1 Fidelius: Protecting User Secrets from Compromised Browsers

1.1 Introduction

1.1.1 Fidelius

Generally speak- ing, once malware infects the user's machine, it can effectively steal all user data entered into the browser. Modern browsers have responded with a variety of defenses aimed at ensuring browser integrity. However, once the machine is compromised, there is little that the browser can do to protect user data from a key logger.
Fidelius, that helps web sites ensure that user data entered into the browser cannot be stolen by end-user malware, no matter how deeply the malware is embedded into the system. When using Fidelius, users can safely enter data into the browser without fear of it being stolen by mal- 1ware, provided that the hardware enclave we use satisfies the security requirements.

1.1.2 Contributions

- The design of Fidelius, a system for protecting user secrets entered into a browser in a fully-compromised environment.
- A simple interface for web developers to enable Fi- delius's security features.
- The first open design and implementation of a trusted path enabling a hardware enclave to interact with I/O devices such as a display and a keyboard from a fully compromised machine.
- A browser component that enables a hardware enclave to interact with protected DOM elements while keep- ing the enclave component small.
- An open-source implementation and evaluation of Fi- delius for practical use cases.

1.2 Background

1.2.1 Thread Model

We assume that our attacker has the power to ex- amine and modify unprotected memory, communication with peripherals/network devices, and communication be- tween the trusted and untrusted components of the system. Moreover, it can maliciously interrupt the execution of an enclave. Note that an OS-level attacker can always launch an indefinite denial of service attack against an enclave, but such an attack does not compromise privacy.

1.2.2 Architecture

The goal of Fidelius is to establish a trusted path between a user and the remote server behind a web application. To achieve this goal, Fidelius relies on two core components: a trusted user I/O path and a web enclave. In practice, this involve subsystems for a secure keyboard, a secure video display, a browser component to interact with a hardware enclave, and the enclave itself.

1.3 Design

1.3.1 Trusted IO Path

The trusted user I/O path consists of a keyboard and display with a trusted dongle placed between them and the computer running Fidelius. Each device consists of trusted and untrusted modes. The untrusted modes op- erate exactly the same as in an unmodified system. The trusted keyboard mode, when activated, sends a constant stream of encrypted keystrokes to the enclave. The en- clave decrypts and updates the state of the relevant trusted input field. The trusted and untrusted display modes are active in parallel, and the trusted mode consists of a series of overlays sent encrypted from the enclave to the dis- play. Overlays include rendered DOM subtrees (includ- ing, if any, the protected user inputs) placed over the un- trusted display output as well as a dedicated portion of the screen inaccessible to untrusted content.

1.3.2 Web Enclaves

A web enclave is essentially a hardware enclave running a minimalistic, trusted browser engine bound to a single web origin. A browser using a web enclave delegates the management and rendering of portions of a DOM tree and the execution of client-side scripts, e.g. JavaScript and Web Assembly, to the enclave. In addition, the web enclave can send and receive encrypted messages to and from trusted devices and the origin server. Finally, the web enclave provides client-side script APIs to access the DOM subtree, secure storage, and secure HTTP commu- nication.

1.3.3 Trusted IO Setup

In order to securely communicate, the web enclave and peripherals (or the dongles connected to them) must have a shared key. One option is to operate in a threat model with an initial trusted phase where we assume the computer is not yet compromised. Pre-shared keys are exchanged when the user configures the computer for the first time. Devices store the key in an internal memory, and the enclave seals the shared keys for future retrieval. The key can be accessed only by the enclave directly and not by user-provided JavaScript running inside it.

1.3.4 IO Secured Communication

The process of switching between trusted and untrusted modes presents an interesting security challenge. An authentication procedure between the enclave and the trusted devices can ensure that only the enclave initiates switches between trusted and untrusted modes, but this ignores the larger problem that the enclave must rely on the untrusted OS to inform it when an event has happened that necessitates switching modes. Avoiding that necessity would require moving a prohibitively large fraction of the browser and UI into an enclave.

1.3.5 Security Analysis

- Enclave Ommission Attack
- Enclave Missuse Attack
- Page Tampering Attack
- Redirection Attack
- Storage Tampering Attack
- Mode Switching Attack
- Replay Attack
- Input Manipulation Attack
- Timing Attack
- Multi Enclave Attack

1.4 Evaluation

1.4.1 Performance

We evaluate Fidelius in order to determine whether the overheads introduced by the trusted I/O path and web enclave are acceptable for common use cases and find that Fidelius outperforms display latency on some recent commercial devices by as much as 2.8x and prior work by 13.3x. Moreover, communication between the browser and enclave introduces a delay of less than 40ms to page load time for a login page.

1.4.2 Trusted Code Base

The trusted code base for Fidelius consists of 8,450 lines of C++ code, of which about 3200 are libraries for handling form rendering and another 3800 are our enclave port of tiny-js. This does not include native code running outside the enclave or in the browser extension because our security guarantees hold even if an attacker could compromise those untrusted components of the system.