-tenant configuration. Trusted execution environments are an important security primitive which can mitigate such a risk, by isolating the user's workload from the cloud provider's software stack. Homomorphic Encryption (HE) schemes are another promising security primitive which enables computation over encrypted data. However they are not currently suitable for general-purpose computations because they suffer from very heavy execution overhead. Newer approaches aim to combine TEEs and HE to provide stronger security guarantees. This "combined model" uses HE to encrypt data, and TEEs to ensure the integrity of the database code at run time [31].

Papers

Paper I

Passwords Are Not Always Stronger on the Other Side of the Fence

I

Ijlal, Loutfi, Audun, Jøsang

Published in the proceedings for the 2015 NDSS Workshop on Usable Security (USEC'15)- San Diego, California.
DOI:10.14722/usec.2015.23005

Abstract

The username-password pair is still a prevalent form of online authentication. However, attacks that are leveraging weak password habits are on the rise. The main response of the security community on the ground is to invest more in educating users. Such an approach leads to believe that the long-held assumption stating that an ignorant user is the cause of an inadequate password behavior, still has many opponents. Although different research studies have presented other more likely reasons, practices are still perpetuating the same solution mindset of increasing end users' education. The behavior of users has not improved dramatically over the last decade despite all these efforts. Therefore, this research work explores the hypothesis that knowledge of good password habits is a necessary but not by itself a satisfactory requirement for a safe password behavior. This will be achieved by studying the password habits of the same people advocating for more end user education. To investigate this hypothesis, we conducted a survey targeting an audience of IT professionals with good knowledge about security. The survey results show that cognitive knowledge of password security does not always materialize into practical and secure password practices. The anticipated results would be that confronting IT professionals with their own password practices which fail to adhere to what they preach to end users, will motivate them to let go of their long-held assumptions that more education is the solution. This will further support the points made by other studies explaining the rationale behind the inadequate password habits of end users.

I.1 Introduction

Despite recent advances in user authentication methods, the most common mechanism used on the Internet today is still the username and password pair.

As a result, users are maintaining a large number of credentials for the many online services they use [38]. This is problematic because password-based user authentication brings serious usability challenges.

The inherent problem with data security is human fallibility. Even with the most advanced security systems in place, if there is a human component to that system, there will be vulnerabilities [45], [100]. Enforced policies are not stopping users from adopting inadequate password habits such as weak passwords, reused passwords and ignoring certificate warnings, just to cite a few. Indeed, security reports over the past decade show that attackers have been leveraging weak passwords in order to gain unauthorized access.

Hence, given the importance of password habits, a number of studies have been conducted by IT professionals with the aim of finding satisfactory solutions. As anticipated, the surveys converge in their results, demonstrating alarming percentages of weak passwords and inadequate password practices [17]. The main response by the security community to these threats against the human link has been users' education. Users are given instructions, advice and mandates as to how to protect themselves and their machines [46]. In this spirit, many IT professionals invest heavily in this regard. From companies including modules while onboarding their new employees, to large online companies posting sophisticated tips to its customers, to researchers investing in designing education modules that ought to be included in schools, the examples are numerous. This behavior presupposes that the idea attributing inadequate password habits to an ignorant and lazy end user, is still being held by IT professionals.

However, more recent research studies have been giving strong evidence for other more likely causes that explain the inadequate password habits of end users. For instance, solid arguments and studies indicate that the security advice received by users does not justify the cost they have to trade for it, making their decision to ignore it a rational one [38]. However, the reality on the ground indicates that IT professionals are still advocating for and implementing more education for end users. This leads to believe that while designing solutions, IT professionals fall back on the traditional explanation of inadequate password habits, which has long been attributed to the ignorance of end users. Several arguments can explain why IT professionals still hold on to their old assumptions despite strong recent evidence for their irrelevance: lack of awareness about such studies, a disagreement with their arguments, or a failure to internalize the implications of their findings while designing solutions. Independently of the reasons, the numbers indicate that password habits have not improved significantly since a decade [2], [117], while more educational efforts have been deployed. This has to change. In this study, we want to explore the hypothesis that knowledge of good password habits are a necessary but not by itself a satisfactory requirement for a safe password behavior. This will be achieved by studying the password habits of the same people who implement such solutions. It is anticipated that confronting IT professionals with their own password practices which fail to adhere to what they preach to end users through several educational channels, will motivate them let go of their long-held assumptions that more education is the solution. This will help further support the points

made by other studies explaining the rationale behind the inadequate password habits of end users.

To answer this question, we conducted our study with an audience that is exactly what the solution of more education strives for: IT Professionals. Hence, the metaphor of the other side of the fence in the title referring to the community of security and IT professionals who are assumed to be knowledgeable about good password practices and the risks associated with failure to comply with them. This stands in opposition to normal users who are assumed to be relatively ignorant about good password practices. Further, we want to investigate what would explain any highlighted differences in the password behavior of our audience. For this purpose, we also chose to remove a bias that, as per our literature review, is always embedded into the studies: the sensitivity level of different online services are regarded as a universally fixed value for all users. However, this study considers the sensitivity level of a service to be subjective. Indeed, it is a measure that should be evaluated from each user's perspective and usage profile for that specific service. For this reason, we included in our survey, questions that capture the user's perception about the sensitivity level of each service, as well as how that correlates or not with their behavior.

I.2 Related Work

I.2.1 Attacks

Users are typically seen as the weak link in any security chain. In the case of individually targeted attacks, it is usually easier to get sensitive information and passwords by social engineering than by direct assault or brute-force attacks against the system. The best way to get software onto any machine is to get the user to install it and human error is behind many of the most serious exploits [5], [38]. Further, over the past decade, many companies have reported breaches to their user accounts that were caused by brute force attacks against passwords. The latter exploit the weaknesses of the passwords chosen by human users. Early in 2013, the annual Data Breach Investigations Report published by Verizon stated that approximately 90% of successful breaches in 2012 analysed by Verizon started with a weak or default password, or a stolen and reused credential [42], [134]. The examples of successful attacks that have compromised the users' credentials are numerous:

- As per the analysis provided by Acunetix for the over 10,000 Hotmail passwords that were leaked online, 42% of them only contain lowercase alpha characters (a-z) and the majority of passwords were between 6-9 characters long [134].
- Usernames, e-mail addresses, password hashes, and password hints for adobe were leaked online. Inspired by this leak, a list of the worst passwords

of 2013 was published which shows passwords like password1lqq letmein, and 123456 are more common than one would think [13], [134].

Attacks against weak passwords are so flourishing that some emerging businesses were built upon this trend: pznedlist is a company specializing in monitoring the market of leaked credentials and reporting back to its subscribers when a positive hit is found on one of their accounts [117].

I.2.2 Surveys

In order to capture the behavior of online password usage directly from end users, a number of research groups relied on surveys as a means to collect feedback: during 2013, a survey was conducted in Norway by Norstat on behalf of EVRY. The sample size was 1012 respondents from Norway [99]. The findings of the survey were publicly shared in order to raise awareness about the current passwords behavior, as well as evangelize for better password habits. In 2012, the organization SCID conducted a consumer survey of password habits among consumers in the USA [17]. Further, SafeNet which conducted a global survey study on passwords, announced an equally alarming password behavior in all the surveyed geographical areas [108].

I.2.3 The Assumed Solution is Education

The above-mentioned surveys and their analysis have reached many similar conclusions: users are not practicing secure password techniques. After presenting their data, most studies suggest that more education is the solution [5]. As many notable institutions are putting forward the idea that investing in increasing educational efforts about security would eventually resolve the bad password behavior, other researchers are now taking this as a mantra and designing their research with an end goal to prepare education modules.

This research work acknowledges that Cormac Herley has presented a compelling case of why more education is not the answer. One possible plausible explanation given by the latter is the high competition for users' attention. Our work aims to further support this point within the IT professionals' community. This would be achieved by confronting them directly with their own password practices which fail to adhere to what they preach to end users through several educational channels, while they don't lack themselves such knowledge. Such a tactic would be anticipated to push them to make more conscious efforts in exploring different venues and implementing more efficient solutions, other than more education [45]. Indeed, research results should not stay confined within the borders of their papers. We should find efficient tactics to reach out not only to end users, but also to IT professional working in the ground: they are indeed in important link in the chain.

In this paper, we argue that we should challenge the assumption stating that investing in more education for the end users is the solution for their inadequate password behavior. Such an approach presupposing that the lack of

end users' knowledge about safe password practices is the reason cause driving their inadequate password behavior. Given the still unsatisfactory status of online password usage despite the education efforts deployed, our hypothesis hence, is that educating users is a necessary yet not a by itself a satisfactory reason to practicing safe online password behavior.

Specifically, this study investigates this hypothesis through a survey whose respondents were chosen to be the same people who design these educational solutions: IT security professionals. (Please refer to the survey methodology section for more details).

The services we enquired the respondents about were: Facebook, Gmail, LinkedIn, Twitter, Work/studies email, bank account, online gaming accounts and online storage services.

Furthermore, unlike the rest of the studies we have surveyed, this research accounts for the bias in identifying the sensitivity of any one service. Indeed, we let the respondents report their own perception the sensitivity level of each service. All possible associations/correlations between the reported sensitivity level, the reported password behavior and the profile of the users is then investigated.

For the sake of clarity, the below concepts are defined as follows, and should be understood and interpreted as such:

- Reported sensitivity level: the level of sensitivity a user judges a service to be to them.
- Reported password behavior: this is a measure induced from the individual answers the users provide about specific aspects of the password they use for each service (e.g.: length, character mix...etc.)
- Perceived password behavior: the judgment the users hold about how healthy their password behavior is.

Lastly, an emerging alternative to the passwords based web authentication is federated login. However, this mechanism raises serious privacy issues. One other goal of the study is to measure to which extent is this a concern for a person who is well informed about the issue [72].

Based on the above, the analysis of the results, coupled with other relevant studies, should enable us to get more insights into the following high-level questions:

- To which extent does cognitive knowledge about passwords behavior materialize into practical behavior?
- To which extent can we claim that education is a necessary yet not a satisfactory requirement for a safe online password behavior
- Are we making the right investment to resolve the password behavior challenge by increasing education channels about it?
- Is there a disparity between the perceived strength of passwords IT professionals use, and the strength we induce from their self-reported behavior?

- Does cognitive knowledge about how sensitive a service correlate with how well the password habits related to that account are?
- Is more granular advice about passwords' behavior the answer?
- Are people who perceive themselves as concerned with their online privacy, less willing to use federated login?
- What would trigger a user to become more aware about their password behavior?

I.3 Survey Methodology

I.3.1 Audience and Methods

Because this survey is aimed at a specific focus group, we did not open it for the large public. The target audience of the survey is IT professionals who are working in different industries.

We used a web-based version of the survey that we have designed with a premium account of Surveymonkey. The URL of the web-based survey was distributed via email.

We solicited the response of 112 people. A number of participants did not complete the survey or answered questions inconsistently. Their responses were removed from the data set, leaving 66 valid responses.

Further, it was our intention not to disclose to the audience that we are targeting IT professionals for this study. This would help minimize any kind of bias the respondent might develop while answering the survey questions.

The participants came from our mailing list of industry partners, as well graduate level students and above at the informatics department at the University of Oslo, Norway. Amongst the latter, we included a list of graduate level students who were taking an advanced security class during the semester the survey was conducted.

Thus, this work assumes that the audience sample chosen has sufficient knowledge about good password practice. However, the study did not employ any further mechanisms to account for any possible dishonesty from the respondents.

I.3.2 Design

The independent variables of the study are gender, age, ethnic background, country of residence, marital status, occupation, education level, number of online accounts of the user and IT Skills. The independent variables that aim at capturing features of the psychology of the user are: view of the world, introversion vs extroversion.

Our dependent variables can be classified in 3 categories:

- Parameters capturing perception: confidence in the strength of the used passwords, perceived sensitivity of an online service, and the reported concern about privacy.

- Parameters capturing the reported behavior: storage behavior of the password for each service, length, characters mix, memorability, reuse, usage of social login features, and usage of password managers.
  - For each one of the above mentioned parameters capturing the reported behavior, the survey results include results about the following online services: Facebook, Gmail, LinkedIn, Twitter, Work/studies email, bank account, online gaming accounts and online storage services
- Looking into the future: expressed willingness to improve password habits.

The placement of the questions was designed in a way that would optimize the accuracy of the responses by minimizing embedded biases [112], [127]. Respondents answer questions about their password habits for each online service prior to rating the sensitivity of these services. The goal is to avoid any minimize any intentional bias that would correlate the sensitivity level of a service with its corresponding password behavior when there is none. Further, the respondents rate their confidence level in their passwords' strength prior to determining their intent to improve their password behavior or not. Lastly, the respondents report their federated login usage before rating their privacy concern. This would help avoid exposing the correlation/association the study aims to measure.

I.4 Results and Analysis

For the purposes of this paper, and in line with the objectives outlined above, we will focus on presenting and analysing the below data:

- The profile of the respondents:
- Password usage
  - The behavior reported by users for each service.
  - The perception the respondents hold about the strength of their password behavior.
  - Association and correlation analysis between the reported/self-perceived behavior and perceived sensitivity of each service.
- Privacy
  - The reported privacy concern.
  - The reported privacy behavior expressed in the federated login scenario.
  - Association/correlation analysis between the self-reported and self-perceived behavior regarding privacy and federated login.

- Profile of people who express a willingness to reconsider their password behavior. Further, for the purposes of this paper, we did not include the data of Bank account behavior in the analysis. Most respondents referred to using 2 factor authentications for this service, and we will be discussing the impact of 2FA in the context of another research activity.

I.4.1 Respondents Profile

Demographic profile	**Psychological profile**	**Digital behavior profile**
Gender, Age, Ethnic background, Residence, Country, Civil status, Occupation, Education.	Mood, World view, Social login activity.	History with digital hacking, Number of online accounts.

Table I.1: Parameters of the respondents profile

The full distribution of the respondents' profile can be found attached in appendix 2.

I.4.2 Personal Usage

For each one of the 8 services studied(Facebook, Gmail, LinkedIn, Twitter, Work/studies email, Bank account, online storage, online video games), the respondents reported on their password behavior by answering questions enquiring for the below information(Questions 19 through 23 in the survey attached in appendix 1):

- Length of the password.
- Characters Mix in a password.
- Frequency of password changes.
- Usage of password recovery.
- Uniqueness of the password.

Further, the respondents reported on their password storage behavior.
From the above, we can note that:

- For each one of the 66 respondents: the study collected 41 individual pieces of information about their password usage.
- For each one of the 8 services: the study collected 494 piece of information about how our respondents interact with their password based authentication.

- Collectively, this makes up a total of 2970 pieces of information about how all of our respondents interact with all the services studied.

The analysis of the data was completed in two iterations. The focus of the first iteration was analysing the dataset as an aggregated set. This iteration is qualified as initial because aggregated data does not provide insight into how the observed behavior relates to neither the user's profile nor to the sensitivity level of the online services. Further, this first iteration provides little insight into the statistical relevance of the results. Indeed, judgement cannot be made about whether the highlighted correlations have any statistical significance. All these noted shortcomings of the aggregated analysis of. Nonetheless, aggregated data provides a great first stone in getting acquainted with the date set and in spotting patterns. The focus of the second phase is the study of more granular data, to the level of each respondent and each service.

The observations made in the initial iteration of data analysis are highlighted below:

1) Hacking Attacks don't Discriminate: 26 percent of respondents have been victims of hacking in the past. This number, naturally, does not account for people who have been victims to hacking without being aware of it. To put things into context, during 2014, 47 percent of Americans were hacked. This goes into showing that IT professionals are not immune to attacks.

2) IT Professionals Are Guilty: 11 percent of the respondents use non safe ways to store their passwords: digitally in the clear or on paper. While this might appear to be a small proportion of the respondents, the result should be read and interpreted in the context that the people surveyed are working in the IT field. These respondents are hence, likely to have responsibilities involving handling whole IT infrastructures and/or end user data. while there are views stating that writing down a password and physically storing in a secure location is a secure behavior, we disagree with this stand. As a matter of fact, a user would store a password physically if they estimate a high likelihood of forgetting it. This assumes that the user would be retrieve the piece of paper physically each time they do forget their password. We consider this behavior to increase the risks associated with exposing the password.

3) Character Mix: As per the table below, a considerable percentage of respondents do not always use a mix of characters when they are not forced to do so. Further, this behavior is more pronounced in services like Facebook and LinkedIn which don't enforce such policies, and that are increasingly being used as identity providers for other online services leveraging the federated authentication method.

4) Password Change Frequency: the respondents do not always exhibit a healthy pace of changing passwords when the policies do not enforce it.

5) What IT Professionals Are Best at: So for, for each one of the surveyed services, 97 percent of users reported password lengths greater than 6 characters.

Perception of the respondents about their password behavior (questions 24, 27 and 16):

	Yes, Always	Yes, Sometimes	No Never
Facebook	76	21	3
Gmail	81	18	2
LinkedIn	71	21	7
Online Video Games	67	30	3
Work/Studies	17	83	0
Twitter	3	24	74
Online Storage	83	17	0
Bank Account	82	14	5

Table I.2: Password character mix

	monthly	6 months	yearly	rarely	When asked
Facebook	2	10	19	40	29
Gmail	2	8	24	34	32
LinkedIn	0	7	16	46	32
Online Game	3	3	12	41	21
Work/Studies	12	9	12	26	41
Twitter	0	5	16	45	34
Online Storage	2	10	16	36	36
Bank Account	3	8	17	42	30

Table I.3: Frequenct of changing passwords

	Reused	Unique
Facebook	47	53
Gmail	42	58
LinkedIn	55	45
Online Games	60	40
Work/Studies	18	82
Twitter	42	58
Online Storage	45	55
Bank Account	17	83

Table I.4: Percentage of uniqueness of the password per service

- Sensitivity of the 8 services
- Perceived privacy concern
- Perceived confidence in the strength of the password used.

From the above we can note that:

- For each one of the 66 respondents: the study collected 9 individual pieces of information about their perceived password behavior.
- For each one of the 8 services: the study collected 132 pieces of information about how our respondents perceive their password usage behavior.
- Collectively, that is a total of 1056 pieces of information about how all of the respondents perceive their password interactions with all the services studied.

Similarly to the reported password behavior, the analysis first considers the results of the initial iteration of data analysis. Hence, the dataset is first analysed as aggregated set. The resulting observations are as follows:

	High Sensitivity	**Moderate Sensitivity**	**Low Sensitivity**
Facebook	36	50	14
Gmail	58	31	11
LinkedIn	22	47	31
Online Video Games	13	30	57
Work/Studies	85	15	0
Twitter	11	37	53
Online Storage	68	30	2
Bank Account	95	5	0

Table I.5: Perceived sensitivity of a service

72 percent of the respondents reported a higher than average level of trust in their behavior. (Average refers to the average score as defined by the scale of choices given to the users to choose from: from 0 to 3).

Expressed confidence level	Percentage
Totally Confident, 4	17
3	55
2	18
1	8
0	3

Table I.6: Expressed confidence in the password strength

6) Reported Behavior Vs. Perceived Sensitivity: From the aggregated analysis of the data above, we can already spot areas in which the password behavior of IT professional is less than satisfactory.

The first question the study would explore is whether the observed passwords behavior for each user correlate with the perceived level of sensitivity for each service. As the data is categorical, the Chi square test will be used [135].

The Chi square test will be performed against the null hypothesis. The latter assumes that two categorical variables are completely independent. The Significance value the set for this study is 0.05.

The study aims not only to explore the statistical associations between the variables, but also to determine the specific pairs of combinations that have yielded the most significant results. To achieve this goal, the analysis of the data also comprises computations of the residual deviation for each pair. The significance range used is (-2,2) [135].The test is first run on all the services combined. The results are as follows:

The residual deviation significance does not always indicate a practically significant association. This, the analysis cross-compares the initial conclusions drawn from the residual deviations, to visualizations of the raw data before making any final conclusions. Such an approach is considered a good practice, because the Chi Square is a test of statistical association and not of linear correlation. The focus of this study is linear associations. The conclusions made as well as the interpretation of their implication are as follows:

- Character length is the feature that the respondents have shown the most ability to materialize from cognitive knowledge into practice. Indeed, more sensitive services exhibited lengthier passwords.
- An association exists between the character mix and the sensitivity level of services. There are strong evidences of a linear correlation in the data.
- Further analysis is needed for the reuse feature: the initial observations are not conclusive and did not initially reveal any significant linear correlations.
- The strong association between the pairs suggest that there is a consistency in the behavior of users who perform well. Specific pairs of combinations require further analysis.

The correlation value measured between different pairs on the reported password behavior yielded p values less than 0.001:

- This was an interesting result for us. Although it did not express what kind of correlation exists between these parameters, it made us think about investigating the behavior of single respondent across all the matrix parameters and see if we can spot consistent behavior of safe/unsafe password behavior.

	Sensitivity level	**Interpretations comments**
Password Length	0.007	The null hypothesis does not hold. Strong derivative chi square between high sensitivity an increased password length
Password Character Mix	Less than 0.01	The null hypothesis does not hold Residual values of values of yes, always and yes, sometimes ones which are higher The residual values observation is confirmed by visualization of the data.
Password Change	P value could not be computed because of a high number of small count cells.	No conclusion about the association from the p value The residual margin analysis significance between(low sensitivity, only when prompted to do it) and (moderate sensitivity, every 6 months) Visualization of data are in sync with the residual margin observations.
Use of password recovery option	0,06	Null hypothesis holds No significant residual margin values observed
Reuse	Less than 0.01	The hypothesis does not hold Residual margin values were significant for all pairs of value Closer look at the date needs to be done to infer the most relevant pairs for our study.

Table I.7: P test Chi Square of the measured password behavior vs. the perceived sensitivity level

At this point, we can already observe that our respondents are not exhibiting a strong correlation between the perceived sensitivity level and the password behavior. Suggesting that indeed, there is a disconnect between cognitive knowledge and practical behavior.

After having completed this round of analysis, we wanted to get a DEEPER understanding of our data, by looking at interesting combinations of responses that are hinted by the p values for Chi Square analysis.

7) Respondents with an Across-the-board Satisfactory Password Behavior: Indeed, and as hinted and highlighted in our previous conclusions, there was a strong suggested association between the parameters capturing the reported password behavior of users. One particular subset we focused on was the one of the respondents who exhibited satisfactory behavior across all metrics. We made interesting observations about this subgroup:

- 69 percent of passwords across all services satisfy all the parameters needed for a safe password at once.

That is a mere 12 percent of the whole of the passwords. The respondents of this subset exhibited the below behavior:

- 100 percent of the Respondents with a satisfactory password and who don't think they should improve their future password behavior have expressed a complete level of confidence in their behavior (4).
- 97 percent of the Respondents with a satisfactory password behavior and who are expressing the intention to improve their password habits in the future, expressed a level 3 confidence level in their password strength.
- 3 percent of the Respondents with a satisfactory password behavior and who are expressing the intention to improve their password habits in the future expressed a level 2 confidence level in their password strength.

There was one more observation which made us zoom more into this group of respondents and study a subset within it:

The Chi Square values revealed a tight association between 3 of the 5 parameters measuring the password behavior. Amongst the 69 passwords, more than 50 percent of the passwords mapped to 30 percent of the respondents that are part of this subset A.

Amongst A, the respondents who exhibited safe online password behavior across at least 50 percent of the services were further studied. These are the users exhibiting the most optimal password usage across all services:

- Interestingly, 75 percent of these respondents answered by yes to the question of whether or not they intent to improve their password habits in the future.

- 0 percent expressed a total confidence in their password strength.
- 100 percent expressed a level 3 confidence in their password strength.

8) Confidence Vs. Willingness to Change: The P value of the Chi Square test revealed a strong correlation between the reported confidence level at the beginning of the survey and the expressed intent to improve one's password behavior at the end of the survey. Indeed, the P test value of 0.042 means that the null hypothesis does not hold. A second look at the visualized dataset in light of this observation confirms it.

- 9 percent of the respondents who have expressed a total confidence in their password strength have expressed no intent to improve their password habits.
- 67 percent of the respondents who have expressed a level 3 confidence in their passwords behavior have expressed no intent to improve their password habits.

9) Perceived Privacy and Federated Login: Ever since their emergence, federated login mechanisms have sparked a lot of controversy in the security community. On one hand, they introduced a convenient way for end users to authenticate and alleviate their identity sprawl problem. On the other, they raised many privacy issues. In the context of this paper, we will not discuss the other security requirements federated login put at risk [72].

One of the assumptions put forward to explain the growing adoption of federated login is the users' ignorance of its related privacy issues. This assumption pre-supposes, once more, that the lack of information is behind this behavior, and that in the presence of such knowledge, people will choose their privacy over convenience and usability.

The aggregated analysis showed that a significant number of respondents rely on federated login. The second iteration analysis did not find any significant statistical correlation between the expressed privacy concern, and the user's federated login behavior.

The above suggests that our respondents did not translate their expressed privacy concern into a corresponding usage pattern of federated login.

I.5 Conclusions and Future Work

The above data and its analysis provide significant insights into the password habits of IT professionals. Although they possess enough cognitive knowledge to be fully aware of what constitutes an adequate password behavior, they fail to materialize it into practical habits in many instances. Evidently, the data analysis revealed no statistically significant correlation between the reported password behavior and the reported sensitivity level of the services. This strongly suggests that the ever more granular advice users get about adopting varying

password behavior for each level of sensitivity, is not very efficient. Indeed, although cognitively convincing, the desired implications of such advice are not reflected in practice. This particular finding is in line with what Cormac Herley has highlighted in the more is not the answer paper [45]. More granular security advice is likely to be ignored due to other competing messages for users' attention.

While one might argue that IT professionals are scoring better than the general public in some metrics, the fact and matter is, the studied data is far from being satisfactory. Surely, we would be expecting a better return on investment for the educational efforts invested in end users. Furthermore, one is to remember that the respondents of this study do not only represent the profile of an ideal end user, but that they are also IT professionals within their organizations. Hence, they are likely to be handling security processes, or at least be holding privileged accounts within their organizations.

The aim of this research study was to explore the hypothesis that education is a necessary yet not by itself a satisfactory condition for ensuring a safe password behavior. The data of the survey supports the hypothesis. The data revealed interesting observations about the subset of respondents who exhibit satisfactory password habits for all services. As future work, we look forward to building upon this work and further investigating the characteristics which set apart this subset.

Further, the data revealed a significant proportion of respondents who had a shift in the expressed willingness to review their password habits between the beginning and the end of the survey. These respondents are also noted to have expressed a level 3 confidence in their password strength.

As a possible future work, we want to explore the following hypothesis: a healthy level of doubt in the strength of one's password habits and an expressed growing mind mentality, might yield a better correlation between cognitive knowledge and password practices. Being absolutely sure of the adequacy of your password habits, might make you more vulnerable, or at best, will not make you more secure.

This study notes that over the last decade, there have been some voices presenting strong explanations for users' inadequate password habits. However, the desired implications of these studies have not materialized yet in the way IT professionals are designing solutions. The old mindset of more education is indeed still prevalent [45]. This study results are meant, hence, to confront IT professionals directly with their own password practices which fail to adhere to what they preach to end users through several educational channels. We would anticipate such a straightforward approach to fasten the mind shift of IT professionals. If their own solutions are failing them, then they would have more reasons to let go of their long held biases, and take mindful steps towards embracing the real reasons explaining end users' inadequate password habits. Implementing novel solutions in line with these studies would be the ultimate outcome.

Great insights have emerged as a result of significant research efforts targeted at resolving the online authentication challenge. However, given the urgency

of the matter, we should strive to make the findings of these research studies relatable to the relevant people. This work subscribes to this philosophy. Indeed, this study targets in a straightforward manner IT professionals, gets them involved, and discusses findings that are very relatable to their concerns. IT professionals are a critical link in the security chain. The results of this study would be anticipated to increase the likelihood of IT professionals to let go of their old mindset, and fasten the pace at which they will start deploying new more appropriate solutions for their end users.

Lastly, as an IT community, we should open up to other disciplines, obtain a deeper understanding of the motivating factors for users' behavior, and become humbler in our perception of human capabilities. Knowing the right thing to do, does not necessarily mean that we will do the right thing. We must learn to practice what we preach.

Acknowledgments

We would like to thank the respondents for their time. We also thank our USEC 2015 anonymous reviewers, as well as our shepherd, Dr. Jens Grossklags for their valuable comments and guidance.

We, the authors, are funded by the informatics department of the University of Oslo, as well as COINS Research School of Computer and Information Security.

I.6 Appendix 1: Overview of the Survey Questions

For questions 19–24, answers were required for each one of the below online services:

- Facebook
- Gmail/Google+/YouTube
- LinkedIn
- Online Video Games
- Work Studies Account
- Twitter.

Below are the answers collected from the questions which define the profile of our respondents:

Q1	Are you male or female?
Q2	What is your age?
Q3	What is your ethnic background?
Q4	Where are you currently living?
Q5	Which of the following best describes your current status?
Q6	Which of the following best describes your current occupation?
Q7	What is the highest degree you have received/working towards completing?
Q8	How would you describe your mood today?
Q9	How strongly do you agree/disagree with the below statement People are inherently bad.
Q10	How Would you describe yourself?
Q11	How often do you log into social media networks (e.g. Facebook, Google+, etc.)?
Q12	How would you rate your computer Skills?
Q13	Have you ever been the victim of online theft...(stolen password, unauthorized transactions in your name)
Q14	How many web accounts do you currently have
Q15	In which year did you get your first email address?
Q16	What is the typical length of the password you use for the below?
Q17	How confident are you in the strength of the passwords you use to access your online accounts?
Q18	How do you store passwords?
Q19	What is the typical length of the password you use for the below?
Q20	For each of the below, do you by your own choice use mixes of different character types?
Q21	For each of the below, do you Ever change your password because you decide you do it?
Q22	For each of the below, how often do you forget your password then use the recovery option?
Q23	For each of the below, do you use a Unique password or is it reused with another account?
Q24	How sensitive do you consider the below online service are to you?
Q25	Do you use an online password Manager to store/manage your credentials?
Q26	Social login is the option to use your profile from one service to register/login into another online service
Q27	Do you worry about the privacy of your online presence?
Q28	Do you think you should improve your password habits?
Q29	Do you have further comments about your web password and online identities that you would like to share?

Table I.8: Overview of the survey questions

Answer Options for gender	**Response Percent**
Male	84.8%
Female	15.2%
17 or younger	0.0%
18-20	1.5%
21-29	43.9%
30-39	22.7%
40-49	13.6%
50-59	13.6%
60 or older	4.5%
Primary School	0.0%
High school degree or equivalent	1.5%
Professional training	0.0%
Bachelor's degree or equivalent	12.1%
Master's degree or equivalent	68.2%
PhD	18.2%
1 online account	0.0%
2-5 online accounts	22.7%
6-10 online accounts	16.7%
11-20 online accounts	9.1%
21-50 online accounts	24.2%
>50 online accounts	27.3%

Table I.9: The respondents' profile

Paper II

FIDO Trust Requirements

Ijlal, Loutfi and Audun, Jøsang

Publsihed in the proceeding for the 20th Nordic
Conference on Secure IT Systems (NordSec'15)
139–155
DOI:10.1000/182

Abstract

FIDO (Fast Identity Online) is a new online identity management architecture, developed and promoted by a large industry consortium. Its goal is to simplify and strengthen online user authentication by relying on local device user authentication. Another goal is to finally put passwords to rest. This solution requires strong trust between players and components in the architecture. These aspects have received little attention from the FIDO consortium. The aim of this paper is to analyse the trust requirements for FIDO, and assess the cost of establishing the required trust.

II.1 Introduction

Nowadays, consuming services online has become an exponentially growing trend within different industries. A robust online identity management solution to these services is a critical requirement for establishing trust between end users and service providers (SP). Within this context, Identity management is defined as the process of representing and recognizing entities as digital identities in computer networks. It includes many security constructs such as authentication, authorization and access control. Authentication, which is the focus of this paper, is an integral part of identity management, as it serves to verify claims about holding specific identities[114]. In today's online ecosystem, the average online end user holds multiple identities with multiple online SPs[38]. Pass- word based authentication methods are by far the most prevalent deployed mechanism. However, their shortcomings are well documented and recognized. As a matter of fact, the rapid growth in the number of online services based on this model now results in the users being overloaded with identifiers and credentials that they need to manage. As a result, end users resort to adopting non secure password habits[18][13]. On the other hand, SPs also fail in protecting the users credentials they have stored on their servers. Early in 2013, the annual Data Breach Investigations Report published by Verizon, stated that approximately 90

percent of successful breaches in 2012 analysed by Verizon started with a weak or default password, or a stolen and reused credential[135] [18]. For this reason, new identity management models are being proposed and implemented. Their goal is to either minimize the reliance on passwords, or eradicate passwords all together. Identity federation has marked the last decade as being the most prominent new authentication mechanism. However, Fast Identity Online (FIDO), which started in 2012 as an industry consortium, is currently gaining exponential momentum in the market with big identity management players, that can influence the reality of online authentication moving forward [41]. For instance, Microsoft is adopting FIDO2.0 protocols for its upcoming windows10 release [138], Google has already enabled FIDO authentication on its email services, and Samsung has equipped its Samsung galaxy S6 with FIDO compliant hardware [104][40]. FIDO, indeed, holds the promise to solve many of the problems inherent to password based authentication. However, it relies on a completely new architecture which introduces new trust requirements between different players and components. Identifying, understanding and quantifying these trust requirements are crucial in making us aware about what could go wrong while using FIDO, and if these are risks we are willing to incur.

II.2 Background and Related Work

FIDO came as a response to the shortcomings of the currently implemented identity management (IdM) models. It also build on many of their concepts. Hence, the first part of this section will focus on presenting the architectures of these models. The identity management architectures to be discussed are as follows:

1. Online isolated identity management.
2. Federated isolated identity management.
3. Local device identity management.
4. Fast Identity Online (FIDO).

II.2.1 Isolated Identity Management

In the simplest case where a set of users access a single SP, the traditional approach is to let users identify themselves through unique identifiers, and authenticate themselves using security credentials such as passwords. What we have here is an isolated IdM model because each identifier that a user possesses can only be used for one isolated service. This model, which is used for all types of access to online services and resources, as well as for digital rights management, is relatively simple for SPs [67]. In this architecture, online SPs act as both a credential provider and an identifier provider to their clients. They control the name space for a specific service domain, and allocate identifiers to users. A user

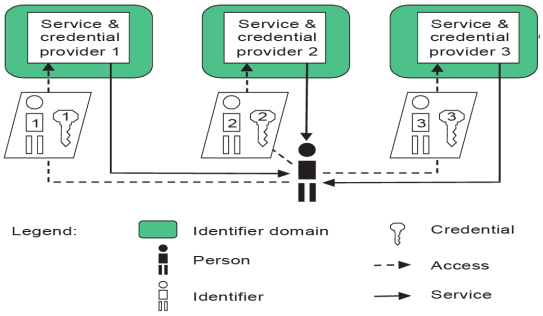

Figure II.1: Isolated online identity management

gets separate unique identifiers from each service/identifier provider he transacts with. In addition, each user will have separate credentials, such as passwords associated with each of their identifiers. This is illustrated in Fig. II.1.

This approach might provide simple IdM from the SP point of view, but is problematic for users, as the number of SPs that they transact with increases. Users are faced with the daunting task of properly managing a large number of passwords. This has led to very inappropriate password behavior which compromises the security of end users (weak passwords, repeated passwords) [67] [45] [100].

Because of all the above, academia and industry alike have been working on alleviating the inherent problems of password based methods. One such approach evolved from the idea that we need to move away from a siloed isolated model, into a federated model. The latter would allow users to minimize the number of credentials they need to manage. Thanks to the ever more increasing adoption of social networks, federated IdM solutions have strongly impacted the commercial end user online IdM scene over the last decade, after having been confined to enterprise perimeters.

II.2.2 Federated Online Identity Management Solution

One of the purposes of identity federation is to address the type of inefficiencies described above. Identity federation can be defined in this context, as the set of agreements, standards and technologies that enable a group of SPs to recognize user identifiers and entitlements from other SPs within the group. The basic idea is to link different identifiers, and thereby their associated identities, owned by the same user across multiple SPs. Then, allow the user to authentication himself with a single identifier to one of the service providers, and thereby be considered identified and authenticated by all the other SPs as well. The isolated

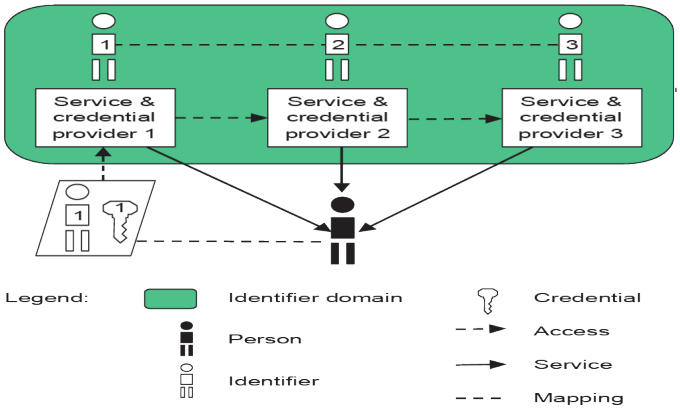

Figure II.2: Federated online identity management

identifier domains within a federated group becomes a single federated identifier domain [67]. This approach is illustrated in Fig. II.2.

Identity federation comes in many variations but is typically based on the SAML1 standard which is implemented in applications such as Shibboleth,and FacebookConnect. However, Identity federation does not fundamentally solve the problem of identity overload. There will always be different federation domains, because not all SPs will merge their respective user identity domains into a single federated domain [81].

II.2.3 Local Owner Device Identity Management

While the two previous IdM models focused on online authentication, we will shift our focus now into local owner device authentication (referred to in the rest of the paper as user device authentication). Over the last decade, networked computing devices have become a common commodity. The global smartphone audience surpassed the 1,75 billion mark in 2014, while the number of computers is still increasing despite the saturation of many developed markets. Furthermore, we are witnessing the emergence of Internet of things where more and more devices are entering the digital world [81]. This trend has been facilitated by the decreasing costs of production of these devices. Numerous interesting aspects of computing have been challenged by this trend. In the context of this paper, the question that is most relevant to us is: How can end users identify themselves to their local devices?

The answer to this question has long been simple: username, password pair. However, numerous alternate authentication mechanisms such as biometrics (fingerprint, iris) and TPMs are finding their place in user device authentication.

This is mainly due to two factors: the advancements made in the implementation of these new mechanisms. Also, its user acceptance is largely thanks to the fact that user device authentication happens locally, and does not require the

disclosure of any sensitive information (e.g.: biometrics) to third party SPs. The market currently shows a growing trend in which users have accepted and embraced alternate authentication mechanisms for their local device authentication [124].

II.2.4 Fast Identity Online: FIDO

FIDO Philosophy: In July 2012, the FIDO alliance became a 501(c)6 non-profit organization. Its core idea relies on the following: leveraging local user device authentication for online user authentication. In a *Non FIDO scenario*, Bob would swipe his fingerprint into his mobile phone so as to authenticate himself to the device as his owner. He would then select the mobile application of the SP of his interest (e.g.: mybank), and use the corresponding online authentication mechanism (username and password) in order to start his transactions with his bank. In this scenario, Bob's local user (owner) device authentication, and his online authentication are two separate processes.

On the other hand, in a *FIDO scenario*, Bob would have a different experience. In order to start his online transactions with his online bank, he would not be required to enter any username, password pair. Bob would only have to swipe his finger print, for example, into his smartphone, while being on the authentication page on the mobile application for his online bank. What is even more interesting in this new FIDO enabled scenario, is that from Bob's perspective, the authentication experience would be the same independently of the SP he is interacting with. If Bob wants to connect into his Facebook account, all he would have to do is open the Facebook mobile application and swipe his finger again into his smartphone. In this scenario, Bob's local user (owner) device authentication has become part of his online authentication ,as illustrated in Fig. II.3.(refer to section II.2.4 for more details about the FIDO architecture).

First Mile Authentication: Figure II.3 above introduces the concept of *hardware authentication* (hardware and authenticator will be used interchangeably). It is expected that users will acquire FIDO Authenticators in various ways: they purchase a new system that comes with an embedded FIDO Authenticator capability; they purchase a device with an embedded FIDO Authenticator, or they are given a FIDO Authenticator by their employer or some other institution such as their bank. After receiving a FIDO Authenticator, the user must go through an authenticator-specific enrolment process. For example, the user must register their fingerprint(s) with the authenticator. Once enrolment is complete, the FIDO Authenticator is ready for registration with FIDO enabled online services and websites. Every time Bob wants to authenticate to an online SP, his *first mile* would be to authenticate himself to his FIDO device [36].

Second Mile Authentication: The first mile authentication required the interaction of Bob with his FIDO device/authenticator. However, the second mile authentication will be transparent to Bob, not requiring him to further interact with his device nor with the SP. After the completion of the first mile authentication (Bob authenticating to the his FIDO device), the authenticator

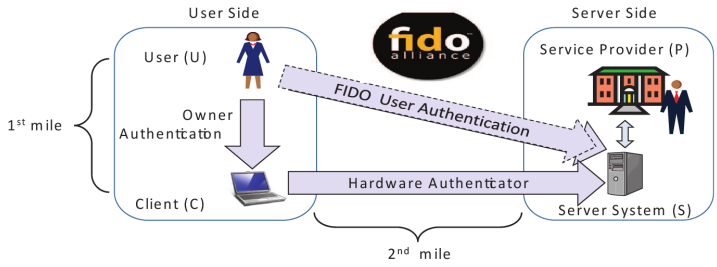

Figure II.3: FIDO authentication: high level overview

can attest to its identity to online SPs, on behalf of Bob. This is achieved by relying on asymmetric public key cryptographic exchange.

FIDO User Authentication: Finally, after the completion of the second mile authentication, the SP will authenticate Bob to his service, if the key exchange was successful. Figure II.5 below is a high level summary of the both the FIDO registration and authentication message flow.

The Concept of FIDO Authenticator: At the heart of FIDO protocols, lies the concept of the authenticator. A FIDO Authenticator is a secure entity, connected to or housed within FIDO user devices, that can create key material associated to a SP. The key can then be used to participate in FIDO strong authentication protocols. For example, the FIDO Authenticator can provide a response to a cryptographic challenge using the key material. A FIDO authenticator can be implemented within the user space or completely separate from it. It is up to every SP to decide on the level of risk it wants to incur, and hence, the type of authenticators it allows its end users to use [36]. Some examples of authenticators are: a fingerprint sensor built into a mobile device, a PIN authenticator implemented inside a secure element, a USB token with built-in user presence verification, a voice or face verification technology built into a device.

FIDO authenticators' implementation are very varied. Each SP has the choice to accept users with using a specific type of authenticators (e.g.: only accept fingerprint readers that store the finger print and cryptographic keys in a secure element, and that are manufactured by hardware manufacturer xyz). Hence, in order to meet the goal of simplifying the integration of trusted authentication capabilities, a FIDO Authenticator will be able to attest to its particular type (e.g., biometric) and capabilities (e.g., supported crypto algorithms), as well as to its provenance. This provides SPs with a high degree of confidence that the user being authenticated is indeed the user that originally registered with the

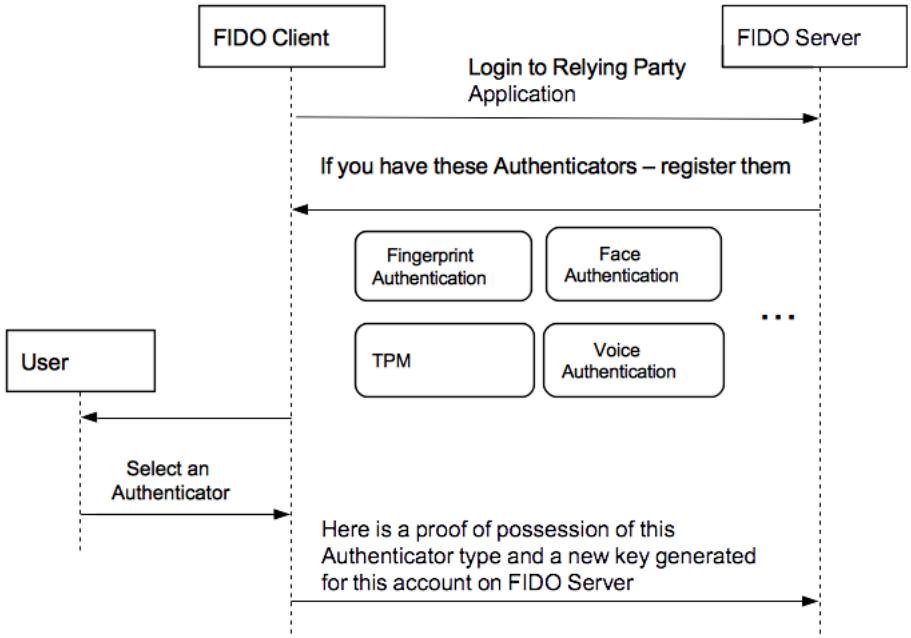

Figure II.4: FIDO registration [37]

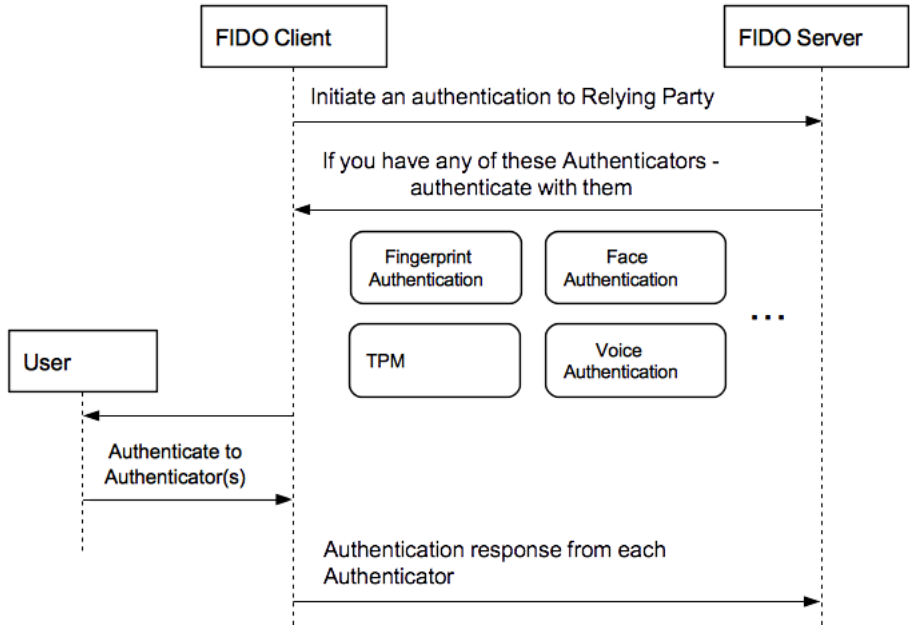

Figure II.5: FIDO registration message flow [37]

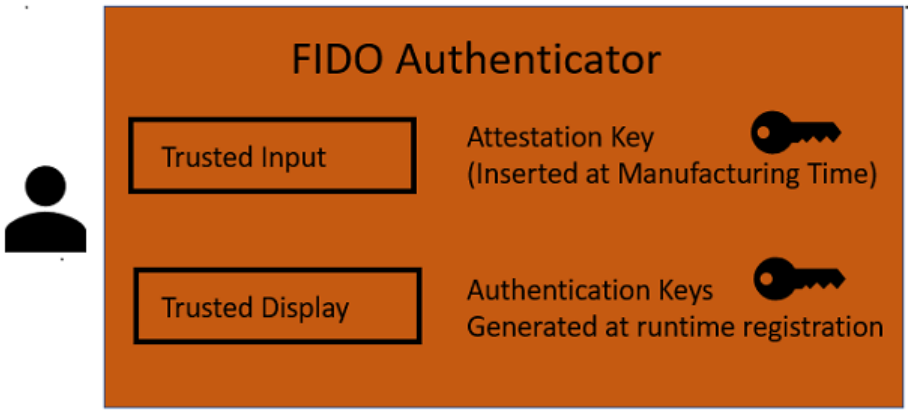

Figure II.6: FIDO authenticator [37]

site [36].

So as to be able to perform the cryptographic operations defined in FIDO protocols, a FIDO authenticator needs to have some type of attestation mechanism. This is achieved through the key structure as illustrated in Figure II.6:

The attestation key is used by an authenticator in order to attest to its type and provenance to the SP. FIDO Authenticators are created with attestation private keys used to create the signatures. FIDO SPs validate the signature using the corresponding authenticator's attestation public key certificate located in the authenticator metadata. The metadata holding attestation certificates is shared with FIDO Servers out of band [36].

Authentication keys: Every time a user wants to register with a new SP, the authenticator generates a new private/public key pair called authentication key pair. This key pair is unique for every combination of (end user, SP, authenticator). The authenticator stores the authentication private key. The corresponding authentication public key is communicated to the SP. After the successful completion of the registration process, the authenticator performs a challenge response exchange with the SP using the authentication keys, every time a new authentication in required.

FIDO Architecture: Figure II.7 pulls all of the above FIDO concepts together, by showing a detailed architecture of both the client side and the server side of FIDO protocols.

II.2.5 Online Identity Management Trust Requirements

Now that we have introduced all the main IdM architectures, we will look at the extent to which their trust requirements have been studied and analysed.

the concept of trust: Trust is typically interpreted as a subjective belief in the reliability, honesty and security of an entity on which we depend for our welfare. In online Environments, we depend on a wide spectrum of devices,

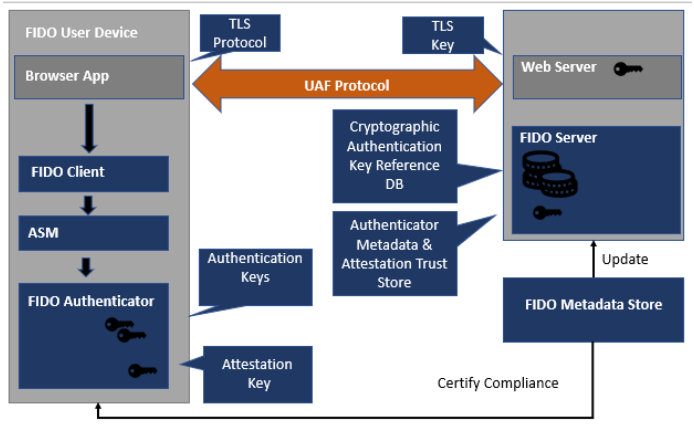

Figure II.7: FIDO architecture [37]

ranging from computer hardware, software and data, to people and organizations. A security solution always assumes that certain entities function according to specific policies. To trust is precisely to make this sort of assumptions, so a trusted entity is the same as an entity that is assumed to function according to policy. A consequence of this is that a trusted component of a system must work correctly in order for the security of that system to hold, meaning that when a trusted component fails, then the systems and applications that depend on it can no longer be considered secure. An often cited articulation of this principle is: a trusted system or component is one that can break your security policy (which happens when the trusted system fails) [65][128].

The transfer of the social constructs of identity and trust into digital and computational concepts help in implementing large scale online markets and communities, and also plays an important role in the converging mobile and Internet environments. IdM (denoted IdM hereafter) is about recognizing and verifying the correctness of identities in online environments. Trust management becomes a component of IdM whenever different parties rely on each other for identity provision and authentication[65].

II.3 Trust Requirements Analysis

While the focus of this paper is on FIDO, we believe that an accurate understanding of its trust requirements cannot be achieved by analysing them in isolation. They need to be compared and evaluated against the trust requirements of the other existing IdM solutions: isolated, federated. Hereby, we will be presenting in this section the trust requirements of all of the previously presently online IdM solutions, with a special focus on FIDO. Subsequently, we will discuss the implications of the varying trust requirements inherent to each solution.

II.3.1 Trust Requirements of Isolated Online Identity Management

In this architecture, the trust requirements between users and SPs are well understood in the form of specific security and privacy assumptions. In addition, the industry has had several decades of experience with this model, and users are familiar with it.

Trust complexity is greatly simplified when the same entity acts as identifier provider, credentials provider and SP. Under these conditions, the client and SP only need to trust each other for a small set of purposes [67]. Indeed, most of the trust requirements fall under one of the two categories:

Client Trust in Service Providers:

1. The service provider has the expected identity.
2. The service provider protects client privacy.
3. The service provider has implemented satisfactory user registration procedures and authentication mechanisms (from the clients perspective).

Service Provider Trust in Client:

1. The client handles their authentication credentials with adequate care.

II.3.2 Trust Requirements of Federated Online Identity Management

While identity federation is aimed at simplifying the user experience, it increases trust complexity both for the SPs and their clients. These trust requirements can be classified as follows [67]:

1. Trust between Federated Service.
    a) Service access by assertions between SPs on behalf of users will only take place when legitimately requested by the client.
2. Trust in the Identity Mapping.
    a) The mapping of identities between service providers is correct.
3. Client Trust in Service Providers.
    a) The service provider adheres to the accepted policy for correlating personal data about the same client from other service providers.

II.3.3 Trust Requirements of FIDO

Given that FIDO is mainly an industry led initiative, and that its final protocols have only been released recently, we found no existing studies to its trust requirements. As far as our knowledge goes, this study will be the first one to provide a basis for understanding FIDO trust issues that are seldom talked about. They will help stakeholders who are joining FIDO better understand what can go wrong with FIDO in real life. It will also help them understand the hidden trust agreements they implicitly consent to, once they consent to using FIDO. We arrived at these results by thoroughly studying FIDO protocols [36] and its security reference[124] [37], then analysing them against the claims put forward by the FIDO consortium in its official release[36].

In order to provide a simple view for our FIDO trust requirements results, we classified every single trust requirement under one of these categories:

1. Trust in FIDO consortium.
2. Trust in service providers.
3. Trust in hardware manufacturer.
4. Trust local device computing platform.
5. Trust in end users.
6. Trust in FIDO protocols.

II.3.4 Trust in FIDO consortium

The FIDO consortium presents itself as an enabler for the promotion and education about FIDO open protocols. However, it is responsible for certifying the hardware manufacturers. It also ensures that the right metadata and root certificates are being trusted. These tasks are detrimental to the overall trust of FIDO as an IdM solution.

In this context, certification refers to the FIDO program that allows members and non-members to measure compliance and ensure interoperability among products and services that support FIDO specifications. Companies completing certification may display the FIDO Certified logo to demonstrate to consumers, customers and partners that they have created a high quality, interoperable FIDO implementation that is known to work with other FIDO implementations. In the case of a FIDO authenticator that is certified, it will be characterized by a set of metadata information. This metadata is associated with an AAID (Authenticator Attestation ID) and available from the FIDO Alliance. FIDO Servers are expected to have access to up-to-date metadata to be able to interact with a given authenticator. AAID is defined as a unique identifier assigned to a model, class or batch of FIDO Authenticators that all share the same characteristics, and which a SP can use to look up an Attestation Public Key

and Authenticator Metadata for the device. Finally, FIDO alliance needs to explicitly trust a root certificate to which all authenticator attestation Certificates chain to, and that will be relied on by the SPs to assert to the validity of the provided attestation certificates, and hence the metadata associated with it [36] [124] [37]. The above process already introduces a number of trust requirements:

T1: *Trust that the FIDO consortium has identified the right set of metadata characteristics that are sufficient to identifying authenticators in ways that are meaningful to SPs to accept or reject them.*

T2: *Trust that the certification is still meaningful throughout the time it is valid.*

T3: *Trust that the FIDO consortium is able to detect and report authenticators breaching the metadata characteristics declared in their certification process, and update the metadata store accordingly.*

T4: *Trust in the validity of the FIDO PKI used (root certificate, attestation certificates).*

II.3.5 Trust in Service providers

In the previous section, we have discussed how FIDO consortium is responsible for updating the content of the metadata. In this section, we will investigate how SPs make use of the attestation PKI as well as the authenticators metadata store. Indeed, SPs have the right to enforce policies about the type of authenticators they want their users to use while consuming their services. It achieves this through the use of an authentication policy. The latter is defined as a JSON data structure that allows a SP to communicate to a FIDO Client the capabilities or specific authenticators that are allowed or disallowed for use in a given operation. The client then responds with an attestation certificate serving as a claim that it possesses an authenticator that is compliant with the policy. It is then the responsibility of the SP to ensure how genuine this claim is, by using information in his metadata store. The latter should always be in sync with the central FIDO metadata store that is supposed to hold the latest updates about all authenticators [36] [124] [37]. The above process introduces two new trust requirements:

T5: *Trust that the SP is able to correctly assess the risk level associated with the usage of his service by all his users.*

The above risk assessment directly influences the service provider's choice of the allowed authenticators, and hence how the authentication policy is defined. If the risk is estimated too high or too low, this will steer away consumers.

T6: *Trust that the SP establishes the appropriate network connection while updating the metadata store.*

II.3.6 Trust in Hardware Manufacturers

The architecture of FIDO has brought hardware manufacturers into prominence in the IdM trust requirements discussion. In all previous IdM models, hardware manufacturers did not incur any significant trust requirements nor

liability. However, in the context of FIDO, hardware manufacturers have central responsibilities. For instance:

1. Manufacturing the hardware that is used as an authenticator for FIDO. The user owner local device authentication is the first step in the new FIDO authentication process.

2. Providing cryptographic evidence to the SP attesting to the type and provenance of the authenticator.

Furthermore, a main tenant of the FIDO privacy-by-design premise is hardware related. On one hand, as discussed in TR4, TR5, and TR6, FIDO protocols provide a mechanism for SPs to get information about the type and provenance of the authenticator being used. This is mainly achieved through the attestation certificate. However, this poses the risk of exposing the privacy of users, in the case that their authenticators can be identified individually by SPs. In order to resolve this issue, FIDO protocols proclaim the following requirement:

UAF authenticators can only be identified by their attestation certificates on a production batch-level or on manufacturer- and device model-level. They cannot be identified individually. The UAF specifications require implementers to ship UAF authenticators with the same attestation certificate and private key in batches of 100,000 or more in order to provide unlikability [37].

All of these mechanisms introduce the following trust requirements:

T7: *Trust that hardware providers will not unintentionally break the unlikability property.*

Due to the recent privacy scandals in which governmental states have been involved, as well as the geopolitical changes the world is witnessing, the IT market tends to be compartmentalized. Indeed, there are a number of states that are wary of buying and using technology of its non-political allies (Russia, China, Korea, USA). Especially at the beginning of FIDO adoption, this might lead the attestation certificate revealing more information than it ought to. This scenario can also happen in the case where specific hardware manufacturers are known to traditionally provide for certain industries (e.g.: military, banking). This risk becomes ever more relevant, with the recent news of the UK and US government entities joining the FIDO alliance[91].

T8: *Trust that hardware providers will not intentionally break the unlikability property.*

The FIDO consortium is not responsible for the ongoing testing process of all authenticator units produced by every hardware manufacturer. Furthermore, the metadata part of the metadata store and that are verified by APs during the attestation process, are only a subset of the authenticators characteristics. These facts give the opportunity to an evil hardware manufacturer to omit security relevant characteristics from the metadata characteristics disclosed during the certification process. This hardware manufacturer can then use this as a backdoor into manipulating some individual or entire authenticators, by making them susceptible for releasing private information for instance. As a more

dangerous exploit, the hardware manufacturer can manipulate its authenticators into compromising the cryptographic material it stores. The latter scenario leads us into identifying another trust requirement [124].

T9: *Trust that hardware manufacturers will not keep a backdoor in the authenticator, and exploit it to release secret cryptographic information of its users.*

II.3.7 Local Device Computing Platform

The currently deployed online IdM solutions focus on defining more secure communication protocols between their end points (client, SP server, identity provider server). The question of whether the client computing platform (e.g.: computer, mobile phone) end users use to connect to SPs is compromised or not has been left out from their solutions. However, a number of the attacks that are directed against online IdM solutions do start from a compromised computing platform(a rooted computer, a keylogger, a compromised browser). This kind of threats is not very far-fetched. According to PandaLabs estimates, 31.63 percent of the world's PCs are infected with some sort of malware (Q2 2012) of 78.92 percent are Trojans [97]. This situation makes it so that even with the most carefully designed communication protocol between end points, and the most safely guarded server platform, the password of end users can still be compromised if they are authenticating to their SPs from a compromised device. Hence, one would deduce that any new online IdM solution aiming to improve the strength of its solution would not introduce the clients' computing platform as a component of its architecture. Unfortunately, this idea lies at the core of FIDO's architecture:

1. Besides the user agent (browser, app), FIDO has introduced the concept of the FIDO client. The latter communicates with both the FIDO authenticator and the user agent, and may be implemented in whole or in part within the boundaries of the user agent [124].
2. Furthermore, FIDO supports authenticators that are implemented as part of the computing platform, where secret cryptographic information are allowed be stored on the client's platform user space.

Indeed, while FIDO has introduced the client's computing platform as a main new component in its architecture, its solutions assumes that the client computing platform is not compromised by viruses or Trojans. Unfortunately, the reality could not be further from that as the studies show.

T10: *Trust that the user computing platform is not compromised by malicious soft- ware.*

II.3.8 End Users

The other side of trusting the local user computing platform to not be compromised is to have trust in the user not compromising it:

T11: *Trust the user will not expose his or her device to compromise in infected platforms.*

II.3.9 FIDO protocols

FIDO is a very recent protocol. However, it has been interesting to see it gaining ground so rapidly with industry players that have a very large base of customer. FIDO protocols have received little challenge from the security community. Most of the security reports and guarantees are coming directly from the FIDO consortium itself. In other words, FIDO protocols have not been put to the test of the market yet, decreasing hence the level of trust we can have in their strength. The trust In FIDO protocols can be expressed in two aspects.

T12: *Trust in the security of the protocol design.*

T13: *Trust in the security of the protocol implementation.*

Indeed, on the user computing platform, the FIDO architecture has introduced more layers of abstraction, that communicate with each other through FIDO specific API. As illustrated in figure II.7, the layers of the abstraction from top to bottom are as follows: user agent (browser, app), FIDO client, ASM, FIDO authenticator. Except from ASM, all other components have already been introduced throughout the paper. ASM is a platform-specific software component offering an API to FIDO clients, enabling them to discover and communicate with one or more available authenticators. A single ASM may report on behalf of multiple authenticators. The interactions between all of these new layers are done through FIDO specific API which has so far received little challenge from the market.

II.4 Discussion

In order to arrive to an accurate assessment of FIDO trust requirements, it is important to evaluate them against the trust requirements of the currently deployed online IdM solutions (isolated, federated). In this discussion, we distinguish between three categories of FIDO trust requirements (hereafter referred to as TR): TRs that FIDO has inherited form the previous solutions, TRs it has eliminated, and finally TRs it has introduced. Table II.1 summarizes the trust requirements of all the previously discussed on- line IdM solutions. The TRs that are common to more than one solution are only cited once.

Eliminated Trust Requirements: One of the main positive trust implications of FIDO is the fact that it has freed service providers from a big liability they had to incur. This has been achieved by the FIDO architecture which doesn't rely on service providers storing end users secret information, e.g. passwords. With the increasing number of compromised service provider servers, this represents indeed a significant trust improvement for FIDO over both the isolated and federated online IdM solutions.

New Trust Requirements: FDO has introduced trust issues that were not present in the previous online IdM solutions, namely: authenticator hardware

Trust Requirements	FIDO	Isolated	Federated
The SP protects client privacy.		√	√
The SP has implemented satisfactory user registration procedures and authentication mechanisms.	√	√	√
The client handles their authentication credentials with adequate care.	√	√	√
Trust in the SP: ability to correctly assess the risk level associated with the usage of his service by all his users.	√	√	√
Trust in the computing platform: it's not compromised by malicious software.	√	√	√
Trust the user will not expose his device to compromise in infected platforms.	√	√	√
Service access on behalf of users will only take place when legitimately requested by the client.	√		√
The SP adheres to the accepted policy for correlating personal data about the same client from other SPs.	√		√
Trust in FIDO consortium: it has identified the right set of meta-data.	√		√
The SP has the expected identity.	√	√	√
Trust between Federated Service.		√	
Trust in the Identity Mapping.			√
Trust in FIDO consortium: certificate is still meaningful throughout its lifetime.	√		
Trust in FIDO consortium: its ability to detect and report authenticators breaching the metadata characteristics declared in their certification process, and update the meta-data store accordingly.	√		
Trust in FIDO consortium: validity of the FIDO PKI used.	√		
Trust in Hardware manufacturers: they will not unintentionally break the unlinkability property.	√		
Trust in Hardware manufacturers: they not intentionally break the unlinkability property.	√		
Trust in hardware manufacturers: they not keep a backdoor in the authenticators.	√		

Table II.1: Trust requirements: FIDO Vs. Isolated IdM Vs. Federated IdM

manufacturers and the FIDO consortium. As we can note from Table II.1, a compromised authenticator can compromise both the secret key as well as the privacy of its user. In order to resolve and mitigate this trust issue, FIDO relies on its FIDO consortium, which will be responsible for controlling the ecosystem of hardware manufacturers (certification process and metadata store). We believe this is a very weak solution for such a severe trust risk (please refer to section II.2.3 for details about the trust requirements related to hardware manufacturers and the FIDO consortium).

Inherited Trust Requirements: Last but not the least, Table II.2.3 shows that FIDO has just inherited several trust requirements from the previous online IdM solutions, as its architecture didn't resolve their underlying issue. The most severe of these Trust requirements are the ones related to trusting the computing platform. With 31.63 percent of the world's PCs infected with some sort of malware (Q2 2012) of which most (78.92 percent) are Trojans [97], working under the assumption that the computing platform is not compromised is not reasonable. This trust requirements is very relevant to FIDO because now the end user secret key is not stored on the server side, but rather on his own authenticator, which has to be connected to his possible compromised computing platform.

The main claim of FIDO is that it is going to make authentication both more usable and stronger. While FIDO does indeed offer a better usability experience to its end users, by alleviating them from the burden of passwords, and unifying their authentication experience across all their (FIDO enabled) service providers, our trust requirement analysis shows that it falls short on the front of strengthening the authentication process. The above discussions led us to conclude that instead of solving the trust requirements of the previous online IdM solutions, FIDO has just shifted them to other components in its architecture. Indeed, FIDO has created a more complex ecosystem, with new components (authenticator hardware manufacturers and the FIDO consortium), to which previous trust requirements (mainly service provider ones) has been delegated. We believe this new FIDO trust requirements map, puts too much power and responsibility in the hands of entities that cannot be trusted, especially in a world where online digital attacks are increasingly becoming state affairs.

II.5 Conclusion

Adequate management of identities in open computer networks is crucial to providing security and improving efficiency. IdM requires an integrated and often complex infrastructure where all involved parties must be trusted for specific purposes depending on their role. The variety and complexity of the trust relationships required in the various IdM models can cause confusion for stakeholders. Satisfying the trust requirements also has a cost. Our study has tried to concisely analyse the trust requirements related to FIDO, and thereby allow these issues to be clarified. This study provides a basis for assessing the cost of satisfying the trust requirements, as well as for discussing and comparing

IdM solutions [67]. We have concluded that instead of solving the real trust requirements of the previous online IdM solutions, FIDO has merely created a new complex ecosystem, with new components to which these trust requirements have been shifted. While we do recognize the great usability advantages this solution brings to end users, we think FIDO is not making online IdM necessarily stronger. it is crucial to not jump on the fast adoption bandwagon before being fully aware of its trust requirements. This study will help all stakeholders understand what can go wrong with such a solution. And if we have learned anything from the unfolding of many security scandals over the last years, it is that: If anything in computer security can go wrong, it will eventually go wrong. In which case, we need to be ready to accept or mitigate the consequences. As an academic community, we have a lot to offer to this big industry led initiative FIDO, especially in terms of reviewing its trust requirements of this.

Paper III

TrustUI: Enabling Trusted User Input and Output Channels to Web Applications in Untrusted Client Platforms

III

Ijlal, Loutfi and Audun, Jøsang

Under Review at the 22nd Euromicro Conference on Digital System Design (DSD'19)

Abstract

As the proliferation of web services continues to increase, so does our reliance on web browsers. On the one hand, users routinely input sensitive data, such as passwords, into its web-pages. Yet, nothing stops a compromised browser from leaking it to malicious third-parties. On the other hand, users also trust the displayed browser web-pages to be non-tampered with. Yet, a Man-In-The-Browser malware (MITB) can stealthily modify the HTML rendering of these sensitive web-pages. MITB can inject authentic looking input fields into legitimate web-pages, prompting unsuspecting users to enter their sensitive data into them. It can also change the displayed value of a payment amount transaction to get the user to approve it, while rerouting the funds to a different account.

Indeed, the lack of trusted input and output channels compromises the trust between web users and their respective web service providers. Therefore, the aim of this paper is to enable trusted input and output channels from end-users to web servers, despite the presence of a malicious browser, and even a compromised system software.

TrustUI is the name of the paper's proposed solution. On the one hand, TrustUI mitigates the lack of trusted output channels by first programmatically capturing the browser's screen directly from the GPU's frame buffer. It then runs image-analysis on the captured screen to compare it to a generic trustworthy reference image. On the other hand, TrustUI solves the lack of trusted input channels by establishing an end-to-end encrypted channel between web users and servers. It achieves this with a combination of two trusted execution environments: An Intel SGX web enclave instantiated within the browser address space, and a personal security device showcasing a secure element and a separate keypad input at the side of end-users.

This paper also experimentally evaluates the core functionalities of TrustUI and reports its results.

III.1 Introduction

The web has become a vital part of today's modern society. Millions of users rely on it daily to retrieve information, transfer money, access e-government services, and conduct business [118]. They also rely on it to configure various safety-critical devices, such as industrial control systems and medical devices, which offer web interfaces for remote configuration [28]. This has made the web an attractive target for cybercriminals, who look for ways to compromise all its three core components: web servers, web clients and the network communication layer [103]. As a defence mechanism, the communication layer between web clients and servers is protected by end-to-end encryption technologies such as the Transport Layer Security, while web servers run on tightly managed commercial server platforms. They are frequently updated and often hardened with enterprise grade security mechanisms, such as Hardware Security Modules [101]. However, the web client remains the weakest attack vector, as it runs on the untrusted platforms of end-users.
Problem Statement. One of the most critical components of the client platform is the browser software. For many web users, it is their primary gateway for reaching and consuming web services. Consequently, browsers process and store various security-sensitive data, such as authentication credentials and credit card details. In this paper, we focus on the two following attack classes:

1. **Trusted Input Channel Attack**: when users need to communicate with their remote web service providers, they enter input data into their platforms through standard input interfaces, such as the keyboard. This input is then processed by the system software and the browser before being sent over the Internet to the web server. The later relies on this input to make various security decisions, such as granting access to the end-user and approving their transactions. However, a malicious browser or a keylogger which has hooked into the kernel or the browser' API is able to leak it to third-parties [92].

2. **Trusted Output Attack**: Web users routinely rely on what the browser displays to them in order to make security decisions. For instance, they check the banking transaction amounts displayed in their screen before approving them for payment. However, nothing stops a compromised malicious browser from redirecting the payment transaction to the attacker's account, then modifying the HTML rendering of the banking web page to still display the legitimate transaction. The end-user would then unsuspectingly confirm the transaction, and the bank would process it as if it originated from a legitimate user. It can also inject new authentic-looking fields prompting the user to enter sensitive information such as their credit card details. As the rest of the web-page, such as the URL and the 'https'

lock, remain unchanged, it is very difficult for users to detect that they are interacting with a maliciously tampered web page [140] [11]. This type of malware is referred to as Malware-in-The-Browser (MITB).

Indeed, the lack of trusted input and output channels compromise the trust between end-users and their service providers. Therefore, the main research question of how to enable web users to securely communicate with web application servers. In other words, we need to enable trusted input and output channels from the end-user to the web application, despite the presence of a malicious browser and even system software.

Our Solution. In this paper, a two-part solution architecture called TrustUI is proposed:

- **TrustUI-Trusted Output Solution**: The intuition behind TrustUI is the observation that it is ineffective to leave it to end users to detect when a web page's rendering has been modified as the new modified, since the rogue web-page can look completely identical to the original one. Furthermore, solutions proposing out-of-band verification, such as SMS's sent by banks to confirm payment details, can still be defeated. This is due to two fundamental weaknesses: on the one hand, out-of-band modalities are subject to habituation. Users simply start confirming the transactions out of habit without double-checking their details, or become inattentive to small transaction modifications such as similar bank accounts, or slightly rounded transaction amounts [3]. On the other hand, even if out-of-band verification is successfully carried out by the user, its effectiveness is only limited to attack scenarios where only dynamic page values, such as transaction amount, are modified. Multi-Factor out-of-band solutions are ineffective against attack scenarios where rogue elements are added into the web page, such as input fields prompting users to enter the passwords into them. However, TrustUI intuition is that a web server, such as a bank, can increase trust in the transaction they are processing if they have a guarantee that it matches the screen displayed to the end-user. In a parallel world, web servers would have a way to access the users' screen before every transaction. In the absence of that, TrustUI proposes a two-step solution which can achieve the same guarantees, albeit in a different manner. The first component is a screen capture engine: it programmatically captures a bitmap representation of the rendered display directly from the GPU frame buffer, which is out of reach from malware-in-the-Browser. The second component is the image analysis engine, which cross-compares the captured image to a reference one. This allows it to detect any rogue injected elements. Furthermore, if the goal is to detect tampering with dynamic values such as the transaction balance, the optical character recognition can be performed.

- **TrustUI-Trusted Input Solution**: to protect the confidentiality of user input from a compromised browser and operating system, TrustUI aims to

build an end-2-end encrypted input channel between the end-users' input interface and their web server. A straightforward solution to this problem would be to encrypt all sensitive input at the level of the keyboard's subsystem, before it becomes accessible to the rest of the platform's system software and browser. However, this solution is impractical with out-of-the box commodity platforms, since the standard keyboards are not able to perform secure cryptographic operations and negotiate keys. Instead, TrustUI proposes to equip users with personal security devices with dedicated input keypads. These PSDs can establish an end-2-end encrypted channel to a web enclave which TrustUI establishes within the browser's address space. Intel SGX can protect user's input secure against all platform malware, including a malicious operating system. Furthermore, the security of PSD's is based on an embedded secure element, and a separate keypad and display[66]. These two Trusted execution environments negotiate and establish an end-2-end encrypted channel. The user then uses the keypad of the trusted device to enter their security sensitive input, which is sent encrypted to the web enclave and finally to the web application.

Outline. The rest of the paper is organized as follows: section 2 presents the needed background. Section 3 introduces the proposed solution. Subsequently, section 4 presents the experimental results. The paper then follows with an overview of the related work, and concludes with a discussion of the results, open research questions and suggested future work.

III.2 Background

III.2.1 Man-in-the-Browser-Attacks

Man-In-The-Browser malware (MITB) is a 'Trojan that infects endpoints through malicious email attachments, links, or even when a user visits an infected website [113]. Once a platform is infected, MITB malware hooks itself into browser elements such as Browser Helper Objects (BHO), plugins, JavaScript, Add-on features, Ajax calls and DOM Object models[11].An MITB can then see and do everything a web user can see and do with a browser. The attacks it launches usually fall into one of two common patterns. First, MITB can silently wait for the user to visit a target web-page, such as the authentication page of a bank. It then intercepts the authentication credentials and leaks them to other malicious third-parties. In the second attack scenario, the MITB takes a more active role. It modifies the rendering of the webpage elements, effectively changing what is being displayed for the user. For instance, it can modify the value of a payment transaction, or even inject new authentic-looking fields prompting the user to enter sensitive information such as their credit card details. As the rest of the web-page such as the URL and the 'https' lock remain unchanged, it is very difficult for users to detect that they are interacting with a maliciously tampered web page.

In this way, MITB malware is different and more dangerous than traditional phishing attacks. while the later uses links or email attachments to get users to a fake website where they input their secure data, MITB still lets the victim interact with the legitimate website, while passively stealing their data or actively modifying their web-page display [113].

III.2.2 Frame Buffer

The frame buffer is part of the graphics subsystem. It is a large, contiguous piece of computer memory that sits between the processor and the display. Whenever the CPU sends a frame update to the graphics subsystem, the graphics processor forms a picture of the screen image and stores it in the frame buffer. 'It typically stores a bitmap representation of the screen, which consists of colour values for every pixel to be shown on the display. Colour values are commonly stored in 1-bit binary (monochrome), 4-bit palettized, 8-bit palettized, 16-bit high colour and 24-bit true colour formats. The data is sent as a digital signal to the display [84] [44].

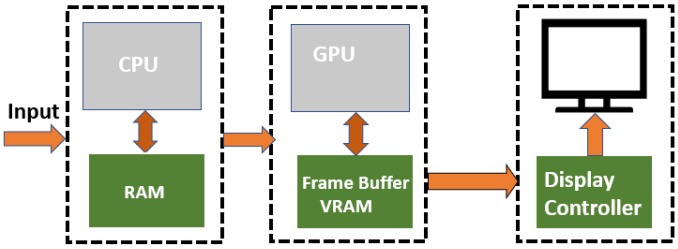

Figure III.1: Buffer frame: a system overview

III.2.3 Intel Software Guard Extensions

A Trusted Execution Environment (TEE) is a system security primitive that provides enforced isolation for executing security sensitive application logic. The TEE isolates its hardware and software resources from a general processing environment, also referred to as the rich execution environment (REE), where most of the platform software, including the primary operating system, runs.

Intel Software Guard Extension (SGX) are a recent security extension for the X86 Intel processor family. Its main aim is to allow ring 3 applications to control their own security by creating enclaves [20]. Enclaves are thus hardware TEEs that applications can create within their address space. Enclaves can then be used to run security-sensitive code and data, where no other software residing on the platform can access it, including the privileged system level software such as the operating system and the hypervisor. Inside the CPU's perimeter, enclave's data is processed in plaintext. Once outside the CPU, it always remains encrypted by the new encryption engine, using a hardware protected key which

is generated upon every platform reboot. However, enclaves lack the ability to execute system calls, as this relies on the operating system which is considered untrusted within the SGX threat model [20].

III.2.4 Personal Security Devices

Personal Security Devices (PSD) are minimal portable embedded devices, which are used to carry security-sensitive operations outside the perimeter of the potentially compromised general-purpose end-point devices. One way to conceptualize them is as a smart card with a display and a keypad. Indeed, they are often designed around a secure element, which is augmented by secure storage, and IO channels. While PSDs have been around for many decades, they have been popularized in recent years by the rise of cryptocurrencies which has prompted the need for wallets to store the cryptographic keys away from the malicious platforms [98].

OffPAD is a one such PSD which has been developed within the university of Oslo. It uses a secure dual-architecture, where certified secure elements are attached to a general-purpose microcontroller which drives the attached UI devices. These are a fingerprint reader, a keypad and a screen [132]

III.3 Solution Overview

III.3.1 TrustUI-Trusted Output Channel Solution

As discussed in the introduction, there are currently no perfect solutions for users and web servers to detect a browser page that has been tampered with. Even multi-factor authentication solutions which rely on out-of-band modalities such as SMS's are not enough. They are completely useless against MITB attacks which inject rogue new elements into the browser's page. Furthermore, even if they can help guard against modifications to dynamic values, such as transaction amounts, they are vulnerable to habituation. Users simply start confirming the transactions out of habit without double-checking their details, or become inattentive to small transaction modifications such as similar bank accounts, or slightly rounded transaction amounts [3].

Therefore, TrustUI intuition is that a web server, such as a bank server, can increase trust in the transaction they are processing if they have a guarantee that it matches the screen displayed to the end-user. TrustUI proposes the following two-steps solution: first, a screen capture engine which programmatically captures a bitmap representation of the rendered display directly from the GPU frame buffer, which is out of reach from malware-in-the-Browser. Second, an image analysis engine, which cross-compares the captured image to a reference one, or which perform Optical Character Recognition, depending on the type of transaction being processed.

III.3.1.1 USE CASE

While we have expanded on the description of such a malware-in-the browser attack in previous sections, we would like to provide the following use case as concrete motivating example to use for the rest of the paper:

1. A user wants to pay 100 USD to the merchant's account A.

2. A Man-In-The-Browser intercepts the transaction and changes the details to 'pay 200 USD to the intruder's bank account B', then forwards it to the bank web server.

3. the bank's web server sends a confirmation request to the user to approve the payment of 200 USD to the intruder's bank account B, in the form of a web page to be rendered by the browser and displayed on the screen.

4. The Man-In-The-Browser malware intercepts the HTML rendering request, and modifies it to display 'approve 100 USD to the merchant's bank account A'.

5. The user is prompted to approve the payment of 100 USD to the Merchant's bank account A.

6. The user approves the payment transaction, and the browser relays it to the bank's server.

7. The bank processes a payment of 200 USD to the intruder's account.

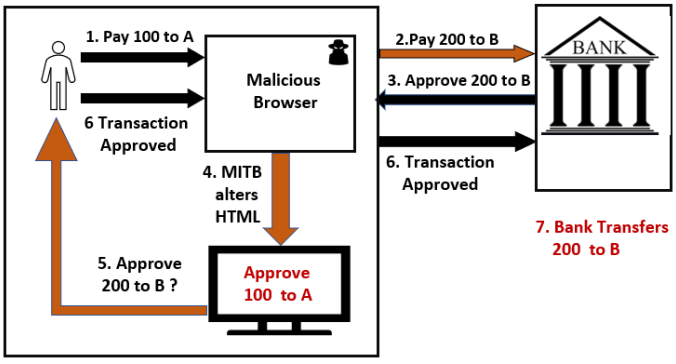

Figure III.2: Malware In the browser attack scenario

III.3.1.2 SOLUTION ARCHITECTURE

TrustUI proposes a two-step solution to allow web servers to gain trust that users are not viewing tampered with web pages.

1. **Screen Capture Engine**: it captures a bitmap representation of the rendered display directly from the GPU frame buffer, which is out of reach for MITB malware.

2. **Image Analysis Engine**: It cross-compares the captured image to a reference one and/or perform OCR to extract numerical and textual values from the capture.

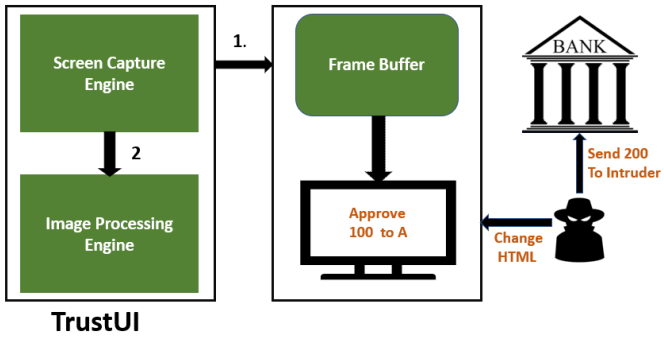

Figure III.3: TrsutUI solution overview for the trusted output channel attack

III.3.1.3 SCREEN CAPTURE ENGINE

To get a trustworthy copy of the displayed image, we need to capture it programmatically from a system component that is deemed trustworthy under our threat mode. The frame buffer is one such component. On the one hand, it always has a bitmap representation of the image to be displayed. On the other hand, MITB malware cannot tamper with its content. To securely access the frame buffer, there exists several options which are platform-dependent. In this section, we discuss solutions relevant to Windows platforms, since we used it to run the experiments part of this paper:

- **Kernel Mode Capturing**: it requires writing a kernel video miniport driver, that would have unprivileged access to the frame buffer [96].

- **User Mode Capturing**: it leverages one of the APIs that are offered by the Windows operating system for applications to programmatically access the frame buffer.

1. **Microsoft DirectX**: This is a collection of application programming interfaces (APIs) for handling tasks related to multimedia, especially game programming and video, on Microsoft platforms [96]. The most interesting API for this paper is the desktop duplication API. It provides controlled access to applications which require remote access to a desktop , and allows them to capture frames as they are updated. 'The interface IDirect3DDevice8 provides GetFrontBuffer() method that takes a IDirect3DSurface8 object pointer and copies the contents of the front buffer onto that surface. Once the screen is captured onto the surface, we can use the function D3DXSaveSurfaceToFile() to save the surface directly to the disk in bitmap format' [96].
2. **Graphics Device Interface** GDI is also a collection of APIs. It is used for representing graphical objects in the frame buffer, and then transmitting them to output devices such as monitors and printers. GDI is based on the Device Context (DC) object, which is handle to either an entire screen, a window, a subset of a window or a printer. Details about how to use GDI for screen capture are outline in the implementation section.

III.3.1.4 IMAGE PROCESSING ENGINE

Once we have programmatically captured the content of the screen from the frame buffer and reconstructed the image, we need to process it so as to decide whether it has been tampered with or not. The most straightforward solution here is to compare the captured image to a reference one, which we assume would have been sent by the web server securely. This is sufficient when TrustUI needs to detect whether MITB has injected new elements into the webpage or not, this would be sufficient to conclude whether the web page has been tampered with. However, if the goal is to detect whether a dynamic value such as the bank transaction details has been modified, then Optical Character Recognition is also needed. It can then be compared to the transaction amount being sent back to the server.

While TrustUI can handle both cases, the experiments reported in this paper focus only on comparing a screen capture to a legitimate reference one, without the OCR part. A straightforward solution for TrustUI is to compare the bitmap representation of the two images pixel by pixel. This is sufficient in our case, since TrustUI needs to capture the visual similarity between two images and not its semantic one.

III.3.2 Trusted Input Channel

When an end-user inputs their sensitive data into the platform's keyboard, its confidentiality can be compromised not only by a malicious browser, but also by all the software stack on that platform. A straightforward solution to this problem would be to encrypt all sensitive input at the level of the keyboard's subsystem, before it becomes accessible to the rest of the platform's system

software and browser. However, this solution is impractical for several reasons: first, standard keyboards are not able to perform secure cryptographic operations and negotiate keys. Second, it is not clear where the decryption would take place in such a scenario.

TrustUI still relies on the same idea of having the sensitive input encrypted while it is present within the platform. However, given the above discussed limitations, TrustUI leverages two additional hardware-based trusted execution environments to enable the solution.

- End-user side: instead of relying on the default platform keyboard interface, TrustUI equips the user with trusted embedded device, otherwise referred to as Personal Security Devices, PSD. At the core of PSDs' architecture is a secure element able of securely performing cryptographic operations, and which communicates securely with the PSD's dedicated UI channels, namely its keypad and display.

- Browser side: TrustUI instantiates an Intel SGX enclave within the browser's address space. This web enclave can then securely negotiate a secure channel with the user's PSD, decrypt all incoming sensitive input, and then forward it to the appropriate service provider. The enclave's operations are protected against any malware residing within either the browser or the platform's system software.

In order to enable this use case, TrustUI relies on a two-step protocol:

1. Secure Channel Establishment: this is a one-time interactive protocol which aims to establish a shared secret key between the web enclave and the personal security device.

2. Message Exchange: it takes place every time a user wants to enter security sensitive web input into a browser's page. The PSD uses the pre-shared secret key, negotiated in step 1, to encrypt the input, while the web enclave uses it for decryption.

III.3.2.1 INTEL REMOTE ATTESTATION

A core component of TrustUI secure channel establishment is the Intel remote attestation procedure. This section outlines its details before moving forward with presenting the solution.

Before a remote service provider provisions secrets into the enclave or accepts the output of its computation, it needs to establish two trust guarantees. First, the enclave needs to be running on a genuine SGX enabled Intel processor, and not an emulated environment. Second, the enclave's initial code and data which were loaded by the operating system should not have been tampered with. Remote attestation is the process which ensures these two guarantees. It is an interactive protocol between the attested platform, the attesting remote service

provider and the Intel Attestation Service (IAS), online service operated by Intel [20] [29].

Each SGX hardware has a unique attestation key, which is only accessible by special Intel SGX architectural enclaves, such as the quoting enclave. During an attestation protocol, the quoting enclave first forms a structure referred to as a quote, composed of a hash of measurements reflecting the order and the content of code which has been initially loaded into the enclave by the operating system. Subsequently, the quoting enclave signs the quote using the attestation key and forwards it to the remote service provider and the Intel Attestation Service. The IAS verifies that the signature has been generated by a genuine non-revoked Intel SGX hardware, while the service provider compares the measurement value to a reference one, so as to ensure that the enclave code has not been tampered with during the loading process [20] .

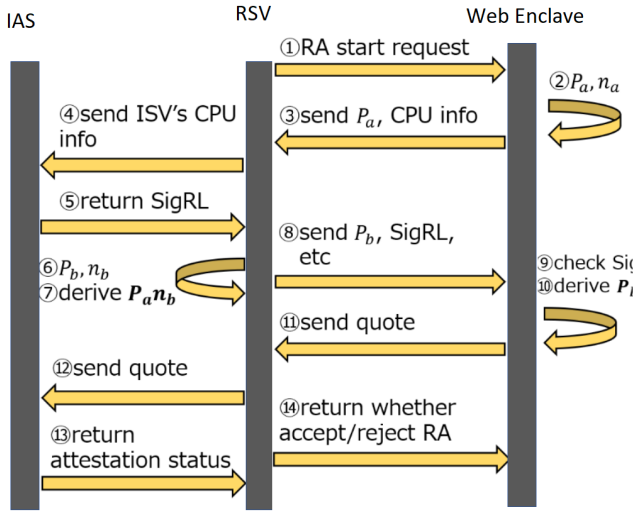

Figure III.4: Malware In the browser attack scenario [57]

III.3.2.2 SOLUTION COMPONENTS

In this section, we present an overview of the components that are part of the TrustUI solution of the trusted input problem, before introducing the secure channel establishment protocol.

Intel Attestation Service: IAS is involved during the remote attestation process of the web enclave. It has an identity key pair and a certificate signed by Intel.

Remote Service Provider: in our use case, RSV is the web server who needs to establish trust that the input it receives is coming from legitimate users and not malware.

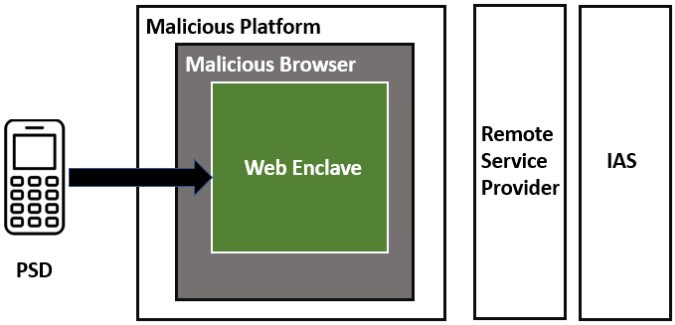

Figure III.5: TrustUI trusted input solution overview

Personal Security Device: the PSD has an identity key pair, whose public key is certified by the manufacturing authority. The certificate authority public key is publicly available to third-parties.

WebEnclave: it is responsible for negotiating the secure channel with the PSD, and then using the pre-shared key to decrypt the PSD's encrypted input.

III.3.2.3 SECURE CHANNEL ESTABLISHMENT

This a one-time interactive protocol performed at set up time, so as to provide the PSD and the web enclave with a symmetric share key they can use for subsequent communication.

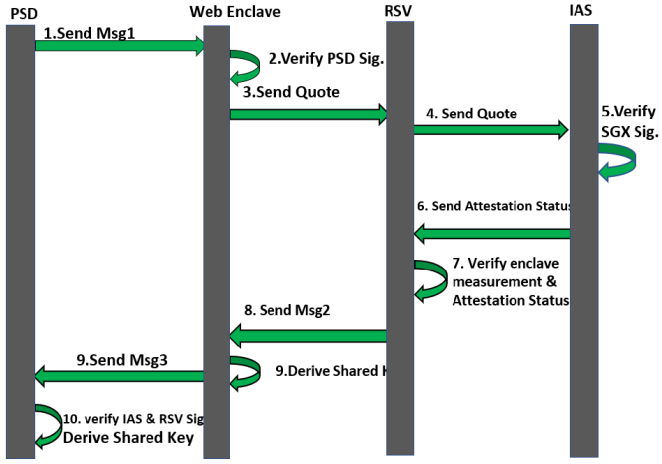

Figure III.6: Secure channel establishment protocol

1. PSD generates Pa and constructs Msg1=(nonce,Pa), signs it with its identity private key and sends it to the web enclave.
2. The web Enclave verifies the PSD signature to ensure it has been generated by a genuine hardware non-revoked PSD. We assume that all manufactured PSDs are signed by an authority whose certificate is public ally available to all parties involved in the protocol.
3. The web enclave invokes the quoting enclave to generate a quote structure and sends it to the RSV.
4. RSV forwards the quote to the IAS.
5. IAS verifies that the quote structure has been generated by a genuine non-revoked Intel SGX hardware.
6. IAS returns the attestation status to the RSV.
7. RSV verifies the attestation status and the enclave measurement, so as to ensure that the enclave has been loaded properly by the untrusted system software.
8. RSV constructs Msg2 = Signed by RSV identity private key (IAS quote response). RSV sends Msg2 to the web enclave.
9. Web enclave generates Pb, and derives Pab and constructs Msg3= (Msg2 + nonce + Pb) encrypted with Pab+ Pb
10. PSD derives Pab, decrypts Msg3, and verifies the IAS and RSV signatures.
11. Web enclave generates gd, and a TPM signature over the quote
12. OffPAD verifies TPM quote with enterprise certificate. Verifies signature of IAS, RS, and device's shared key

At the end of this protocol, both enclave and OffPAD derives a shared secret key K.

III.4 Experimental Set-Up and Evaluation

III.4.1 Trusted Output

The experiments were performed on an HP machine with an Intel Core(TM)i7 -75000 CPU, a 32GB RAM, and an Intel(R) HD Graphics 620. The platform runs a 64-bit Windows 10 Operating system Home edition. The experiment uses Instagram's login page as a use case. In order to mimic the MITB attack which injects rogue input fields into a page, we draw a rogue Instagram mock-up login page. This malicious page hides the legitimate username and password fields, which are centrally indented in the legitimate page, and replaces them with identical looking username and password input fields which are indented to

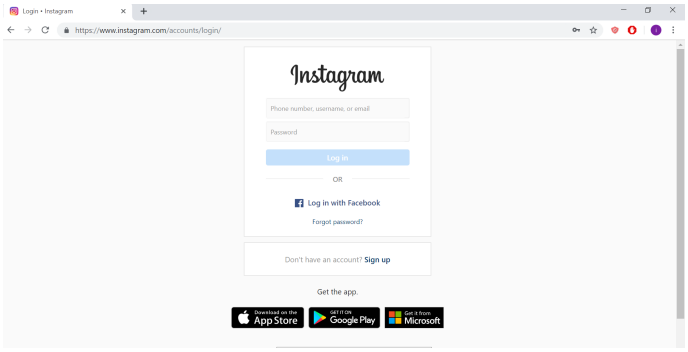

Figure III.7: Legitimate web page-username/password fields are in the center

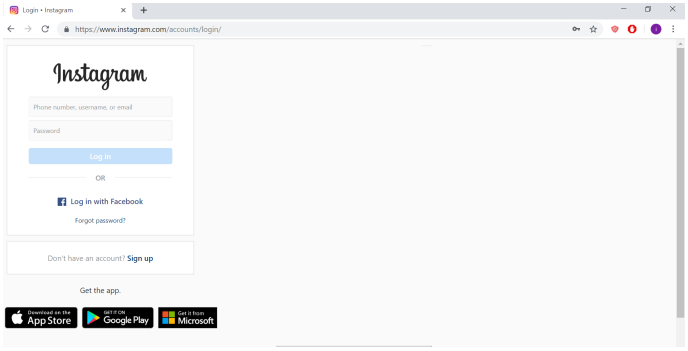

Figure III.8: Malicious web page- rogue injected username/password field are left indented

the left. Except from this modification, the webpage including the URL looks completely identical to the legitimate Instagram login page.

In order to capture the displayed screen, TrustedUI used an open source implementation of the GDI screen capture solution.[131]. GDI is based on the Device Context (DC) object, which is a handle to either an entire screen, a window, a subset of the window or a printer. Therefore, when TrustUI needs to programmatically capture the screen displayed by the browser, the following steps are followed:

- Get the device context of the desktop to be captured using the function GetDeksopWindow().
- Get the DC of the desktop window using the function GetDC();
- Create a compatible DC for the Desktop DC along with a compatible bitmap to select into that compatible DC using CreateCompatibleBitmap()

After having captured the image from the frame buffer, we perform a pixel-by-pixel comparison of the bitmap representations of the reference image and the captured one, which is supposedly malicious. As the two images we used are different, we get a trigger that the webpage is malicious.

While the delay introduced by the screen capture image is insignificant, the image analysis takes 4 seconds to complete.

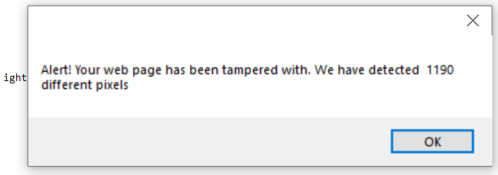

Figure III.9: MITB attack detected

III.4.2 Trusted Input

The implementation of TrustUI trusted input focuses on its core component, which is the client side. The experiments were run on an HP machine with an Intel Core(TM)i7 -75000 CPU that is SGX enabled, a 32GB RAM. The platform runs a 64-bit Windows 10 Operating system Home edition. The provisioning was left out of this paper's experimented and assumed to have been carried at a prior time as per the above described protocol, or that the shared key has been instantiated manually during the platform initialization. For the personal security device, the experiment uses an Arduino Uno, which simulates the keypad. The Arduino Uno sends encrypted messages over a serial communication port to the application which creates an enclave. This experiment used AES-256 for encryption. The main overhead introduced in this scenario is the fact that the user needs to use an additional piece of hardware to enter his input.

III.5 Related Work

Trusted Input: The lack of a trusted channel from the web user to the web server is a recognized problem. Since the release of Intel SGX, it became apparent that enclaves can play an important role in protecting security-sensitive input data from malicious client platforms. However, Intel SGX lacks support for trusted input channels to the end-user. Therefore, there have been several proposals to resolve this. One set of solutions relies on the hypervisor to establish generic secure IO channels, as is the case of SGXIO [137]. The other set of solutions uses external embedded hardware devices to establish a secure communication channel to either the SGX enclave of the web server. Indeed, the idea of using an external trusted embedded for trusted UI is not new, as it dates

```
#endif
{
    /*************************************/

    arduino = new SerialPort(portName);
    while (!arduino->isConnected()) {
        arduino = new SerialPort(portName);
    }

    //Checking if arduino is connected or not
    if (arduino->isConnected()) {
        std::cout << "Connection established at port " << portName << endl;
    }


    std::cout << "Receiving data from arduino " << portName << endl;
    arduino->readSerialPort(incomingData, MAX_DATA_LENGTH);
    printf("%s \n", incomingData);

    std::cout << "Sending data to arduino " << portName << endl;
    arduino->writeSerialPort("hello\n", MAX_DATA_LENGTH);
```

Figure III.10: Arduino serial communication

back to the 80s, before the proliferation of smart phones and personal security devices [126]. More recently, there has been a renewed interest in using these trusted devices with Intel SGX. Just as we were working on this research paper, SGX-USB proposed a solution with practically the same architecture as TrustUI [63]. Fidelius and ProximetEE also propose very similar architectures [34] [29]

Trusted Input: Authentication credentials are a common attack target for active Man-In-The-Browser malware. Therefore, many service providers, such as banks, propose two-factor authentication which rely on out-of-band verification to mitigate this attack. Such solutions can be defeated because of two fundamental weaknesses. On the one hand, the out-of-band modality such, as SMSs sent by the bank to confirm transaction details, are subject to user habituation. Users simply start to confirm the transactions out of habit without verifying the SMS details, or become inattentive to small transaction modifications such as similar bank accounts, or slightly rounded transaction amounts [3]. On the other hand, even if out-of-band verification is successfully carried out by the user, its effectiveness is only limited to attack scenarios where only dynamic page values are modified. It is, however, ineffective against attack scenarios where rogue elements are added into the web page, such as input fields prompting users to enter the passwords into them. Therefore, solutions which aim at improving the overall security of the browser software have been proposed. Promos integrates support for browsers at the OS-level [120]. Cloud terminal runs a lightweight secure thin terminal locally and outsources the rest of the security-sensitive computation to a remote server [1]. Shadowcrypt uses a Shadow DOM to allow encrypted input/output for web applications [43].

III.6 Conclusions and Future Work

Users and web browsers make various security decisions based on client input and output channels. Therefore, this paper proposes TrustUI, a new architecture solution for enabling secure input and output channels from the end-user to the

web server, despite the presence of a malicious browser, able to leak sensitive users' input, and modify the browser pages displayed to them. In order to mitigate the trusted output attacks, we design a two-component solution, where we first programmatically capture the screen directly from the frame buffer, then compare it to a reference trustworthy screen. The paper then experimentally evaluates the screen capture engine using an open source GDI API implementation. As for the trusted input attack, we propose to equip users with personal security devices with dedicated input keypads. These PSDs are able to establish an end-2-end encrypted channel to a web enclave which we establish within the browser's process address space. We use an Arduino Uno to simulate user's input into the PSD and send AES encrypted input messages to the application.

As a future work, we would like to implement and run both the image processing and screen capture engines within a more trustworthy execution environment, such as an Intel SGX enclave. In this case, TrustUI would need to utilize libraries which allow applications to run unmodified within the enclaves, such as Graphene [129]. Finally, TrustUI acknowledges the fact that it burdens the user with the need to carry an extra hardware device. Therefore, we would to explore as a future work to explore the feasibility of using the new android transaction confirmation API. This would allow TrustUI to use smartphones instead the personal security devices, while providing similar security guarantees and more usability [25].

Paper IV

SMMDecoy: Detecting GPU Keyloggers Using Security By Deception Techniques

Ijlal, Loutfi

Published in the Proceedings of the 5th International Conference on Information Systems Security and Privacy-ICISSP'19 pp. 580-587
DOI:10.5220/0007578505800587

Abstract

Human computer interaction is a fundamental part of the modern computing experience. Everyday, millions of users rely on keyboards as their primary input interface, and use them to enter security sensitive information such authentication credentials. These can be passwords, but also multi-authentication factors received from other devices, such as One Time Passwords and SMS's. Therefore, the security of the keyboard interface is critical. Unfortunately, both PS/2 and USB keyboards have open buffers which are vulnerable to sniffing by keyloggers. This paper focuses on the detection of the stealthiest variance of keyloggers, which is deployed within IO devices firmware, such as GPUs. We propose to use principles of security by deception: We inject decoy credentials into the open keyboard buffers, and give GPU keyloggers the opportunity to sniff them. These decoy credentials are then sent to a remote server that can raise an alarm anytime an attacker uses them. We assume a strong adversary that can infect both the GPU and the kernel. Therefore, we propose to deploy the solution within System Management Mode, and leverage Intel Software Guard Extensions for network communication. Both SMM and SGX are hardware protected against the OS and DMA, and provide thus strong security guarantees to our solution, which we name SMMDecoy.

IV.1 Introduction

Despite recent advances in user authentication, the most commonly deployed mechanism in the internet today is still passwords. Everyday, millions of users rely on their username/password credentials to gain access to security-sensitive digital services. However, password-based authentication is routinely compromised. The main attack vectors can be classified in three categories: 1) the

server side where password files are stored, 2) users' bad cognitive habits of choosing weak and redundant passwords or falling prey to social engineering 3) and finally, vulnerabilities within end-user devices where passwords are stored and operated on [47] [39].
As a matter of fact, over the past decade, we have witnessed many password disclosures of high profile companies such as Sony, Yahoo, and LinkedIn. Analysis of these breaches revealed that many of these online service providers implement bad security practices, such as using unsalted hashes, and weak deprecated hashing algorithms. The silver lining of this slew of high-profile password compromises, is the increased awareness about the issues, which has prompted many companies to review and harden their password servers' security. Consequently, this has been increasing the cost of attack of database servers, and thus, turning the attention of attackers to endpoint devices as a more attractive attack vector [125] [48].
Furthermore, the adoption of multi-factor authentication by many online service providers has further motivated attackers to focus on endpoints. For example, many banks distribute One Time Password generators (OTP) to their users, while others rely on out-of-band methods such as SMS to send second factor authenticators. However, users still need to input these second factors into the same endpoint device where the password is entered. This mechanism consolidates the whole authentication process into one attack vector, aka the endpoint device, where they can be exposed to a host of attacks. One such critical endpoint attack vector is the keyboard device. Since users almost exclusively use it as an input channel, it is of uttermost importance that it preserves the confidentiality and integrity of users' data [60].

IV.1.1 Problem statement

Unfortunately, keyboards suffer from a big sniffing problem. This is due to their hardware interface, which is open for direct reading by privileged software as well as DMA capable devices. This is true for both PS/2 and USB keyboards [93].

- for PS/2 keyboard, the data output buffer which resides on the chipset keyboard controller and is used to transfer scan codes upon a key press/release, is exposed to the platform host [145].
- For USB keyboards, the data transaction buffers which are memory mapped are also exposed to the host[70].

One critical malware category that takes advantage of this insecure keyboard interface is keyboard sniffers, more commonly referred to as keyloggers. Once they infect a platform, keyloggers log keyboard activity, and leak data to remote third parties. As documented in numerous disclosed attacks, keyloggers pose a serious threat against personal and financial data, despite continuous efforts to guard against them. For instance, keyloggers have been used to steal 10775 unique bank account credentials for customers who have shopping at Barnes and

Noble stores over a period of 7 months [111] . Even governmental agencies have relied for years on hard disk based keyloggers, as documents by recent leaks [50] .While keyloggers can be implemented on either hardware or software, we focus in this paper on the latter, as it is much more widespread and assumes a stronger attacker. Software keyloggers can be classified under three categories: [93]:

- User-level keyloggers: They often hook into application-level API, but can also reliably detected, through more privileged system level hook-based techniques.
- Kernel-level keyloggers: they hook into kernel-level API, and by doing so, inadvertently modify the kernel's code base, and thus its signature. Therefore, they can often be detected based on integrity verification and code attestation techniques.
- Firmware level keylogger: They can exploit firmware level vulnerabilities within the BIOS or any IO device, especially ones which are open to general computations such as modern GPUs. Therefore, they live outside of the CPU execution environment, and can evade its detection mechanisms. They can rely on the IO device DMA capabilities to directly read and sniff the keyboard's data registers.

In this paper we focus on firmware level keyloggers, and formulate our research questions as follows:

How can we reliably detect the presence of stealthy GPU keyloggers on endpoint devices? This research question is important the following reasons:

- The increased complexity of devices functionality, which is correlated with an increased complexity of its corresponding firmware, has made the firmware attack vector significantly larger.
- Many firmware malware categories, such keyloggers, do not need to hook into any kernel API or structures. Therefore, system level detection mechanisms such as code integrity and control flow integrity are inefficient against it.
- CPU-based monitoring solutions, e.g. antivirus systems, cannot monitor code residing on other execution environments within other devices.
- The increasing ease of deploying firmware Malware. This is especially true for devices such as GPUs, which have become open to general-purpose computations. In fact, GPUs have been traditionally used to process graphics rendering code, relieving the CPU this way from these heavy computations. However, the popularity of the gaming and AI industries, and their increasing demand for more GPU computational power and functionality, has made GPU general purpose computing much more extensive. Attacker are naturally interested in exploiting this large attack vector. Furthermore, 99 percent of worldwide GPUs support GPGPU

computations, which greatly increases the infection ratio of GPU malware. This is different from previous firmware attacks, which had to be more targeted, and thus limited to smaller infection ratios [70].

*For the remainder of this paper, we will use GPUs as an example for devices' firmware level software.

The solution proposed in this paper is inspired by deception-based techniques, which are traditionally used within the server side in order to harden the detection of password database files breaches. An example of such solutions are honeywords, where each user is associated with one legitimate password, and n fake ones. This directly increases the effort required by attackers to brute force passwords. Furthermore, in case a fake one is used, it automatically triggers an alarm signalling a potential breach [136].

Similarly, this paper's main intuition is that we need to deploy a transparent mechanism which can inject intentionally crafted noise, which mimics authentication credentials, to the keyboard's buffer, and allow any potential GPU keyloggers to monitor and sniff it. SMMDecoy would subsequently send a list of injected decoy credentials to a remote third party. We propose to deploy such a solution within System management mode, SMM, in order to take advantage of its integrity and transparency guarantees. We also propose to use Intel Software Guard Extension remote attestation capabilities to send decoy credentials over the internet. We call our solution, SMMDecoy. The rest of the paper is organized as follows: section 2 presents the necessary background. We then introduce the threat model, the solution design, and its message flow. An overview of previous related work is presented next. The paper closes with a set of conclusions as well as an overview of open questions and suggested future work.

IV.2 Background

IV.2.1 Keyboards Hardware Interface

Keyboards are such a prolific part of everybody's computing experience. They are also a critical security component, since they are used as the main input channel on many types of endpoint devices and are relied upon to communicate security sensitive information to the system software and eventually to our trusted online service providers. Examples of such information are authentication credentials. There are several ways of connecting a keyboard to an x86 Intel platform: they can be either wireless or wire-base. For the latter, we can further classify the connections as either PS/2 or USB based. While the latter rely on a serialized protocol, their interface to the host is different:

- The PS/2 Keyboard interface is composed of keyboard processor which resides inside the keyboard itself, and a keyboard controller which is part of the host chipset. The interface of the keyboard controller has 2 pairs of output buffers in the direction of the CPU, and 1 pair in the direction of the keyboard. The output buffer is used to transfer the scan code if a

key is pressed. It is readable by any software and causes keyboard sniffing problem [10].

- USB based keyboard don't have a chipset-resident keyboard controller. They have instead a host controller and a root hub. The host controller is represented by a set of buffers and structures, which are mapped to main memory, and are accessible through a set of system registers, also creating a snigging vulnerability [10].

IV.2.2 Scan Codes

"A Scan Code is a data packet that represents the state of a key. If a key is pressed, released, or held down, a scan code is sent to the computers onboard keyboard controller. There are two types of scan codes: Make Codes and Break Codes", which are used for the events of key press or release. For every keyboard key, there exists a unique make code and break code. When a user presses a key on the keyboard, a scan key is sent to the keyboard chipset-resident controller, which then buffers it on the output data buffer. It then raises the Interrupt request line, which will cause the IRQ 1 to be fired if it is not masked. When the interrupt is scheduled, the corresponding keyboard handler reads the output buffer and converts the scan codes into their corresponding key value. It is important to note that the scan codes within the output buffer can be read by software, even outside the interrupt handler procedure, and this is the crux of the keyboard sniffing challenge [87] [10] .

IV.2.3 System Management Mode

System Management Mode, SMM, is a highly privileged x86 CPU mode. SMM code is part of the BIOS code that resides on the SPI flash memory. During the system boot up and before the operating system is loaded, the BIOS loads SMM into a hardware protected memory area referred to as SMRAM, and which is not addressable from any other CPU mode, including kernel and VMX modes [142]. SMM implements a number of SMI handlers, which traditionally handle system control functions, such as power management. In order to execute SMI handlers, an SMM pin should be asserted, which will then trigger an SMM interrupt. Before the system switches to SMM mode, the CPU state is securely saved into SMRAM, so that it can return to it upon exiting SMM. This makes SMM highly transparent to all privileged system level software. This feature has been recently motivating many novel ways of using SMM for non-traditional purposes, e.g. debugging and system introspection [27].

IV.2.4 Software Guard Extensions

Intel's SGX are security extensions which are come as part newer Intel X86 CPUs. Its main aim is to instantiate an isolated trusted execution environment within the user space, called an enclave. Enclave code and data reside in

specialized protected memory called enclave page cache (EPC), which encrypted, and hardware protected. No privileged mode code can access the enclave, including the OS and hypervisor. SGX enclaves also rely on the intel management engine EPID group identity to establish a remote attestation protocol with Intel attestation servers, and through it to third party service providers [130].

IV.3 Solution Overview

IV.3.1 Threat Model

SMMDecoy assumes an active attacker who has unlimited computing resources and can exploit zero-day vulnerabilities of the host OS and user level applications. It also assumes that the GPU is compromised, and so are all other I/O devices, with the exception of the keyboard. Therefore, the only trusted components of the system, are the BIOS and the keyboard. SMMDecoy requires BIOS to be trusted only upon boot up, and not during runtime. We also assume that the attacker does not have physical access to the machine. We do not consider Denial-of-Service (DoS). The BIOS is trusted because newer X86 platforms are equipped with a Static Root of Trust of Measurement, SRTM, with a corresponding secure implementation of a Core Root of Trust of Measurement, which can ensure the code integrity of the BIOS upon boot. Some solutions such as HP SureStart also ensure that BIOS recovery as well, in case a code integrity compromise is detected.

IV.3.2 High Level Architecture

The intuition behind SMMDecoy is to inject specially crafted noise which mimics genuine authentication credentials into the keyboard output buffers. We assume that potential firmware keyloggers will be monitoring the buffers. We will then communicate these decoy credentials to relevant remote third parties. If we detect any authentication attempt using any of the reported decoy credentials, an alarm should be raised. Such a detection mechanism would subsequently provide a wealth of information about how the malware attack space, and the platforms from which it spreads. Thus, the requirements of such a solution are as follows:

1. SMMDecoy should be implemented within a system component that would allow it to be transparent to both the OS kernel and to the GPU malware.

2. The decoy authentication generation algorithm should mimic real world passwords as much as possible.

3. SMMDecoy should provide a mechanism for communicating with third party remote servers securely.

As figure 1 illustrates, SMMDecoy is made up of two parts:

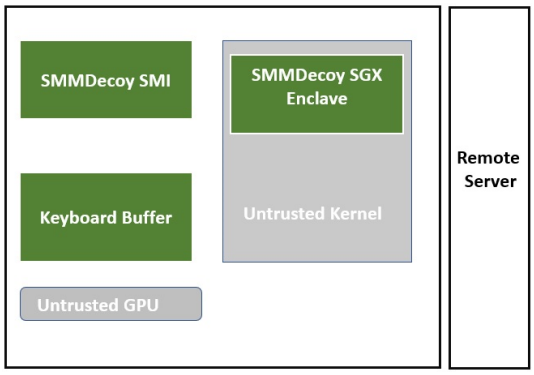

Figure IV.1: SMMDecoy architecture: trusted components in green

- SMMDecoy SMI handler :it is the trusted part, and is used to generate decoy credentials, inject them into the buffer interface, and then report them to a remote server.
- SMMDecoy SGX enclave; It used to establish an end-to-end secure channel between SMM and a remote server, which will be responsible for raising alarms when a decoy credential is used by attackers.

IV.3.3 SMMDecoy Message Flow

At a conceptual level, SMMDecoy adopts the same architecture for both PS/2 and USB keyboards. However, their interfaces are different and so are their implementation details. We present SMMDecoy for each keyboard separately.

Step 0: This is a common step for both implementations, and it takes place before the deployment of the solution. We need to customize the BIOS firmware, and add to it the SMMDecoy SMI interrupt. If this solution is deployed within an enterprise environment, this can be done as part of the platform provisioning by the IT department. Upon system boot up, SMMDecoy SMI will be loaded securely into SMRAM. From this point on, SMMDecoy message flow will diverge depending on the keyboard connection, PS/2 or USB, which can be detected upon boot up.

IV.3.3.1 PS/2 Message flow

- In step 1, SMMDecoy SMI generates fake user authentication credentials, which mimic legitimate credentials (more details in section 4.5) and converts them into scan codes.
- In step 2, SMMDecoy SMI is triggered based on A timer. The system then enters SMM modes, saves the CPU system state, as well the keyboard buffer content into SMRAM. o SMIDecoy then sends a 0xD2 command

into the keyboard control register, whose address is 0X64. This command allows anything that is subsequently written into the output buffer to appear as if it was generated by the keyboard.

- step 3, SMMDecoy decoy injects the scan code corresponding to the generated Decoy credentials into the keyboard data output buffer, by writing into address 0x60.

- In step 4, and after enough time elapsed to allow any potential firmware keylogger to sniff the keyboard output buffer, SMMDecoy SMI restores the state of the keyboard buffer, and exits SMM mode. This restores the CPU state, and gives back control to the OS so that it can resume its normal execution.

- In Step 5, SMMDecoy periodically communicates the list of decoy credential used to a remote server. We differ the details of this step to section 3.7.

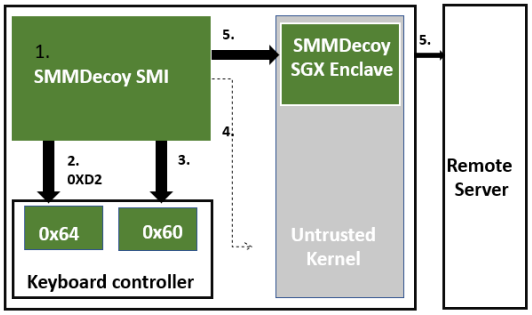

Figure IV.2: PS/2 SMMDecoy message flow

IV.3.4 USB Keyboard

- In step 1, SMMDecoy searches the system memory in order to find the memory address for the system keyboard buffer. In Linux, an attached USB device is represented by a USB Request Block (URB) structure, defined in the linux/usb.h header file of the Linux source tree. The keyboard buffer is part of this URB structure. The SMI then saves the physical address so as it can properly access it [70].

- In step 2, SMMDecoy SMI generates decoy credentials, which mimic legitimate credentials (more details in section 3.6), and converts them into scan codes. SMMDecoy SMI injects the scan codes corresponding to the generated Decoy credentials into kernel keyboard bugger, which it addresses using its physical address.

- In step 3, and after enough time elapsed to allow any potential firmware keylogger to sniff the keyboard output buffer, SMMDecoy SMI restores the state of the keyboard buffer, and exits SMM mode. This restores the CPU state, and gives back control to the OS so that it can resume its normal execution.

- In Step 4, SMMDecoy periodically communicates the list of decoy credential used to a remote server. We differ the details of this step to section 3.7.

IV.3.5 Writing Into The Buffer

For both the PS/2 and USB keyboards, we mentioned that the SMMDecoy SMI injects decoy credentials into either the output buffer of the system buffer. However, this is an abstraction, since the buffer size is limited and would require multiple coordinated writes. For instance, the Linux keyboard buffer has 16 bytes, and each scan code is 3 bytes long. Therefore, SMIDecoy is expected to perform multiple consecutive writes into the buffer [87].

IV.3.6 Generating Fake Credentials

The injected decoy credentials need to be statistically indistinguishable from genuine credentials. This can be achieved by encoding password generation policies into the SMMDecoy SMI handler, such as using a combination of characters and numbers, and having a minimum password length. Furthermore, we can dynamically update this algorithm with contextual user specific data, that the SMMDecoy handler would collect from the system. Such as other passwords used by the user, his name/ID. . . etc

IV.3.7 Establishing a secure channel between SGX enclave and SMM

Deception techniques are useful only if we are able to detect the decoy credentials being used or leaked by potential malware at some point in time. For SMMDecoy, this can happen at two points:

- Local detection: SMM can choose well-crafted patterns for the decoy credentials it injects. Therefore, it could also intercept all outgoing networks packets and look for the same pattern. Such solutions have been previously explored in the literature and are not the focus on this paper [94]

- Remote detection: SMMDecoy can send the injected decoy credentials to a remote server, which might be the service provider whose credentials we have been injecting into the keyboard buffer. While the actions the remote server takes upon detection of a used decoy credential are outside of the scope of the paper, we discuss a number of options here for the same of completeness. In fact, decoy credentials can be used to augment an already existing honeyword implementation. In this case, SMMDecoy

will increase the probability of the attacker choosing a decoy honeyword to authenticate to the server provider. Furthermore, unlike a traditional honeyword which would only signal the existence of a breach, SMMDecoy reveals a wealth of information about the malware attack vector and the platforms from which it is spreads. Decoy credentials could also be used as a standalone honeyword where decoy accounts are provisioned in order to allow the attacker to log into them, and leave traces of their attack details, such as the amount of money they transfer.

We have considered two approaches to achieving the remote detection:

- Porting trusted network drivers into SMM.
- Relying on Intel SGX remote attestation.

While SMMDecoy proposes to use intel SGX, we discuss both approaches subsequently for completeness.

IV.3.7.1 SMM Trusted Network Drivers

If SMMDecoy wants to send data over the network, it needs to make use of the network drivers which are part of OS. However, the latter is assumed to be malicious within our threat model. This question has been a common challenge for many SMM based solution. One way it has been solved is by porting commodity drivers into SMRAM. This has been possible because SMM mode is similar to kernel mode where privileged CPU instructions are available. Aurora authors also argue that the mechanism of interrupt rerouting helps SMM driver design concentrate on the interrupt handling rather than device initialization or resource management, making it thus faster [75].

IV.3.7.2 Intel SGX Remote Attestation

In this approach, we propose to keep the SMMDecoy SMI simple, and rely on the remote attestation capabilities of Intel SGX to communicate with a remote server. This approach also respects the threat model, since Intel SGX enclaves are hardware protected from the operating system.

Provisioning Key provisioning happens once, and it involves the following steps:

1. Authenticating SMMDecoy to Intel Remote Server.
    - During start-up, the BIOS uses the Intel remote server PKI to establish a secure channel with it.
    - The BIOS computes a token on the SMMDecoy to be loaded into SMRAM and sends its hash signature to verify its integrity and prove its identity.

2. Authenticating SGX enclave to Intel Remote Server, using intel SGX remote attestation

3. The enclave generates a symmetric secret key K which it securely forwards to the IAS, which securely forwards the key to the SMMDecoy SMI on the same platform as the SMMEnclave.

At this point, a unique session has been successfully established.

Communication Once a shared secret key is established between SMMDecoy enclave and SMI, the interrupt is ready to secretly send decoy credentials to the enclave. The enclave then engages in a standard remote attestation protocol and establishes a secure channel with the remote server.

IV.3.7.3 Proposed implementation details

We propose to use Coreboot as a BIOS distribution to implement SMMDecoy on. Coreboot is "an extended firmware platform that delivers a lightning fast and secure boot experience on modern computers and embedded systems. As an Open Source project, it provides auditability and maximum control over technology"[141]. This is important for us to be able to implement and deploy the custom SMMDecoy Interrupt handler into the platform.

IV.4 Related Work

In this section we discuss three lines of related work: GPU malware detection, SMM based systems and security by deception.
In order to detect stealthy GPU malware, prior work suggested monitoring the side effects the malware generates, as CPU solutions are unable to access and scan the GPU. However, these measurements are only reliable in the case of malware which performs bulk DMA transfers, which is not the case for GPU keyloggers. Other work suggests using the cuda-gdb real time debugging capabilities in order to monitor the GPU's access patterns. However, GPU malware could remote debug points from its code base [32].

SMM has been traditionally used to secure the execution of platform management functions such as power and hear control. However, it has been increasingly used to deploy security sensitive solutions, which require strong hardware access control guarantees. Such systems are HyperCheck that is used for hypervisor integrity verification and IOCheck and SMMDumper that scans system memory and dumps it for forensic analysis [106] [141]. Aurora leverages SMM to provide intel SGX enclaves with trusted network and time services, by porting their corresponding drivers into SMM [75]. Researchers have also been long aware of the keyboard sniffing problem. TrustLogin proposes a solution to prevent credentials leakage while they are from the keyboard to the Network Interface Card, NICl. It uses SMM to encrypt the credentials, and forward them securely to the NIC. [145].

Finally, Deception and decoy has always been part of the defence arsenal of cybersecurity. The most widely discussed deception-based solution is arguably honeypots. The intuition behind honeypots is to "provide fake information which is attractive to attackers. The attacker, in searching for the honey of interest comes across the honeypot, and starts to taste of its wares. If they are appealing enough, the attacker spends considerable time and effort getting at the honey provided. If the attacker has finite resources, the time spent going after the honeypot is time not spent going after other things the honeypot is intended to protect. If the attacker uses tools and techniques in attacking the honeypot, some aspects of those tools and techniques are revealed to the defender in the attack on the honeypot" [16]. The earliest and most notable use of honeypots was in 1991 from ATT researcher in a paper called “jail”, which aims to lure attacks in order to monitor their behaviour. Since that time, deception has increasingly been explored as a key technology area for innovation in information protection [136].The idea of honeypots has been further explored more recently, and applied to detect authentication into financial institutions, by creating an account which mimics a real account through all its attributes, minus the fact that it isn’t backed with any money which can actually be stolen. When an attacker gains access to such accounts, it will be indistinguishable to them from any other real account, as they will have access to the same services, except the fact that the bank will not be validating any money transfers linked to the faked account. This constitutes a very efficient solution not only for the detection of account breaches, but also a great opportunity to learn about the behaviour, strategy and intention of attackers [68].

IV.5 Conclusions and Future Work

In this paper, we presented SMMDecoy, a deception-based technique to detect GPU keyloggers, which sniff the open keyboard interface. We protect against a strong adversary which can take control over the platform’s user applications, kernel, and GPU. SMMDecoy generates and injects decoy credentials, which should indistinguishable from legitimate ones. They are then sniffed by the GPU malware. SMMDecoy relies on strong hardware enabled access control mechanisms. It uses SMM to protect the integrity and transparency of the decoy credentials’ injection. It also uses SGX to establish a secure channel to a remote server, over which the injected decoy credentials would be forwarded. If a decoy credential is detected to be used by a malware, an alarm should be raised. Unlike traditional honeywords, an SMMDecoy alarm reveals a wealth of information about how the malware spreads. The paper also discusses the implementation feasibility of SMMDecoy. Future work is to naturally implement the solution and evaluate its performance. We also plan to use SMMDecoy as part of a longitude study in which we aim at detecting GPU, and other firmware, malware which is not possible to detect using traditional CPU based mechanisms.

Acknowledgements

The author would like to thank all of the reviewers. This work is supported by the department of informatics of the university of Oslo, and COINS Research School of Computer and Information Security.

Bibliography

[1] *20th Annual Computer Security Applications Conference (ACSAC 2004), 6-10 December 2004, Tucson, AZ, USA*. IEEE Computer Society, 2004. ISBN: 0-7695-2252-1. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=9473.

[2] Anne Adams and Martina Angela Sasse. "Users Are Not the Enemy". In: *Communications of the ACM* 42.12 (Dec. 1, 1999), pp. 40–46. ISSN: 00010782. DOI: 10.1145/322796.322806. URL: http://portal.acm.org/citation.cfm?doid=322796.322806 (visited on 02/09/2019).

[3] Mohammed Hamad Alzomai and Bander AlFayyadh. "Display Integrity Assurance for SMS Transaction Authorization". In: 2011.

[4] *An Introduction to Qubes OS*. 2018. URL: https://www.qubes-os.org/intro/.

[5] R. J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd Edition*. Wiley Publishing, 2008. 1088 pp. ISBN: 978-0-470-06852-6. URL: https://www.wiley.com/en-us/Security+Engineering3A+A+Guide+to+Building+Dependable+Distributed+Systems2C+2nd+Edition-p-9780470068526 (visited on 02/09/2019).

[6] W. Brian Arthur, David Challener, and Kenneth Goldman. "A Practical Guide to TPM 2.0". In: *Apress*. 2015. DOI: 10.1007/978-1-4302-6584-9.

[7] Will Arthur and David Challener. *A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security*. Apress, 2015.

[8] N. Asokan, Jan-Erik Ekberg, and Kari Kostiainen. "The Untapped Potential of Trusted Execution Environments on Mobile Devices". In: *Financial Cryptography and Data Security*. Ed. by Ahmad-Reza Sadeghi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 293–294. ISBN: 978-3-642-39884-1.

[9] *Attestation Identity Key (AIK) Certificate Enrollment SpecificationFrequently Asked Questions*. Sept. 2011. URL: https://trustedcomputinggroup.org/wp-content/uploads/IWG-AIK-CMC-enrollment-FAQ.pdf.

[10] Andries Brouwer. *The AT Keyboard Controller*. July 7, 2009. URL: https://www.win.tue.nl/~aeb/linux/kbd/scancodes-11.html.

[11] Chris Cain. *Analyzing Man-in-the-Browser (MITB) Attacks*. SANS Institute, 2014, p. 23. URL: https://www.sans.org/reading-room/whitepapers/forensics/analyzing-man-in-the-browser-mitb-attacks-35687.

[12] *Cards and security devices for personal identification — Contactless proximity objects — Part 4: Transmission protocol*. Standard. International Organization for Standardization, 2018.

[13] Huseyin Cavusoglu, Birendra K. Mishra, and Srinivasan Raghunathan. "The Effect of Internet Security Breach Announcements on Market Value: Capital Market Reactions for Breached Firms and Internet Security Developers". In: *International Journal of Electronic Commerce* 9.1 (Oct. 1, 2004), pp. 70–104. ISSN: 1086-4415. DOI: 10.1080/10864415.2004.11044320. URL: https://doi.org/10.1080/10864415.2004.11044320.

[14] Zhiqun Chen. *Java Card Technology for Smart Cards: Architecture and Programmer's Guide*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000. ISBN: 0201703297.

[15] Andy Chou et al. "An Empirical Study of Operating Systems Errors". In: *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*. SOSP '01. Banff, Alberta, Canada: ACM, 2001, pp. 73–88. ISBN: 1-58113-389-8. DOI: 10.1145/502034.502042. URL: http://doi.acm.org/10.1145/502034.502042.

[16] Fred Cohen. "The Use of Deception Techniques : Honeypots and Decoys". In: 2004.

[17] *Consumer Survey: Password Habits. A Study of Password Habits among American Consumers*. 2012. URL: https://www.csid.com/wp-content/uploads/2012/09/CS_PasswordSurvey_FullReport_FINAL.pdf.

[18] *Control Computer Crime News*. 2013.

[19] *Coreboot*. 2018. URL: https://www.coreboot.org/vendors.html.

[20] Victor Costan and Srinivas Devadas. "Intel SGX Explained". In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 86.

[21] *CVE-2017-15126*. Available from MITRE, CVE-ID CVE-2014-0160. 2018. URL: https://nvd.nist.gov/vuln/detail/CVE-2017-15126.

[22] *CVE-2017-18017*. Available from MITRE, CVE-ID CVE-2014-0160. 2018. URL: https://nvd.nist.gov/vuln/detail/CVE-2017-18017\#vulnDescriptionTitle.

[23] *CVE-2018-1000026*. Available from MITRE, CVE-ID CVE-2014-0160. 2018. URL: https://nvd.nist.gov/vuln/detail/CVE-2018-1000026.

[24] *CVE-2018-8822*. Available from MITRE, CVE-ID CVE-2014-0160. 2018. URL: https://nvd.nist.gov/vuln/detail/CVE-2018-8822.

[25] Janis Danisevskis. *Android Protected Confirmation: Taking transaction security to the next level*. June 13, 2012. URL: https://android-developers.googleblog.com/2018/10/android-protected-confirmation.html (visited on 02/09/2019).

[26] Nathan D Dautenhahn. "Protection in commodity monolithic operating systems". PhD thesis. 2016. ISBN: 978-1-267-82489-9.

[27] Brian Delgado and Karen L. Karavanic. "EPA-RIMM: A Framework for Dynamic SMM-based Runtime Integrity Measurement". In: *CoRR* (2018). arXiv: 1805.03755. URL: http://arxiv.org/abs/1805.03755.

[28] Aritra Dhar et al. “IntegriKey: End-to-End Integrity Protection of User Input”. In: *IACR Cryptology ePrint Archive* (2017), p. 1245.

[29] Aritra Dhar et al. “ProximiTEE: Hardened SGX Attestation and Trusted Path through Proximity Verification”. In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 902. URL: https://eprint.iacr.org/2018/902.

[30] Alexandra Dmitrienko et al. “Market-Driven Code Provisioning to Mobile Secure Hardware”. In: *Financial Cryptography and Data Security*. Ed. by Rainer Böhme and Tatsuaki Okamoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 387–404. ISBN: 978-3-662-47854-7.

[31] Nir Drucker and Shay Gueron. “Achieving trustworthy Homomorphic Encryption by combining it with a Trusted Execution Environment”. In: *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* 9 (Mar. 2018), pp. 86–. DOI: 10.22667/JOWUA.2018.03.31.086.

[32] Shawn Embleton, Sherri Sparks, and Cliff Zou. “SMM Rootkits: A New Breed of OS Independent Malware”. In: *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*. SecureComm ’08. Istanbul, Turkey: ACM, 2008, 11:1–11:12. ISBN: 978-1-60558-241-2. DOI: 10.1145/1460877.1460892. URL: http://doi.acm.org/10.1145/1460877.1460892.

[33] Santiago Escobar, Catherine Meadows, and José Meseguer. “Maude-NPA: Cryptographic Protocol Analysis Modulo Equational Properties”. In: *Foundations of Security Analysis and Design V: FOSAD 2007/2008/2009 Tutorial Lectures*. Ed. by Alessandro Aldini, Gilles Barthe, and Roberto Gorrieri. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–50. ISBN: 978-3-642-03829-7. DOI: 10.1007/978-3-642-03829-7_1. URL: https://doi.org/10.1007/978-3-642-03829-7_1.

[34] Saba Eskandarian et al. “Fidelius: Protecting User Secrets from Compromised Browsers”. In: *CoRR* abs/1809.04774 (2018). arXiv: 1809.04774. URL: http://arxiv.org/abs/1809.04774.

[35] Carol Hovenga Fancher. “In Your Pocket: Smartcards”. In: *IEEE Spectr.* 34.2 (Feb. 1997), pp. 47–53. ISSN: 0018-9235. DOI: 10.1109/6.570830. URL: http://dx.doi.org/10.1109/6.570830.

[36] *FIDO Draft Reference Architecture*. 2014. URL: https://fidoalliance.org/wp-content/uploads/2014/12/DraftD-FIDO-Refarch-00.pdf.

[37] *FIDO Security Reference*. Dec. 8, 2014. URL: https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-security-ref-v1.0-ps-20141208.html.

[38] Dinei Florencio and Cormac Herley. “A Large-Scale Study of Web Password Habits”. In: *Proceedings of the 16th International Conference on World Wide Web* (Banff, Alberta, Canada). WWW ’07. New York, NY, USA: ACM, May 2007, pp. 657–666. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242661. URL: http://doi.acm.org/10.1145/1242572.1242661 (visited on 02/09/2019).

[39] Dinei Florencio and Cormac Herley. "A Large-scale Study of Web Password Habits". In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. Banff, Alberta, Canada: ACM, 2007, pp. 657–666. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242661. URL: http://doi.acm.org/10.1145/1242572.1242661.

[40] *Google Launches Security Key, World's First Deployment of Fast Identity Online Universal Second Factor (FIDO U2F) Authentication*. Oct. 21, 2014. URL: https://fidoalliance.org/google-launches-security-key-worlds-first-deployment-of-fast-identity-online-universal-second-factor-fido-u2f-authentication/ (visited on 02/09/2019).

[41] *Google, Samsung, and 16 Others Receive Post-Password Certification - FIDO Alliance*. May 21, 2015. URL: https://fidoalliance.org/google-samsung-and-16-others-receive-post-password-certification/ (visited on 02/09/2019).

[42] Mark Hatton. *The Human Side of Security*. May 21, 2013. URL: https://www.securityweek.com/human-side-security.

[43] Warren He et al. "ShadowCrypt: Encrypted Web Applications for Everyone". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. CCS '14. Scottsdale, Arizona, USA: ACM, 2014, pp. 1028–1039. ISBN: 978-1-4503-2957-6.

[44] Paul Heckbert. "Color Image Quantization for Frame Buffer Display". In: *SIGGRAPH Comput. Graph.* 3 (July 1982), pp. 297–307. ISSN: 0097-8930.

[45] C. Herley. "More Is Not the Answer". In: *IEEE Security Privacy* 12.1 (Jan. 2014), pp. 14–19. ISSN: 1540-7993. DOI: 10.1109/MSP.2013.134.

[46] Cormac Herley. "So Long, and No Thanks for the Externalities: The Rational Rejection of Security Advice by Users". In: *Proceedings of the 2009 Workshop on New Security Paradigms Workshop* (Oxford, United Kingdom). NSPW '09. New York, NY, USA: ACM, Sept. 2009, pp. 133–144. ISBN: 978-1-60558-845-2. DOI: 10.1145/1719030.1719050. URL: http://doi.acm.org/10.1145/1719030.1719050 (visited on 02/09/2019).

[47] Cormac Herley. "So Long, and No Thanks for the Externalities: The Rational Rejection of Security Advice by Users". In: *Proceedings of the 2009 Workshop on New Security Paradigms Workshop*. NSPW '09. Oxford, United Kingdom: ACM, 2009, pp. 133–144. ISBN: 978-1-60558-845-2. DOI: 10.1145/1719030.1719050. URL: http://doi.acm.org/10.1145/1719030.1719050.

[48] Thorsten Holz, Markus Engelberth, and Felix Freiling. "Learning More About the Underground Economy: A Case-study of Keyloggers and Dropzones". In: *Proceedings of the 14th European Conference on Research in Computer Security*. ESORICS'09. Saint-Malo, France: Springer-Verlag, 2009, pp. 1–18. ISBN: 3-642-04443-3, 978-3-642-04443-4. URL: http://dl.acm.org/citation.cfm?id=1813084.1813086.

[49] *https://github.com/torvalds/linux*. 2019. URL: https://www.heise.de/newsticker/meldung/Linux-Kernel-durchbricht-die-20-Millionen-Zeilen-Marke-2730780.html.

[50] Thomson Iain. *How the NSA Hacks PCs, phones, routers, hard disks*. Dec. 31, 2013. URL: https://www.theregister.co.uk/2013/12/31/nsa_weapons_catalogue_promises_pwnage_at_the_speed_of_light/.

[51] *Identification cards – Integrated circuit cards – Part 1: Cards with contacts – Physical characteristics*. Standard. International Organization for Standardization, Feb. 2011.

[52] Loutfi Ijlal and Jøsang Audun. "Privacy Concerns of TPM 2.0". In: ECCWS. Munich, Germany, 2016, pp. 205–211. ISBN: 978-1-4503-3245-3.

[53] *Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model*. Standard. International Organization for Standardization, Dec. 2009.

[54] *Integrity Considerations for Secure Computer Systems*. Standard. MTR-3154, MITRE., 1977.

[55] *Intel® 64 and IA-32 Architectures Software Developer's Manual*. Specifications. Intel, Oct. 2016. URL: https://software.intel.com/en-us/articles/intel-sdm.

[56] *Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3A: System Programming Guide, Part 1*. Intel. Sept. 2016.

[57] *Intel SGX Remote Attestation*. 2019. URL: https://qiita.com/Cliffford/items/095b1df450583b4803f2 (visited on 02/09/2019).

[58] *Introduction to Secure Elements*. Technical Report. Global Platform, 2018. URL: https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Secure-Element-15May2018.pdf.

[59] Alon Jackson. "Trust is in the Keys of the Beholder:Extending SGX Autonomy and Anonymity". MA thesis. Herzliya, Israel: Efi Arazi School of Computer Science School, the Interdisciplinary Center, Herzliya, 2017.

[60] Robert J. K. Jacob. "Human-computer Interaction: Input Devices". In: *ACM Comput. Surv.* 1 (Mar. 1996), pp. 177–179. ISSN: 0360-0300. DOI: 10.1145/234313.234387. URL: http://doi.acm.org/10.1145/234313.234387.

[61] Bhushan Jain et al. "SoK: Introspections on Trust and the Semantic Gap". In: *2014 IEEE Symposium on Security and Privacy* (2014), pp. 605–620.

[62] Ekberg JanErik. "Securing Software Architectures for Trusted Processor Environments". PhD thesis. 2013. ISBN: 978-952-60-3632-8.

[63] Yeongjin Jang. "Building Trust In The User IO In Computer Systems". PhD thesis. 2017.

[64] Trevor Jim et al. "Cyclone: A Safe Dialect of C." In: *USENIX Annual Technical Conference, General Track*. 2002, pp. 275–288.

[65] Audun Jøsang. “Identity Management and Trusted Interaction in Internet and Mobile Computing”. In: *IET Information Security* 2 (Nov. 26, 2013), pp. 67–79. ISSN: 1751-8717. DOI: 10.1049/iet-ifs.2012.0133. URL: https://digital-library.theiet.org/content/journals/10.1049/iet-ifs.2012.0133 (visited on 02/09/2019).

[66] Audun Jøsang et al. “Local User-Centric Identity Management”. In: *Journal of Trust Management* 2.1 (Jan. 8, 2015), p. 1. ISSN: 2196-064X. DOI: 10.1186/s40493-014-0009-6. URL: https://doi.org/10.1186/s40493-014-0009-6 (visited on 02/09/2019).

[67] Audun Jøsang et al. “Trust Requirements in Identity Management”. In: *Proceedings of the 2005 Australasian Workshop on Grid Computing and E-Research - Volume 44* (Newcastle, New South Wales, Australia). ACSW Frontiers ’05. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2005, pp. 99–108. ISBN: 978-1-920682-26-2. URL: http://dl.acm.org/citation.cfm?id=1082290.1082305 (visited on 02/09/2019).

[68] Ari Juels and Ronald L. Rivest. “Honeywords: Making Password-cracking Detectable”. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer ; Communications Security*. CCS ’13. Berlin, Germany: ACM, 2013, pp. 145–160. ISBN: 978-1-4503-2477-9. DOI: 10.1145/2508859.2516671. URL: http://doi.acm.org/10.1145/2508859.2516671.

[69] Radhesh Krishnan Konoth, Victor van der Veen, and Herbert Bos. “How Anywhere Computing Just Killed Your Phone-Based Two-Factor Authentication”. In: *Financial Cryptography*. Vol. 9603. Lecture Notes in Computer Science. Springer, 2016, pp. 405–421.

[70] Evangelos Ladakis et al. “You Can Type , but You Can ’ t Hide : A Stealthy GPU-based Keylogger”. In: 2013.

[71] Evangelos Ladakis et al. “You Can Type, but You Can’t Hide: A Stealthy GPU-Based Keylogger”. In: 2013.

[72] Susan Landau, Hubert Le Van Gong, and Robin Wilton. “Achieving Privacy in a Federated Identity Management System”. In: *Financial Cryptography and Data Security*. Ed. by Roger Dingledine and Philippe Golle. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 51–70. ISBN: 978-3-642-03549-4.

[73] *Ledger-Personal Security Devices*. 2019. URL: https://ledger.readthedocs.io/en/latest/background/personal_security_devices.html.

[74] Ninghui Li, Ziqing Mao, and Hong Chen. “Usable Mandatory Integrity Protection for Operating Systems”. In: *2007 IEEE Symposium on Security and Privacy (SP ’07)* (2007), pp. 164–178.

[75] Hongliang Liang et al. “Aurora: Providing Trusted System Services for Enclaves On an Untrusted System”. In: *CoRR* (2018). arXiv: 1802.03530. URL: http://arxiv.org/abs/1802.03530.

[76] Hongliang Liang et al. “Aurora: Providing Trusted System Services for Enclaves On an Untrusted System”. In: *CoRR* abs/1802.03530 (2018). URL: http://arxiv.org/abs/1802.03530.

[77] Moritz Lipp et al. “Meltdown: Reading Kernel Memory from User Space”. In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018, pp. 973–990. ISBN: 978-1-931971-46-1. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/lipp.

[78] Ijlal Loutfi. “SMMDecoy: Detecting GPU Keyloggers using Security by Deception Techniques”. In: Jan. 2019, pp. 580–587. DOI: 10.5220/0007578505800587.

[79] Ijlal Loutfi and Audun Josang. “Passwords Are Not Always Stronger on the Other Side of the Fence”. In: *Proceedings 2015 Workshop on Usable Security*. Workshop on Usable Security. San Diego, CA: Internet Society, 2015. ISBN: 978-1-891562-40-2. DOI: 10.14722/usec.2015.23005. URL: https://www.ndss-symposium.org/ndss2015/ndss-2015-usec-programme/passwords-are-not-always-stronger-other-side-fence (visited on 02/10/2019).

[80] Ijlal Loutfi and Audun Jøsang. “1,2, Pause: Lets Start by Meaningfully Navigating the Current Online Authentication Solutions Space”. In: *Trust Management IX*. Ed. by Christian Damsgaard Jensen et al. IFIP Advances in Information and Communication Technology. Springer International Publishing, 2015, pp. 165–176. ISBN: 978-3-319-18491-3.

[81] Ijlal Loutfi and Audun Jøsang. “1,2, Pause: Lets Start by Meaningfully Navigating the Current Online Authentication Solutions Space”. In: *Trust Management IX*. Ed. by Christian Damsgaard Jensen et al. IFIP Advances in Information and Communication Technology. Springer International Publishing, 2015, pp. 165–176. ISBN: 978-3-319-18491-3.

[82] Ijlal Loutfi and Audun Jøsang. “FIDO Trust Requirements”. In: Jan. 2015, pp. 139–155. ISBN: 978-3-319-26501-8. DOI: 10.1007/978-3-319-26502-5_10.

[83] Vincent Maraia. “The Build Master: Microsoft’s Software Configuration Management Best Practices”. In: (Jan. 2005).

[84] Margaret Rouse. *Man In The Browser Attack*. 2017. URL: https://searchstorage.techtarget.com/definition/video-RAM (visited on 02/09/2019).

[85] Keith Marzullo et al. “Tools for distributed application management”. In: *Computer* 24.8 (1991), pp. 42–51.

[86] Brian McGillion et al. “Open-TEE – An Open Virtual Trusted Execution Environment”. In: *Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA - Volume 01*. TRUSTCOM ’15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 400–407. ISBN: 978-1-4673-7952-6. DOI: 10.1109/Trustcom.2015.400. URL: http://dx.doi.org/10.1109/Trustcom.2015.400.

[87] Mike. *Operating Systems Development - Keyboard*. 2009. URL: http://www.brokenthorn.com/Resources/OSDev19.html.

[88] *Minimum Security Requirements for Federal Information and Information Systems*. Standard. National Institute of Standards and Technology, Mar. 2006.

[89] *MULTOS-Security Architecture*. 2019. URL: https://www.multos.com/technology/security_architecture.

[90] Aleksandr Ometov et al. "Multi-Factor Authentication: A Survey". In: *Cryptography* 2 (Jan. 2018). DOI: 10.3390/cryptography2010001.

[91] *Online Identity Group Gains Traction with Government Involvement*. June 18, 2015. URL: https://fidoalliance.org/online-identity-group-gains-traction-with-government-involvement/ (visited on 02/09/2019).

[92] Stefano Ortolani and Bruno Crispo. "NoisyKey: Tolerating Keyloggers via Keystrokes Hiding". In: *Proceedings of the 7th USENIX Conference on Hot Topics in Security* (Bellevue, WA). HotSec'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 2–2. URL: http://dl.acm.org/citation.cfm?id=2372387.2372389.

[93] Stefano Ortolani and Bruno Crispo. "NoisyKey: Tolerating Keyloggers via Keystrokes Hiding". In: *Proceedings of the 7th USENIX Conference on Hot Topics in Security*. HotSec'12. Bellevue, WA: USENIX Association, 2012, pp. 2–2. URL: http://dl.acm.org/citation.cfm?id=2372387.2372389.

[94] Stefano Ortolani, Cristiano Giuffrida, and Bruno Crispo. "Bait Your Hook: A Novel Detection Technique for Keyloggers". In: *Recent Advances in Intrusion Detection*. Ed. by Somesh Jha, Robin Sommer, and Christian Kreibich. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 198–217. ISBN: 978-3-642-15512-3.

[95] Thomas J. Ostrand and Elaine J. Weyuker. "The Distribution of Faults in a Large Industrial Software System". In: *Proceedings of the 2002 ACM SIGSOFT International Symposium on Software Testing and Analysis*. ISSTA '02. Roma, Italy: ACM, 2002, pp. 55–64. ISBN: 1-58113-562-9. DOI: 10.1145/566172.566181. URL: http://doi.acm.org/10.1145/566172.566181.

[96] Gopalakrishna Palem. *Capturing the Screen*. June 13, 2012. URL: https://gk.palem.in/screencap.html (visited on 02/09/2019).

[97] *PandaLabs Quarterly Report*. 2012. URL: https://www.pandasecurity.com/mediacenter/src/uploads/2012/08/Quarterly-Report-PandaLabs-April-June-2012.pdf.

[98] *Personal Security Devices*. June 13, 2012. URL: https://ledger.readthedocs.io/en/latest/background/personal_security_devices.html (visited on 02/09/2019).

[99] *Poor Password Security among Norwegians*. 2013. URL: https://www.evry.no/bedrift/investor/bors-ogpressemeldinger/evry-darlig-passordsikkerhet-hos-nordmenn-1867810/.

[100] D. Prabhu, M. Adimoolam, and P. Saravannan. "A Novel DNA Based Encrypted Text Compression". In: *IJCA Special Issue on Network Security and Cryptography* NSC.2 (Dec. 2011), pp. 36–41. DOI: 10.5120/4332-025. URL: http://research.ijcaonline.org/nsc/number2/SPE025T.pdf.

[101] Niels Provos et al. "The Ghost in the Browser Analysis of Web-based Malware". In: *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*. HotBots'07. Cambridge, MA: USENIX Association, 2007, pp. 4–4.

[102] Jean-Jacques Quisquater. "The Adolescence of Smart Cards". In: *Future Gener. Comput. Syst.* 13.1 (July 1997), pp. 3–7. ISSN: 0167-739X. DOI: 10.1016/S0167-739X(97)89108-X. URL: http://dx.doi.org/10.1016/S0167-739X(97)89108-X.

[103] Samani Raj. *Cybercrime Exposed Cybercrime-as-a-Service*. 2013. URL: https://www.networksasia.net/article/cybercrime-exposed-cybercrime-service-1372901941.

[104] Ramesh Kesanupalli. *China's Online Giant Alibaba Endorses FIDO Authentication*. Oct. 1, 2014. URL: https://fidoalliance.org/chinas-online-giant-alibaba-endorses-fido-authentication/.

[105] Alessandro Reina et al. "When Hardware Meets Software: A Bulletproof Solution to Forensic Memory Acquisition". In: *Proceedings of the 28th Annual Computer Security Applications Conference* (Orlando, Florida, USA). ACSAC '12. New York, NY, USA: ACM, 2012, pp. 79–88. ISBN: 978-1-4503-1312-4. DOI: 10.1145/2420950.2420962. URL: http://doi.acm.org/10.1145/2420950.2420962.

[106] Alessandro Reina et al. "When Hardware Meets Software: A Bulletproof Solution to Forensic Memory Acquisition". In: *Proceedings of the 28th Annual Computer Security Applications Conference*. ACSAC '12. Orlando, Florida, USA: ACM, 2012, pp. 79–88. ISBN: 978-1-4503-1312-4. DOI: 10.1145/2420950.2420962. URL: http://doi.acm.org/10.1145/2420950.2420962.

[107] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. "Trusted Execution Environment: What It is, and What It is Not". In: Aug. 2015, pp. 57–64. DOI: 10.1109/Trustcom.2015.357.

[108] *SafeNet Announces Results of Second Annual Global Password Survey; 2004 Results Indicate Increased Concern by Enterprise Management; Employees Not Diligent Enough*. Mar. 7, 2005. URL: https://www.businesswire.com/news/home/20050307005147/en/SafeNet-Announces-Results-Annual-Global-Password-Survey.

[109] Joshua Serratelli Schiffman. "Practical System Integrity Verification in Cloud Computing Environments". AAI3534761. PhD thesis. University Park, PA, USA, 2012. ISBN: 978-1-267-82489-9.

[110] Michael S. Schmidt. *Credit Card Data Breach at Barnes Noble Stores*. Oct. 23, 2012. URL: https://www.nytimes.com/2012/10/24/business/hackers-get-credit-data-at-barnes-noble.html?_r=3\&.

[111] Michael S. Schmidt. *Credit Card Data Breach at Barnes Noble Stores*. Oct. 23, 2012. URL: https://www.nytimes.com/2012/10/24/business/hackers-get-credit-data-at-barnes-noble.html?_r=3\&.

[112] Anne G. Scott and Lee Sechrest. "Survey Research and Response Bias". In: *Proceedings of the Survey Research Methods Section*. American Statistical Association, 1993, pp. 238–243. URL: http://www.asasrms.org/Proceedings/papers/1993_036.pdf.

[113] Secure Box. *Man In The Browser Attack*. URL: https://securebox.comodo.com/ssl-sniffing/man-in-the-browser-attack/ (visited on 02/09/2019).

[114] *Security Architecture for Open Systems Interconnection for CCITT Applications*. Recommendation X.800. 1991. URL: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.800-199103-I!!PDF-E&type=items.

[115] Jianxiong Shao et al. "Formal Analysis of Enhanced Authorization in the TPM 2.0". In: *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*. ASIA CCS '15. Singapore, Republic of Singapore: ACM, 2015, pp. 273–284. ISBN: 978-1-4503-3245-3. DOI: 10.1145/2714576.2714610. URL: http://doi.acm.org/10.1145/2714576.2714610.

[116] Weisong Shi et al. "Edge computing: Vision and challenges". In: *IEEE Internet of Things Journal* 3.5 (2016), pp. 637–646.

[117] Elizabeth Stobert and Robert Biddle. "The Password Life Cycle: User Behaviour in Managing Passwords". In: *Proceedings of the Tenth USENIX Conference on Usable Privacy and Security* (Menlo Park, CA). SOUPS'14. Berkeley, CA, USA: USENIX Association, July 2014, pp. 243–255. ISBN: 978-1-931971-13-3. URL: http://dl.acm.org/citation.cfm?id=3235838.3235860 (visited on 02/09/2019).

[118] M. Szydlowski, C. Kruegel, and E. Kirda. "Secure Input for Web Applications". In: *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*. 2007, pp. 375–384. DOI: 10.1109/ACSAC.2007.28.

[119] Frederick T. Grampp and Robert H. Morris. "The UNIX system UNIX operating system security". In: *AT&T Bell Laboratories Technical Journal* 63 (Oct. 1984), pp. 1649–1672. DOI: 10.1002/j.1538-7305.1984.tb00058.x.

[120] Richard Ta-Min, Lionel Litty, and David Lie. "Splitting Interfaces: Making Trust Between Applications and Operating Systems Configurable". In: *OSDI*. 2006.

[121] Sandeep Tamrakar. *Applications of Trusted Execution Environments (TEEs)*. en. G5 Artikkeliväitöskirja. 2017. URL: http://urn.fi/URN:ISBN:978-952-60-7463-4.

[122] *TCG Specification Architecture Overview*. Standard. Trusted Computing Group, 2007.

[123] *The Evolution of Authentication*. Tech Report. FIDO Alliance, Mar. 2013.

[124] *The FIDO Alliance: Privacy Principles Whitepaper*. Feb. 2014. URL: https://fidoalliance.org/wp-content/uploads/2014/12/FIDO_Alliance_Whitepaper_Privacy_Principles.pdf.

[125] Kurt Thomas et al., eds. *Data breaches, phishing, or malware? Understanding the risks of stolen credentials*. 2017.

[126] Joaquin Torres, Antonio Izquierdo, and Jose Maria Sierra. "Advances in Network Smart Cards Authentication". In: *Comput. Netw.* 51.9 (June 2007), pp. 2249–2261. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2007.01.010. URL: http://dx.doi.org/10.1016/j.comnet.2007.01.010.

[127] William M. Trochim. *The Research Methods Knowledge Base, 2nd Edition*. (version current as of October 20, 2006). 2006. URL: http://www.socialresearchmethods.net/kb/.

[128] *Trusted Computer System Evaluation Criteria*. 1985.

[129] Chia che Tsai, Donald E. Porter, and Mona Vij. "Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX". In: *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. Santa Clara, CA: USENIX Association, 2017, pp. 645–658. ISBN: 978-1-931971-38-6.

[130] Jo Van Bulck, Frank Piessens, and Raoul Strackx. "SGX-Step: A Practical Attack Framework for Precise Enclave Execution Control". In: *Proceedings of the 2Nd Workshop on System Software for Trusted Execution*. SysTEX'17. Shanghai, China: ACM, 2017, 4:1–4:6. ISBN: 978-1-4503-5097-6. DOI: 10.1145/3152701.3152706. URL: http://doi.acm.org/10.1145/3152701.3152706.

[131] *Various methods for capturing the screen*. 2006. URL: https://www.codeproject.com/Articles/5051/Various-methods-for-capturing-the-screen (visited on 02/09/2019).

[132] Kent A. Varmedal et al. "The OffPAD: Requirements and Usage". In: *NSS*. 2013. DOI: 10.1007/978-3-642-38631-2_7.

[133] Amit Vasudevan et al. "Trustworthy Execution on Mobile Devices: What Security Properties Can My Mobile Platform Give Me?" In: *TRUST*. 2011.

[134] Verizon. *Control Computer Crime News*. 2009.

[135] Jennifer L. Waller. "How to Perform and Interpret Chi-Square and T-Tests". In: SAS Global Forum. 2012, p. 155. URL: https://support.sas.com/resources/papers/proceedings12/155-2012.pdf.

[136] Ding Wang et al. "A Security Analysis of Honeywords". In: *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. 2018.

[137] Samuel Weiser and Mario Werner. “SGXIO: Generic Trusted I/O Path for Intel SGX”. In: *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. CODASPY ’17. Scottsdale, Arizona, USA: ACM, 2017, pp. 261–268. ISBN: 978-1-4503-4523-1.

[138] *Windows Hello Waves Off Passwords*. May 1, 2015. URL: https://fidoalliance.org/windows-hello-waves-off-passwords-2/ (visited on 02/09/2019).

[139] Johannes Winter. “Trusted computing building blocks for embedded linux-based ARM trustzone platforms”. In: *Proceedings of the 3rd ACM workshop on Scalable trusted computing*. ACM. 2008, pp. 21–30.

[140] Zishuang (Eileen) Ye, Sean Smith, and Denise Anthony. “Trusted Paths for Browsers”. In: *ACM Trans. Inf. Syst. Secur.* 8.2 (May 2005), pp. 153–186. ISSN: 1094-9224. DOI: 10.1145/1065545.1065546. URL: http://doi.acm.org/10.1145/1065545.1065546.

[141] F. Zhang. “IOCheck: A framework to enhance the security of I/O devices at runtime”. In: *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*. 2013, pp. 1–4. DOI: 10.1109/DSNW.2013.6615523. URL: doi.ieeecomputersociety.org/10.1109/DSNW.2013.6615523.

[142] F. Zhang et al. “Using Hardware Features for Increased Debugging Transparency”. In: *2015 IEEE Symposium on Security and Privacy (SP)*. 2015, pp. 55–69. DOI: 10.1109/SP.2015.11. URL: doi.ieeecomputersociety.org/10.1109/SP.2015.11.

[143] Fengwei Zhang. “IOCheck: A framework to enhance the security of I/O devices at runtime”. In: June 2013, pp. 1–4. DOI: 10.1109/DSNW.2013.6615523.

[144] Fengwei Zhang et al. “TrustLogin: Securing Password-Login on Commodity Operating Systems”. In: *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security* (Singapore, Republic of Singapore). ASIA CCS ’15. New York, NY, USA: ACM, 2015, pp. 333–344. ISBN: 978-1-4503-3245-3. DOI: 10.1145/2714576.2714614. URL: http://doi.acm.org/10.1145/2714576.2714614.

[145] Fengwei Zhang et al. “TrustLogin: Securing Password-Login on Commodity Operating Systems”. In: *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*. ASIA CCS ’15. Singapore, Republic of Singapore: ACM, 2015, pp. 333–344. ISBN: 978-1-4503-3245-3. DOI: 10.1145/2714576.2714614. URL: http://doi.acm.org/10.1145/2714576.2714614.