Fidelius: Protecting User Secrets from Compromised Browsers

Saba Eskandarian, Jonathan Cogan, Sawyer Birnbaum, Peh Chang Wei Brandon, Dillon Franke, Forest Fraser, Gaspar Garcia, Eric Gong, Hung T. Nguyen, Taresh K. Sethi, Vishal Subbiah, Michael Backes, Giancarlo Pellegrino, Dan Boneh
In Browsers we Trust
In Browsers we Trust
Modern browsers provide powerful client-side protection mechanisms to safeguard user data.
Modern browsers provide powerful client-side protection mechanisms to safeguard user data.
Modern browsers provide powerful client-side protection mechanisms to safeguard user data.

But if a browser or even the whole OS are compromised,
Modern browsers provide powerful client-side protection mechanisms to safeguard user data.

But if a browser or even the whole OS are compromised, none of these defenses can prevent an attacker from stealing data.
Hardware Enclaves

A trusted component in an untrusted system

- Uses protected memory to isolate enclave execution from compromised OS
- Proves that it is an authentic enclave running the desired code with *attestation*
- Enclaves in our implementation use Intel SGX
Hardware Enclaves

A trusted component in an untrusted system

- Uses protected memory to isolate enclave execution from compromised OS
- Proves that it is an authentic enclave running the desired code with *attestation*
- Enclaves in our implementation use Intel SGX

* Attacks on current SGX architecture (Foreshadow, SGXpectre, etc) -- could implement Fidelius on any enclave, not just SGX.
1. Enclave needs untrusted OS to interact with user, devices, and other software, to use network, etc.
Challenges

2. Browsers have a LOT of code and many bugs/vulnerabilities.
Challenges

2. Browsers have a LOT of code and many bugs/vulnerabilities.

Vulnerable code in enclave → super-malware!
Introducing Fidelius

Goal: protect user keyboard inputs to browser from fully compromised OS
Introducing Fidelius

Keeps browser outside of hardware enclave
Introducing Fidelius

Support for HTML forms, simple JavaScript, local storage, and XmlHttpRequests
Introducing Fidelius

Trusted path from enclave to secure I/O devices
Introducing Fidelius

UI designed to alert user of secure components
Introducing Fidelius

Minimizes overhead due to trusted components
Introducing Fidelius

Minimal changes for developers

```html
<br />
<script type="text/JavaScript" src="validator.js">
console.log("Secure " + "Sign:" + sign);
</script>
```
Video Demo

https://crypto.stanford.edu/fidelius
Backup Image for Video Demo
Outline

Introducing Fidelius
Related Projects
Security Goals and Non-Goals
Fidelius Design
  - User Interface
  - Developer Interface
  - Page load behavior
  - Trusted I/O Path
  - Web Enclave Features
Our Prototype & Performance Evaluation
Related Projects

Enclaves and Trusted I/O

Bumpy (McCune, Perrig, and Reiter, NDSS 2007)

SGXIO (Weiser and Werner, CODASPY 2017)

SGX-USB (Jang, Dissertation, 2017)

Enclaves and the Web:

TrustJS (Goltzsche et al, EUROSEC 2017)

Node-SecureWorker (mitar, wh0, panpan2, cvan, Git Repository)
What Fidelius Does

Trusted user inputs never revealed to attacker with full control of OS and network

Authentic data only sent/received from authentic server

Communication between enclave and I/O devices protected from tampering
What Fidelius Does Not Do

Secure hardware enclave against side-channel attacks

Secure data once it arrives at the server

Protect against malicious JavaScript sent from server

Prevent DoS attack from malicious OS against Fidelius system
Fidelius for Users

Security indicator lights for keyboard and display

Green overlay verifies who gets data and what data you are giving
Fidelius for Users

Security indicator lights for keyboard and display

Green overlay verifies who gets data and what data you are giving

Note: security relies on users watching indicators

Potential UI Attacks

**Enclave omission attack:** malicious OS pretends to use enclave when it really doesn’t

**Mode switching attack:** malicious OS forces exit from trusted mode
Potential UI Attacks

**Enclave omission attack:** malicious OS pretends to use enclave when it really doesn’t

**Mode switching attack:** malicious OS forces exit from trusted mode

**Page reorganization attack:** untrusted components moved to mislead user to enter data incorrectly or in wrong place

**Redirection attack:** data sent to unexpected destination (e.g. phishing)
Fidelius for Developers

Goal
Minimal changes, backwards compatible

HTML
<form> and <script> tags get new optional attributes secure and sign

Javascript
Secure scripts run in separate scope
Access to trusted form inputs, trusted storage, etc
Fidelius Page Load Behavior

1. Browser extension scans page for **secure** attributes, activates enclave, and sends it list of secure components, if any found
Fidelius Page Load Behavior

2. Enclave performs remote attestation if needed, verifies remote origin which will receive data (derived from form action or current page)
Fidelius Page Load Behavior

3. Enclave verifies signatures on secure components, stores form contents and scripts, begins rendering overlays. Secure display indicator LED lights
Fidelius Page Load Behavior

4. User interacts with page as usual, using secure forms and scripts as needed and watching indicator LEDs/secure overlay bar for pertinent information.
Trusted User I/O Path

Each device has a trusted/untrusted mode

Untrusted mode behaves like normal, unmodified device

Trusted mode control & functionality built into external dongles in our system

Deployment could use dongles or new secure keyboards/displays
Trusted User I/O Path

Setup

Need shared key between enclave and trusted keyboard/display

Option 1: one-time setup before compromise

Option 2: mutual attestation between enclave and trusted components in devices
Trusted User I/O Path

Trusted keyboard sends enclave constant stream of encrypted keys

Forces short delay before exiting trusted mode
Trusted User I/O Path

Trusted display receives encrypted (pre-rendered) overlays to decrypt and place on screen.
Potential Attacks on Trusted Path

**Enclave omission attack:** malicious OS only pretends to use enclave
Potential Attacks on Trusted Path

**Enclave omission attack:** malicious OS only pretends to use enclave

**Defense:** shared keys or attestation between trusted devices and enclave
Potential Attacks on Trusted Path

**Enclave omission attack:** malicious OS only pretends to use enclave

**Rapid mode switching attack:** malicious OS rapidly flips between trusted/untrusted modes to steal some data
Potential Attacks on Trusted Path

**Enclave omission attack**: malicious OS only pretends to use enclave

**Rapid mode switching attack**: malicious OS rapidly flips between trusted/untrusted modes to steal some data

**Defense**: short delay when exiting trusted mode
Potential Attacks on Trusted Path

**Enclave omission attack:** malicious OS only pretends to use enclave

**Rapid mode switching attack:** malicious OS rapidly flips between trusted/untrusted modes to steal some data

**Timing side-channel attack:** learn secrets from frequency of key presses or display updates
Potential Attacks on Trusted Path

**Enclave omission attack**: malicious OS only pretends to use enclave

**Rapid mode switching attack**: malicious OS rapidly flips between trusted/untrusted modes to steal some data

**Timing side-channel attack**: learn secrets from frequency of key presses or display updates

**Defense**: Constant rate of encrypted messages to enclave -- send null if no key pressed
Potential Attacks on Trusted Path

**Enclave omission attack**: malicious OS only pretends to use enclave

**Rapid mode switching attack**: malicious OS rapidly flips between trusted/untrusted modes to steal some data

**Timing side-channel attack**: learn secrets from frequency of key presses or display updates

**Replay attack**: keystrokes/overlays replayed
Potential Attacks on Trusted Path

**Enclave omission attack:** malicious OS only pretends to use enclave

**Rapid mode switching attack:** malicious OS rapidly flips between trusted/untrusted modes to steal some data

**Timing side-channel attack:** learn secrets from frequency of key presses or display updates

**Replay attack:** keystrokes/overlays replayed

**Defense:** Include counter in each ciphertext, different counter per device, per origin
Web Enclave Features

HTML Forms

JavaScript

XmlHttpRequests

Local Storage
Web Enclave Features

**JavaScript**

When a secure script is needed, browser calls the enclave.

Secure scripts have access to form inputs, local storage (for the same origin), and can send data encrypted with a key known only to the enclave and server.

**Note:** Content of secure JS is NOT secret, the only secrets are its inputs.
Web Enclave Features

Local Storage
Data stored to disk via authenticated encryption with an enclave *sealing* key
Prevent rollback attacks either with generic solutions or a *revision number* and server assistance
Potential Attacks on Web Enclave

**Enclave misuse attack:** Send unexpected or incorrect calls to enclave
Potential Attacks on Web Enclave

Enclave misuse attack: Send unexpected or incorrect calls to enclave

Defense: Can lead to DoS or crash, but not privacy compromise
Potential Attacks on Web Enclave

**Enclave misuse attack**: Send unexpected or incorrect calls to enclave

**Page tampering attack**: Modify secure form or script tag contents
Potential Attacks on Web Enclave

**Enclave misuse attack**: Send unexpected or incorrect calls to enclave

**Page tampering attack**: Modify secure form or script tag contents

**Defense**: Signatures on secure tags
Potential Attacks on Web Enclave

**Enclave misuse attack:** Send unexpected or incorrect calls to enclave

**Page tampering attack:** Modify secure form or script tag contents

**Redirection attack:** Send data to destination other than intended server
Potential Attacks on Web Enclave

**Enclave misuse attack:** Send unexpected or incorrect calls to enclave

**Page tampering attack:** Modify secure form or script tag contents

**Redirection attack:** Send data to destination other than intended server

**Defense:** Key shared between enclave and intended data recipient
Potential Attacks on Web Enclave

Enclave misuse attack: Send unexpected or incorrect calls to enclave

Page tampering attack: Modify secure form or script tag contents

Redirection attack: Send data to destination other than intended server

Storage tampering attack: Read/change contents of local storage
Potential Attacks on Web Enclave

**Enclave misuse attack:** Send unexpected or incorrect calls to enclave

**Page tampering attack:** Modify secure form or script tag contents

**Redirection attack:** Send data to destination other than intended server

**Storage tampering attack:** Read/change contents of local storage

**Defense:** Local storage encrypted with key only available inside enclave. Enclave policy only gives access to data for same origin
Our Prototype
Our Prototype: Keyboard Dongle
Our Prototype: Display Dongle
Evaluation

TCB Size

TCB size: ~8,500 lines of C++ in enclave (other code is untrusted)

Comparison: Google Chrome has ~18,800,000 lines

Keyboard Latency

Bumpy (prior work) key press to processing latency: 141ms

Fidelius key press to processing latency: 10.6ms (~13x faster)

McCune, Perrig, and Reiter, Safe Passage for Passwords and Other Sensitive Data. NDSS 2009.
Evaluation

Display Latency

Refresh rate 2.8x faster than latest Kindle

Speed due to only sending small overlay rather than encrypting full display

Graph shows latency for Fidelius rendering a username/password login form
Evaluation

Display Bottlenecks

Expensive Render/Refresh due to implementation hacks, easily improvable
Evaluation

Display Bottlenecks

Expensive Render/Refresh due to implementation hacks, easily improvable

Display Scaling

Doubling trusted display size only slightly increases display latency
Evaluation

Page Load Overhead

Main overhead: message from browser to initiate enclave and describe trusted components

Times shown excluding remote attestation
Summary

Fidelius uses enclave to protect user secrets even if entire OS compromised

Support for forms, JS, persistent local storage, and XmlHttpRequests

Trusted path for user I/O

Thank You.

https://crypto.stanford.edu/fidelius