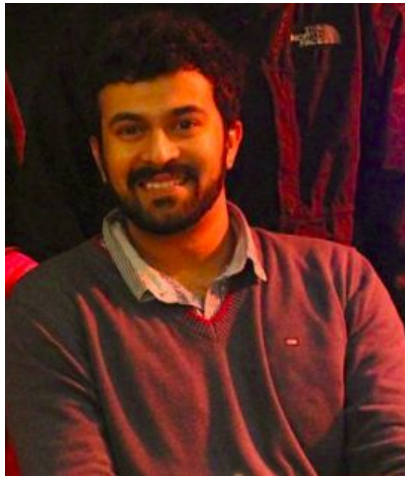
Last mile authentication problem

Exploiting the missing link in end-to-end secure communication

Our team

Thanh Bui
Doctoral Candidate
Aalto University
Finland

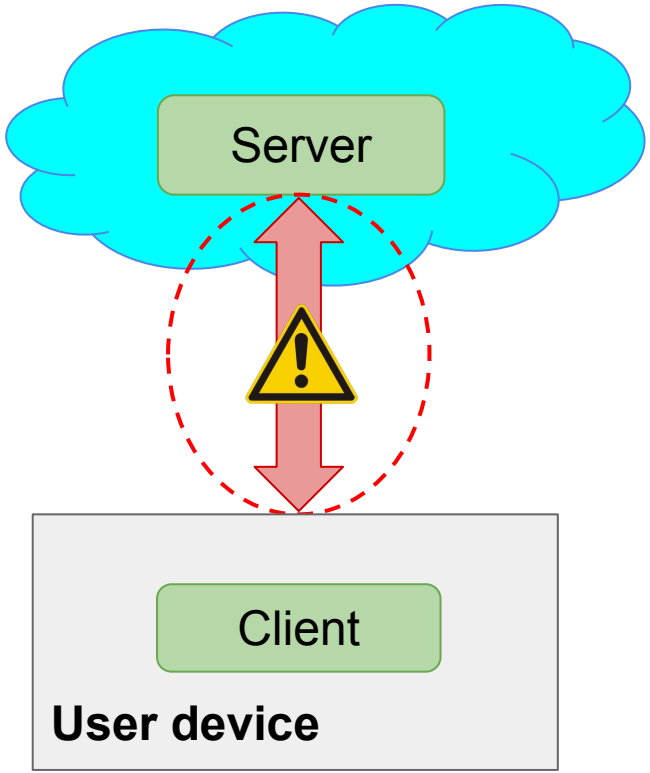


Sid Rao
Doctoral Candidate
Aalto University
Finland

Dr. Markku Antikainen
Post-doc researcher
University of Helsinki
Finland

Prof. Tuomas Aura
Professor
Aalto University
Finland

Traditional network threat model

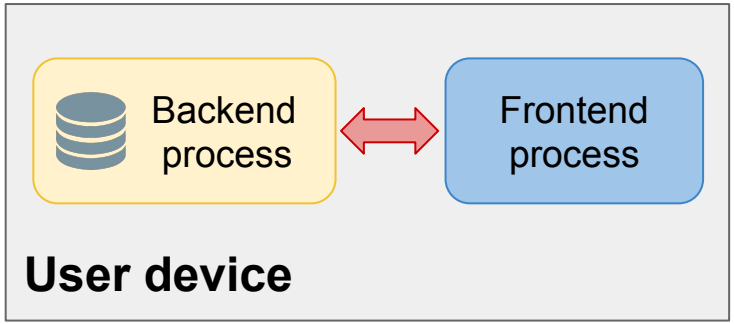


- Server and user device are trusted
- Software is trusted
- Untrusted network:
  - “man in the middle”

- E.g. web server and web browser
- Crypto (TLS and web PKI) to protect communication

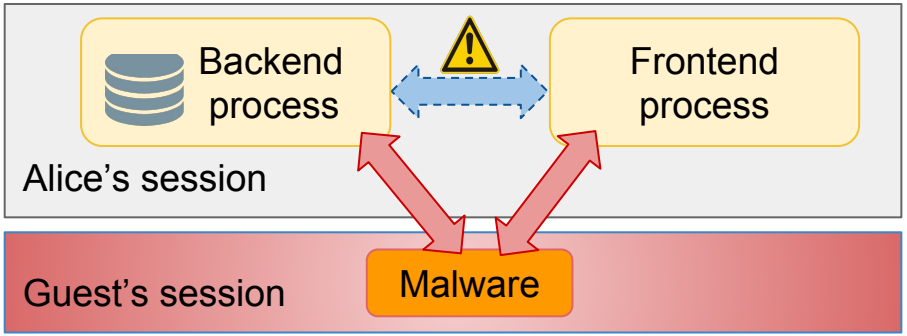
Communication inside a computer - IPC

- Not all communication goes over the traditional network
  - **Inter Process Communication**
- Multiple untrusted services run inside the user device



We try to understand security of communication inside the computer

Man in the Machine (MitMa)



- **MitMa attacker**: Another user at home or at work, including guest
- **Attacker goal**: Impersonate communication endpoints in IPC
- **Attack method**:
  - Runs a malicious process in the background (no privilege escalation required)
  - Fast user switching, remote access (e.g. ssh, remote desktop)

This talk

- We exploit “last mile communication” inside the computer
- Structure
  - IPC and attack vectors
  - Case studies
    - Password managers
    - USB security tokens
  - Mitigation
  - Conclusion

This talk

- We exploit “last mile communication” inside the computer
- Structure
  - IPC and attack vectors
  - Case studies
    - Password managers
    - USB security tokens
  - Mitigation
  - Conclusion

Credentials and **second authentication factors** can be compromised from inside the computer

Inter Process Communication

IPC methods

- File system
- Signal and semaphore
- **Network socket** and Unix domain socket
- Message queue
- Anonymous pipe and **named pipe**
- Shared memory
- Clipboard
- Remote Procedure Call (RPC)
- **USB** (though not strictly an IPC method)
- …

Network sockets

- Mainly for communication across network but used for IPC as well
  - Over loopback interface: `127.0.0.0/8` or `::1/128`
- Server listens on a specified port and waits for incoming client requests
  - Only a single process can bind to a port at a time
  - Any process regardless of its owner can listen on ports > 1024
- Local processes can connect to the server as clients if the port is known
- No built-in access control

Network sockets - Attack vectors (1)

Client impersonation

- Any local process can connect to any server port on `localhost`
- If server accepts only one client, connects before the legitimate client

Server impersonation

- Bind to the port before the legitimate server does

Network sockets - Attack vectors (1)

Client impersonation

- Any local process can connect to any server port on `localhost`
- If server accepts only one client, connects before the legitimate client

Server impersonation

- Bind to the port before the legitimate server does

Legitimate and malicious servers cannot bind to the same port at the same time

Man-in-the-Middle?

Network sockets - Attack vectors (2)

Man-in-the-Middle attack

- **Port agility**
  - If primary port is taken, the server might failover to another port from a predefined list
  - MitMa attacker can
    - Listen on the primary port to receive client connection, and
    - Connect himself to the legitimate server on the secondary port

Network sockets - Attack vectors (2)

Man-in-the-Middle attack

- **Port agility**
  - If primary port is taken, the server might failover to another port from a predefined list
  - MitMa attacker can
    - Listen on the primary port to receive client connection, and
    - Connect himself to the legitimate server on the secondary port
- **No port agility**
  - Replay messages by alternating between the client and server roles
  - Rate of messages passing through the attacker will be slow but might still be practical

Windows named pipe

- Similar client-server architecture as network socket's, but multiple processes can simultaneously act as servers
  - A named pipe can have multiple **instances** that share the same name
  - Each instance connects exactly one **pipe server** and one **pipe client**
  - New pipe clients connected to the pipe servers in round-robin order
- Named pipes placed in a special path `\\.\pipe\`
  - Every user (including guest) has access to the path
- Have built-in access control!

Windows named pipe - Access control

- If a named pipe doesn't exist, any user can create it (including guest)
- If it exists, only users with `FILE_CREATE_PIPE_INSTANCE` permission can create new instances
- The first instance of the named pipe decides the maximum instances and security descriptor (DACL)
  - Default DACL:
    - `READ` access to everyone
    - `FULL` access to the creator and admins

Windows named pipe - Attack vectors

Client impersonation

- Any process can connect to any open pipe instance but subject to access control check

Server impersonation

- **Pipe name hijacking**: create the first instance to control the pipe's DACL and maximum instances
  - Set DACL to allow everyone to create new instances

Client + Server impersonation = **Man-in-the-middle**

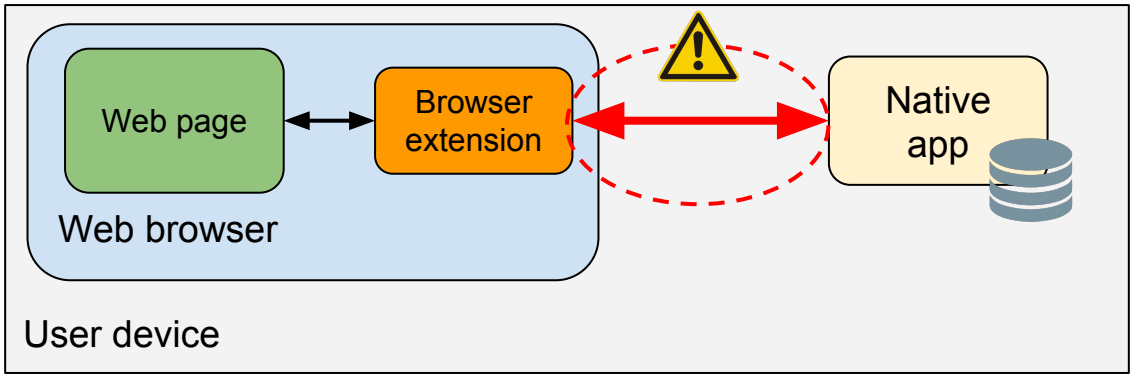
USB Human Interface Devices (USB HID)

- E.g. keyboards, pointing devices, hardware security tokens
- On Linux and macOS,
  - USB HIDs accessed from only current active user session
- On Windows,
  - USB HIDs accessed from any user session, including those in the background
  - Security of the devices depends on application-level security mechanism implemented in the hardware or software

Case studies

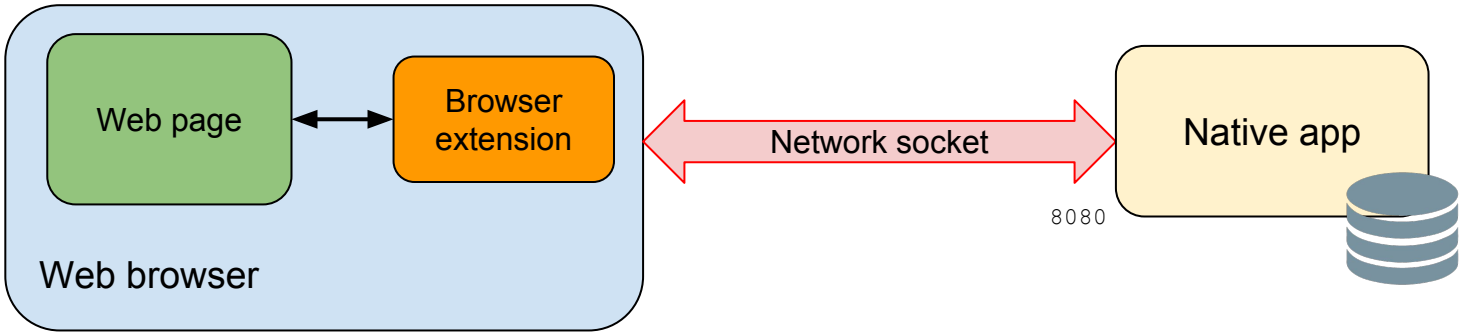
Password managers

- **Native (desktop) app** manages the password vault
  - Password vault encrypted with a key derived from a master password
- **Browser extension** creates and stores passwords, and enters them into login pages
- Browser extension and native app communicate via IPC



Password managers w/ Network socket

- Native app runs a HTTP/WebSocket server on a predefined port
- Browser extension connects as a client to the server
- Threats:
  - Browser extension is sandboxed → unable to perform most checks on the server process



Case study - Dashlane

- Dashlane app runs a WebSocket server on port 11456
- Communication:
  - Messages encrypted with a key derived from hard-coded constant
  - Server verifies client by checking:
    - Browser extension ID in `Origin` header of each message
    - Code signature of the client process to make sure it is a known browser
    - Client process owned by same user as the server's
  - Client does NOT verify server

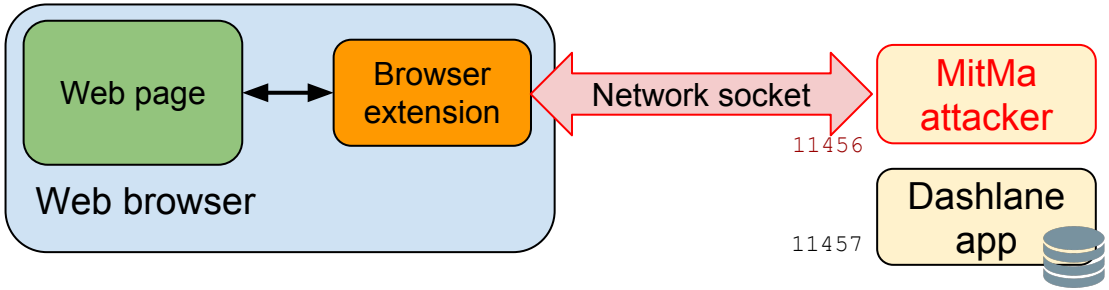
Case study - Dashlane

- Dashlane app runs a WebSocket server on port 11456
- Communication:
  - Messages encrypted with a key derived from hard-coded constant
  - Server verifies client by checking:
    - Browser extension ID in `Origin` header of each message
    - Code signature of the client process to make sure it is a known browser
    - Client process owned by same user as the server's
  - Client does NOT verify server

Server impersonation on Dashlane (1)

Attack steps:

1. Extract hard-coded encryption key from the browser extension's code
2. Run WebSocket server on port 11456 before the benign server does
   - Benign server silently failovers to another port
3. Use the hard-coded key to communicate with the browser extension



Server impersonation on Dashlane (2)

Results:

- Dashlane browser extension collects and sends DOM elements from web pages to app for analysis

  → Attacker obtains personal data, such as emails and messages, from web pages

- Server can instruct the browser extension to collect web-form data and send to it

  → Attacker obtains any text typed by the user, including credentials

Case study - 1Password

- 1Password app runs a WebSocket server on port 6263
- Communication:
  - Server verifies client in the same way as Dashlane does
  - In the first communication, server and client run a self-made protocol to agree on a shared encryption key

1. C → S: "hello"
2. C ← S: code (random 6-digit string)
3. C → S: hmac_key
4. Both extension and app displays the code
5. User confirms to the app if they match
6. C ← S: "authRegistered"
7. C → S: $nonce_C$
8. C ← S: $nonce_S$, m_S=HMAC(hmac_key, $nonce_S$||$nonce_C$)
9. C → S: m_C=HMAC(hmac_key, m_S)
10. C ← S: "welcome"
11. Both sides derive encryption key K=HMAC(hmac_key, m_S||m_C||"encryption")

Case study - 1Password

- 1Password app runs a WebSocket server on port 6263
- Communication:
  - Server verifies client in the same way as Dashlane does
  - In the first communication, server and client run a self-made protocol to agree on a shared encryption key

❌ **Client impersonation**

✓ **Server impersonation**

1. C → S: "hello"
2. C ← S: code (random 6-digit string)
3. C → S: hmac_key
4. Both extension and app displays the code
5. User confirms to the app if they match
6. C ← S: "authRegistered"
7. C → S: $nonce_C$
8. C ← S: $nonce_S$, m_S=HMAC(hmac_key, $nonce_S$||$nonce_C$)
9. C → S: m_C=HMAC(hmac_key, m_S)
10. C ← S: "welcome"
11. Both sides derive encryption key K=HMAC(hmac_key, m_S||m_C||"encryption")

MitMa

on

1Password

(Demo)

Native messaging

- Browser's built-in method for communicating between browser extension and native code
- App registers with the browser:
  - an executable, called **Native Messaging Host (NMH)**, and
  - a configuration file specifying which browser extensions have access to the NMH
- Browser starts the NMH in a child process and lets the browser extension communicate with it

Native messaging

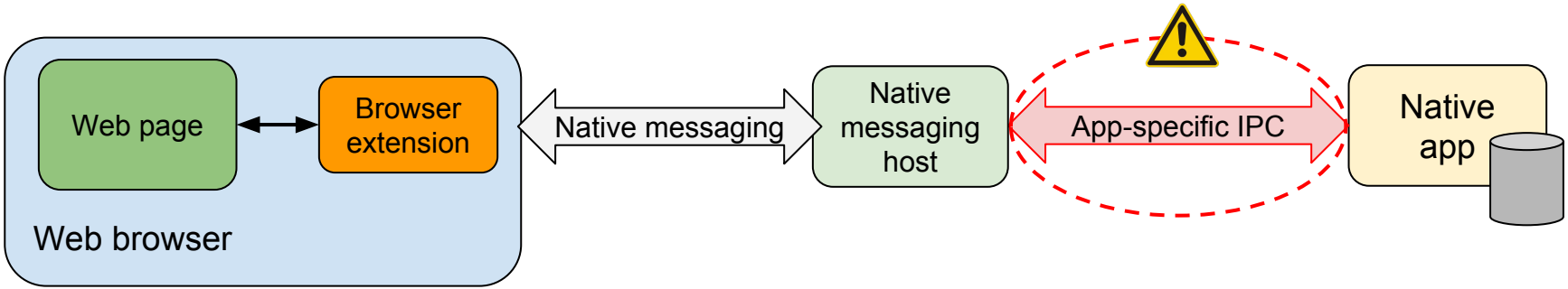
- Browser's built-in method for communicating between browser extension and native code
- App registers with the browser:
  - an executable, called **Native Messaging Host (NMH)**, and
  - a configuration file specifying which browser extensions have access to the NMH
- Browser starts the NMH in a child process and lets the browser extension communicate with it

✓ **Native messaging is immune to MitMa attacks**

Password managers w/ Native messaging

- Password manager app registers a NMH with the browser and allows only its browser extension to communicate with the NMH
- However, NMH and the native app are still two separate processes

  → they use IPC to communicate with each other



Case study - Password Boss

- On Windows, named pipe used for IPC between NMH and the native app
- When the app starts, it creates a named pipe with:
  - Fixed name
  - Maximum instance = 50
  - DACL allowing all authenticated users to have `FULL` access
- NMH connects to the named pipe as a pipe client and forwards messages between browser extension and native app
- All messages are sent in plaintext

Case study - Password Boss

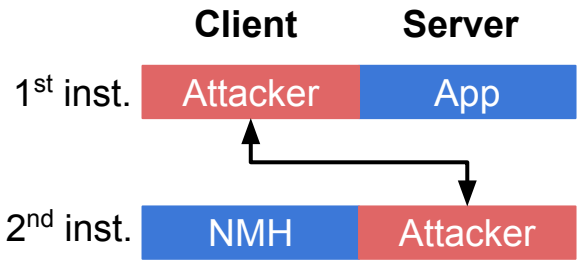
- On Windows, named pipe used for IPC between NMH and the native app
- When the app starts, it creates a named pipe with:
  - Fixed name
  - Maximum instance = 50
  - DACL allowing all authenticated users to have `FULL` access
- NMH connects to the named pipe as a pipe client and forwards messages between browser extension and native app
- All messages are sent in plaintext

✓ **Man-in-the-Middle**

Man-in-the-Middle on Password Boss (1)

By authenticated user:

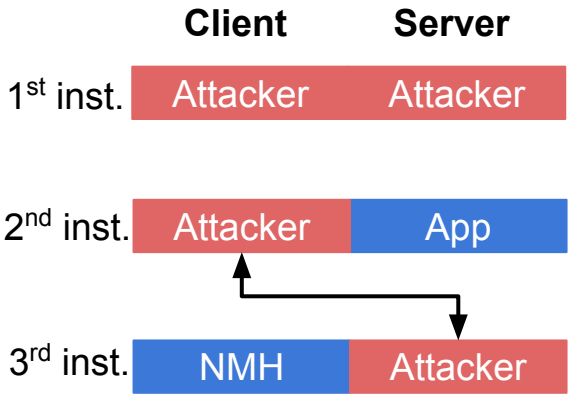
1. Connects as a client to the app's named pipe instance
2. Creates another instance of the named pipe and waits for the NMH
3. NMH connects to the attacker's instance because it is the only open instance
4. Forwards messages between the two pipe instances



Man-in-the-Middle on Password Boss (2)

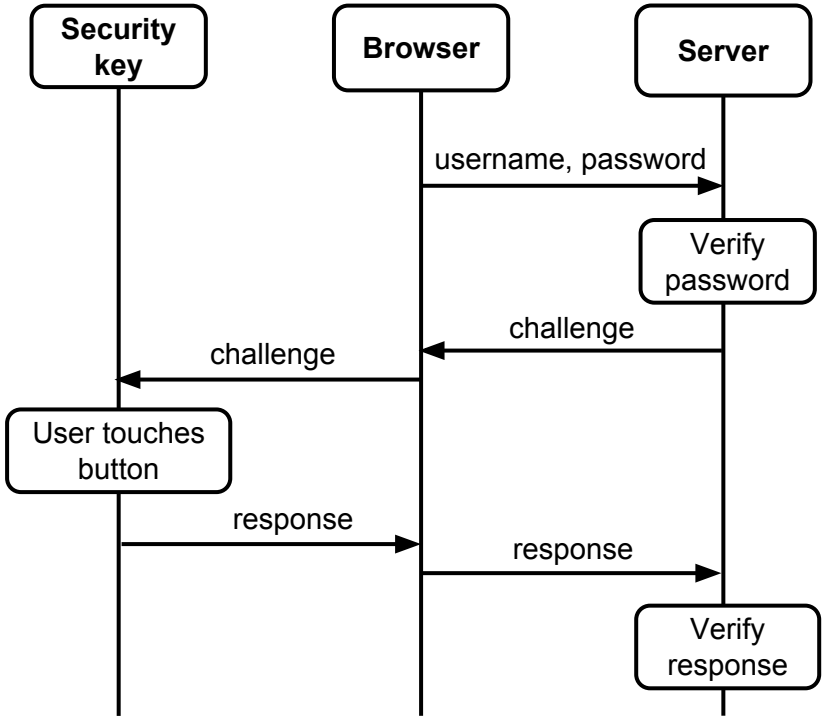
By guest user:

- Guest cannot access the app's pipe instance or create a new one
- Solution: Pipe name hijacking
  - Create the first instance and set a `FULL` access DACL to all users
- Rest is same as the attack by authenticated user



Case study - FIDO U2F security key

- 2nd authentication factor based on public key crypto and a USB device
- **Challenge-response protocol**
  - Browser keeps sending the challenge from the server to the device until it receives response (every 300 ms on Chrome)
  - User activates the device by touching a button on it
  - The device responds to only the first request after the touch



Unauthorized access of FIDO U2F key

Reminder: On Windows, USB HIDs can be accessed from any user session

Assumption: Attacker has obtained the 1st authentication factor

Attack steps:

1. Signs in to the service in the background using the 1st factor
2. Sends the challenge from the service to the device at a high rate
3. Victim (in the foreground) signs in to ANY service using the same security key and touches the button on the device
   - The first button touch had no effect, but such minor glitches are typically ignored
4. The attacker's request gets signed instead of the victim's with high probability, and then the attacker obtains the 2nd factor

MitMa

on

FIDO U2F Key

(Demo)

Discovered vulnerabilities

Application		macOS	Windows	Linux	IPC Channel	Attacks
Password managers	Roboform	**Y**	**N**	**-**	Network socket	Client imp.
	Dashlane	**Y**	**Y**	**-**	Network socket	Server imp.
	1Password	**Y**	**N**	**-**	Network socket	Server imp.
	F-Secure Key	**Y**	**Y**	**-**	Network socket	Client imp. Server imp.
	Password Boss	**N**	**Y**	**-**	Named pipe	Man-in-the-Middle
	Sticky Password	**Y**	**N**	**-**	Network socket	Client imp. Server imp.
Hardware tokens	FIDO U2F Key	**N**	**Y**	**N**	USB	Unauthorized access
	DigiSign	**Y**	**Y**	**Y**	Network socket	Client imp.
Others	Blizzard	**Y**	**Y**	**-**	Network socket	Client imp.
	Transmission	**Y**	**Y**	**Y**	Network socket	Client imp.
	Spotify	**Y**	**Y**	**Y**	Network socket	Client imp.
	MySQL	**N**	**Y**	**N**	Named pipe	Man-in-the-Middle
	Keybase	**N**	**Y**	**N**	Named pipe	Server imp.

Mitigation

- **Spatial and temporal separation of users**
  - Limit the number of users that can access a computer
  - Disable remote access, such as SSH and Remote desktop
- **Attack detection easier in IPC than in network**
  - Compare owner of client and server processes
  - Query client/server binary
- **Cryptographic protection**
  - User-assisted pairing vs TLS and PKI
  - Avoid self-made crypto!

Conclusion

Software developers beware, IPC is not inherently secure!

- Unprivileged user or process can attack IPC of another user on the same computer
- Traditional network security threats exist also in IPC
- We are doing a more exhaustive survey
  - Contribute, if you are interested!

Thank you!

Contact

Thanh Bui: thanh.bui@aalto.fi
Siddharth Rao: siddharth.rao@aalto.fi
Markku Antikainen: markku.antikainen@helsinki.fi
Tuomas Aura: tuomas.aura@aalto.fi

Read more

"Man-in-the-Machine: Exploiting ill-secured communication inside the computer", USENIX Security 2018