BASTION-SGX: Bluetooth and Architectural Support for Trusted I/O on SGX

Travis Peters¹, Reshma Lal², Srikant Varadarajan², Pradeep Pappachan², David Kotz¹
Dartmouth¹, Intel²

Hardware and Architectural Support for Security and Privacy (HASP)
@ ISCA 2018

June 2nd, 2018
Los Angeles, CA, USA
Outline

• Motivation
  App security & the insecurity of I/O — we need app security + I/O security!

• BASTION-SGX
  A novel Bluetooth Trusted I/O architecture

• Challenges
  Fine-grained channel selection & security policy enforcement

• Proof-of-Concept
  Delivering secure input from Bluetooth keyboards to SGX apps

• Conclusion
  Take-aways and future work
App Security is Imperative...

Financial Apps

Bank of America

Secure Sign-in
- Online ID
- Passcode
- Sign In
- Save Online ID
- Security & Help
- Forgot ID
- Forgot Passcode
- Enroll

Messaging Apps

Health & Wellness Apps

MedicalApp

Travis Peters (traviswp@cs.dartmouth.edu)

BASTION-SGX, HASP’18
Properties:
• Has its own code and data
• Provides confidentiality & integrity
• Full access to app memory

Highlights:
• Small attack surface (app + processor)
• Prevents even privileged SW from stealing or tampering w/ app secrets
I/O Security is **Also** Imperative!

Client Devices (client)  
Bluetooth Devices (device)
Example: Password Theft

**Objective:** Secure input from BT keyboard to TApp.

![Diagram of Bluetooth security setup](image)

- **TApp**: Target Application
- **App1, ..., AppN**: Other applications
- **BT Profiles**: BlueZ (BT Host SW)
- **BlueZ**: BlueZ (BT Host SW)
- **Linux**: Linux
- **BT Device 1, ..., BT Device M**: Bluetooth devices
- **Bluetooth Controller**: Device controller
- **Client Device**: User device
- **Unprivileged Software**: Software not with full privileges
- **Untrusted**: Software not trusted
- **Trusted**: Software trusted
- **Plaintext**: Data in plain text
- **Secure**: Data encrypted

Travis Peters (traviswp@cs.dartmouth.edu)

Log in

Don’t have an account? Sign up for free!

Email address

123

Password

Example: Password Theft

Objective: Secure input from BT keyboard to TApp.
Example: Password Theft

**Objective:** Secure input from BT keyboard to TApp.

1. User enters a password field and types her password…

Unprivileged Software

Privileged Software

Hardware (CPU+ Intel BT HW/FW)

Client Device

BT Profiles

BlueZ (BT Host SW)

Linux

Bluetooth Controller

BT Device 1

…

BT Device M

TApp* → App1 → … → AppN

Log in
Don't have an account? Sign up for free!
Enter your email and password.
Email address
Password

Example: Password Theft

1. User enters a password field and types her password…

Travis Peters (traviswp@cs.dartmouth.edu)

BASTION-SGX, HASP’18
Example: Password Theft

**Objective:** Secure input from BT keyboard to TApp.

1. User enters a password field and types her password...

2. The password is encapsulated within various BT protocol layers for transport and routing.

BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client’s BT controller.

---

Travis Peters (traviswp@cs.dartmouth.edu)
Example: Password Theft

Objective: Secure input from BT keyboard to TApp.

1. User enters a password field and types her password...

2. The password is encapsulated within various BT protocol layers for transport and routing.

BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client’s BT controller.

Travis Peters (traviswp@cs.dartmouth.edu)
Example: Password Theft

**Objective:** Secure input from BT keyboard to TApp.

1. User enters a password field and types her password...

2. The password is encapsulated within various BT protocol layers for transport and routing.

   BT security protects the password during OTA transport.

   The OTA packet is decrypted as soon as it arrives in the client’s BT controller.

Travis Peters (traviswp@cs.dartmouth.edu)
Example: Password Theft

**Objective:** Secure input from BT keyboard to TApp.

1. User enters a password field and types her password...

2. The password is encapsulated within various BT protocol layers for transport and routing.

   BT security protects the password during OTA transport.

   The OTA packet is decrypted as soon as it arrives in the client’s BT controller.

Example: Password Theft

1. User enters a password field and types her password...

2. The password is encapsulated within various BT protocol layers for transport and routing.

   BT security protects the password during OTA transport.

   The OTA packet is decrypted as soon as it arrives in the client’s BT controller.

Travis Peters (traviswp@cs.dartmouth.edu)
Example: Password Theft

**Objective:** Secure input from BT keyboard to TApp.

1. User enters a password field and types her password...

2. The password is encapsulated within various BT protocol layers for transport and routing.

BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client’s BT controller.

Travis Peters (traviswp@cs.dartmouth.edu)
**Example: Password Theft**

**Objective:** Secure input from BT keyboard to TApp.

1. User enters a password field and types her password...

2. The password is encapsulated within various BT protocol layers for transport and routing.

BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client’s BT controller.

---

Travis Peters (traviswp@cs.dartmouth.edu)

BASTION-SGX, HASP’18
Example: Password Theft

**Objective:** Secure input from BT keyboard to TApp.

1. User enters a password field and types her password...

2. The password is encapsulated within various BT protocol layers for transport and routing.

   BT security protects the password during OTA transport.

   The OTA packet is decrypted as soon as it arrives in the client’s BT controller.

3. HCI transport and L2CAP routing...

Travis Peters (traviswp@cs.dartmouth.edu)
Example: Password Theft

**Objective:** Secure input from BT keyboard to TApp.

1. User enters a password field and types her password...

2. The password is encapsulated within various BT protocol layers for transport and routing.

BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client’s BT controller.

3. HCI transport and L2CAP routing...

---

Travis Peters (traviswp@cs.dartmouth.edu)

BASTION-SGX, HASP’18
Example: Password Theft

**Objective:** Secure input from BT keyboard to TApp.

1. User enters a password field and types her password...

2. The password is encapsulated within various BT protocol layers for transport and routing.

   BT security protects the password during OTA transport.

   The OTA packet is decrypted as soon as it arrives in the client’s BT controller.

3. HCI transport and L2CAP routing...

   **Password is stolen!**

   **Without Trusted I/O, data is vulnerable!**

Example: Password Theft

1. User enters a password field and types her password...

2. The password is encapsulated within various BT protocol layers for transport and routing.

   BT security protects the password during OTA transport.

   The OTA packet is decrypted as soon as it arrives in the client’s BT controller.

3. HCI transport and L2CAP routing...

   **Password is stolen!**

   **Without Trusted I/O, data is vulnerable!**

Travis Peters (traviswp@cs.dartmouth.edu)
Our Goal:
• E2E security for select I/O data
• No new HW
• No changes to BT stack/devices
• No dependency on system SW
→ Minimal TCB!

This paper/talk:
• Focus on feasibility
• Secure input data from keyboard

Key Insight: Break path into two subpaths (E1-E2, E3-E4).
Re-encrypt data between E1-E2 (enclave-controller).
Use existing OTA security between E3-E4 (client-device).

Travis Peters (traviswp@cs.dartmouth.edu)
Proposed Architecture: BASTION-SGX

**Bluetooth Trusted I/O Monitor & Filter**
- Monitor all ingress/egress packets
- Update Metadata Table according to BT channel/connection-related events
- Send packets matching security policy to BT-TIO Security Module

**Bluetooth Trusted I/O Metadata Table**
- Store connection/channel metadata

**Bluetooth Trusted I/O API**
- Enable apps to program security policies (i.e., tuple of (CHANNEL-ID & KEY))
- Use extensible interface for 3rd party features (Vendor Specific Debug Commands)

**Bluetooth Trusted Security Module**
- Cryptographic operations (e.g., encryption, decryption)

Travis Peters (traviswp@cs.dartmouth.edu)
Bluetooth Architecture Overview

Apps that send/receive data via Bluetooth

Enable apps to use Bluetooth

Interface between high-level and low-level components

Enable a client device to communicate with other Bluetooth devices

Apps

App1  App2  ...  AppN

Profiles

L2CAP

Host Controller Interface

Bluetooth Baseband & Radio

Controller

Host Software

Travis Peters (traviswp@cs.dartmouth.edu)
1. All packets are multiplexed within the Client’s Bluetooth Controller & sent to Host SW in a single stream.

2. Host SW is responsible for using HCI and L2CAP packet headers for HCI transport and routing.

3. Security should only be applied to data packets, not control packets.

4. Security applied to one channel should not affect other BT channels.
Anatomy of BT Connection

**Q:** How can a Bluetooth Controller identify specific channels over which to enforce Trusted I/O security?

**CONN_HDL/BD_ADDR** identifies specific device connection

**L2CAP CIDs** identify individual channels within a connection

**CoD** defines device type

**PSM** defines protocol/service type

---

Travis Peters (traviswp@cs.dartmouth.edu)
Case Study: Securing HID Input

Setup:
- Implement BASTION-SGX architecture (Section 4)
- Implement trusted app (TApp) for password input
- Install privileged keylogger malware — logs all HID data

Goals:
- Validate Bluetooth Controller’s capabilities (re: fine-grained channel selection)
- Validate that even privileged malware cannot decipher input while security policy is programmed into the Bluetooth Controller

We show that end-to-end (device-to-app) security is possible where....

E1-E2 is secured w/ **new in-host security**

E3-E4 is secured w/ **existing over-the-air security**
Secure Input Flow

Unprivileged Software

Privileged Software
Hardware
(CPU+ Intel BT HW/FW)

Untrusted
Trusted
Plaintext
Secure

New

BT-TIO API*
Metadata Table*
BT-TIO Security*

Client Device

BT Device 1
...
BT Device M

TApp*
App1...
AppN

BT Profiles
BlueZ (BT Host SW)

Linux

BT-TIO API*
Metadata Table*
BT-TIO Security*

Travis Peters (traviswp@cs.dartmouth.edu)
Secure Input Flow

1. User enters password field context - TA generates a symmetric key (κ) and programs security policy into Controller.
Secure Input Flow

1. User enters password field context - TA generates a symmetric key (          ) and programs security policy into Controller.
Secure Input Flow

1. User enters password field context - TA generates a symmetric key (key) and programs security policy into Controller.

2. User types password

Log in

Don't have an account? Sign up for free!

Email address
123

Password

Remember me

Log in

Travis Peters (traviswp@cs.dartmouth.edu)
Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( khóa ) and programs security policy into Controller.

2. User types password

Log in

Don't have an account? Sign up for free!

Email address

123

Password

···

Forgot it?

Remember me

Log in

Travis Peters (traviswp@cs.dartmouth.edu)
Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

Log in

Don't have an account? Sign up for free!
Email address
123
Password
...|

Remember me
Log in

Travis Peters (traviswp@cs.dartmouth.edu)

Travis Peters (traviswp@cs.dartmouth.edu)
Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ), and programs security policy into Controller.

2. User types password

Log in
Don't have an account? Sign up for free!
Email address
123
Password
Forgot it?

Log in

Travis Peters (traviswp@cs.dartmouth.edu)
Secure Input Flow

1. User enters password field context - TA generates a symmetric key (\( \text{\textasciitilde} \)) and programs security policy into Controller.

2. User types password

Log in

Don't have an account? Sign up for free!
Email address
123
Password
\( ... \)

Remember me
Log in

Travis Peters (traviswp@cs.dartmouth.edu)
**Secure Input Flow**

1. User enters password field context - TA generates a symmetric key (key) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching any programmed security policy (key).

Matching packets are sent to BT-TIO security module before transporting to host SW (use key to secure payload).

---

Travis Peters (traviswp@cs.dartmouth.edu)

BASTION-SGX, HASP'18
Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password


Matching packets are sent to BT-TIO security module before transporting to host SW (use to secure payload).
Secure Input Flow

1. User enters password field context - TA generates a symmetric key (\(\mathcal{K}\)) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching *any* programmed security policy (\(\mathcal{F}\), \(\mathcal{K}\)).

Matching packets are sent to BT-TIO security module before transporting to host SW (use \(\mathcal{K}\) to secure payload).

Travis Peters (traviswp@cs.dartmouth.edu)
Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password


Matching packets are sent to BT-TIO security module before transporting to host SW (use to secure payload).

Travis Peters (traviswp@cs.dartmouth.edu)
Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching any programmed security policy ( ). Matching packets are sent to BT-TIO security module before transporting to host SW (use to secure payload).

4. HCI transport and L2CAP routing

Travis Peters (traviswp@cs.dartmouth.edu)
Secure Input Flow

1. User enters password field context - TA generates a symmetric key (🔒) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching any programmed security policy (🔒, 🛡️). Matching packets are sent to BT-TIO security module before transporting to host SW (use 🛡️ to secure payload).

4. HCI transport and L2CAP routing

Travis Peters (traviswp@cs.dartmouth.edu)
Secure Input Flow

1. User enters password field context - TA generates a symmetric key (◎) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching any programmed security policy (◎, ◎). Matching packets are sent to BT-TIO security module before transporting to host SW (use ◎ to secure payload).

4. HCI transport and L2CAP routing

Travis Peters (traviswp@cs.dartmouth.edu)
Secure Input Flow

1. User enters password field context - TA generates a symmetric key and programs security policy into Controller.

2. User types password

3. Controller filters packets matching any programmed security policy. Matching packets are sent to BT-TIO security module before transporting to host SW (use to secure payload).

4. HCI transport and L2CAP routing
Secure Input Flow

1. User enters password field context - TA generates a symmetric key (\(\text{\textbullet}\)) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching any programmed security policy (\(\text{\textbullet}\), \(\text{\textbullet}\)).

Matching packets are sent to BT-TIO security module before transporting to host SW (use \(\text{\textbullet}\) to secure payload).

4. HCI transport and L2CAP routing

Travis Peters (traviswp@cs.dartmouth.edu)

BASTION-SGX, HASP’18
Secure Input Flow

1. User enters password field context - TA generates a symmetric key (рут) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching any programmed security policy (рут,рут). Matching packets are sent to BT-TIO security module before transporting to host SW (useрут to secure payload).

4. HCI transport and L2CAP routing

Travis Peters (traviswp@cs.dartmouth.edu)
Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( setPassword ) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching any programmed security policy ( setPassword , ). Matching packets are sent to BT-TIO security module before transporting to host SW (use setPassword to secure payload).

4. HCI transport and L2CAP routing

Travis Peters (traviswp@cs.dartmouth.edu)
Secure Input Flow

1. User enters password field context - TA generates a symmetric key (.decrypt()) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching any programmed security policy (encrypt(), decrypt()).

Matching packets are sent to BT-TIO security module before transporting to host SW (use encrypt() to secure payload).

4. HCI transport and L2CAP routing

5. TApp decrypts and consumes the password; then clears security policy.
Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password


Matching packets are sent to BT-TIO security module before transporting to host SW (use to secure payload).

4. HCI transport and L2CAP routing

5. TApp decrypts and consumes the password; then clears security policy.

Trustworthy Input!

Travis Peters (traviswp@cs.dartmouth.edu)
Conclusion

Take-aways

• Achieved E2E (app-to-device) security by extending the Bluetooth Controller firmware.
• Our extensions unobtrusively collect per-connection/per-channel metadata for Bluetooth Trusted I/O.
• Use metadata to secure Bluetooth I/O data between SGX app and Bluetooth Controller without...
  • relying on untrusted host software.
  • requiring changes to SGX, Bluetooth device, or Bluetooth standard.
• PoC demonstrates how privileged keylogger cannot access user input data from connected Bluetooth device (keyboard).

Look in the paper* for details on...

• Dynamic key provisioning (Section 4.1.4) to establish secure channel for security policy key programming — re: PCIe & USB-C approach
• Future considerations
  • Extensions to other I/O paths (e.g., Wi-Fi, NFC)
  • Performance evaluation

*BASTION-SGX: Bluetooth and Architectural Support for Trusted I/O on SGX
BASTION-SGX: Bluetooth and Architectural Support for Trusted I/O on SGX

Travis Peters¹, Reshma Lal², Srikanth Varadarajan², Pradeep Pappachan², David Kotz¹
Dartmouth¹, Intel²

Hardware and Architectural Support for Security and Privacy (HASP)
@ ISCA 2018

June 2nd, 2018
Los Angeles, CA, USA