Lecture Series on HPI Research SoSe 2024
Philipp Keese
ITSE-Master
Research Project Proposal
Based on *Wireless Security* by Dr.-Ing Jiska Classen

Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Exploiting **Weak Security** in Wireless Peripheral Communication

Unifying Exploit Development

Goal

Reverse-engineering Device Firmware to Find Vulnerabilities

This project builds on prior findings[1,2,3] on the security of protocols used in wireless peripheral devices in use today. Our goal is to further investigate the impact of insecure communication between a peripheral device and the user's machine with regards to firmware security and potential for further exploitation or infection of connected devices. This project will focus on Logitech's *Unifying* protocol in particular due to the wide range of devices it supports.

- Reverse-engineer device firmware
- Find security vulnerabilites arising from insecure wireless communication
- Demonstrate exploits on real hardware
- Suggest potential fixes to protocols or device firmware to mitigate attacks

Example Device (Logitech K400+)[4]

Problem

Insecure Wireless Protocols

Widely-used protocols for communication between wireless peripherals (mice, keyboards) and the user's device have been shown to use insecure encryption or leak encryption keys in plain text[1]. These shortcomings allow attackers to inject keypresses, read input data remotely or deny the victim the ability to use their device.

Furthermore, manufacturer response has been limited[2] and many devices remain vulnerable to malicious transmissions. It is unclear wether these exploits could serve as a foothold to more harmful attacks.

- History of exploitable wireless protocols
- Wide range of affected devices
- Low-cost hardware required for attack
- Older devices left vulnerable
- Larger impact as research topic

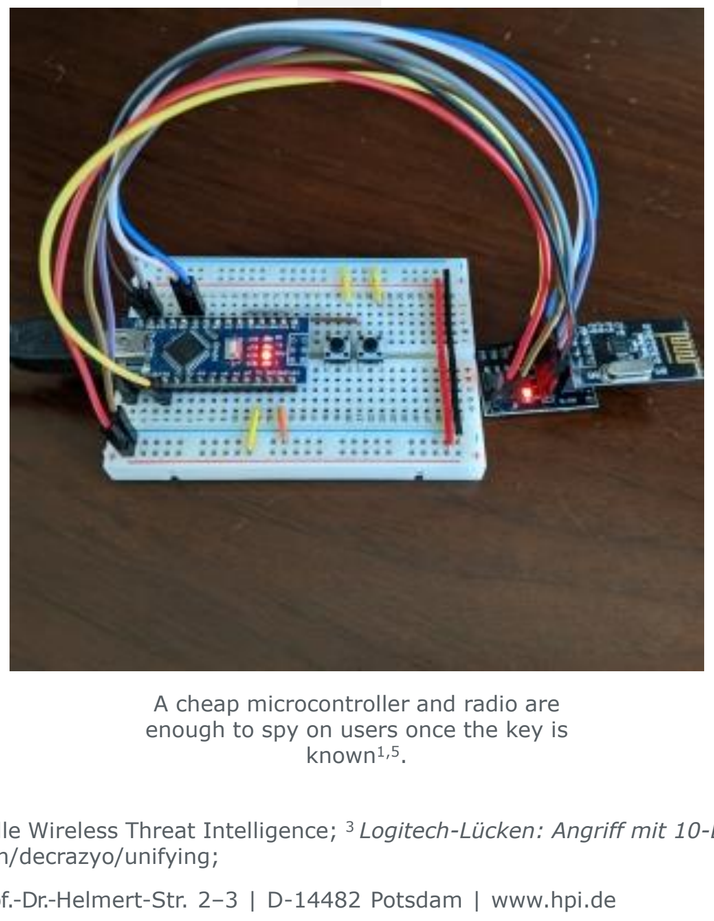
Solution

Finding (and Fixing) Vulnerabilities

Should any vulnerabilities be found in device fimware, our hope is to develop a mitigation for it or provide an outright fix. This of course would need to be compatible with unpatched devices and not degrade the user experience compared to a vulnerable device.

To find vulnerable code paths, techniques like input fuzzing and simulation could prove useful to enhance reverse-engineering of firmware.

- Develop mitigations for found vulnerabilities
- Ensure compatibility with unmodified devices
- Implement patch for firmware
- Test usability and performance



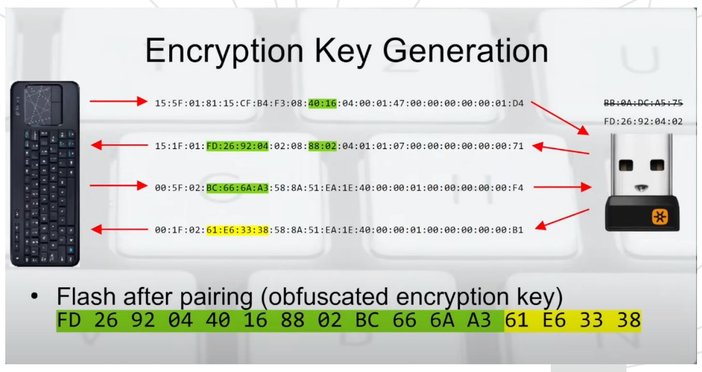
A cheap microcontroller and radio are enough to spy on users once the key is known[1,5].

Connection

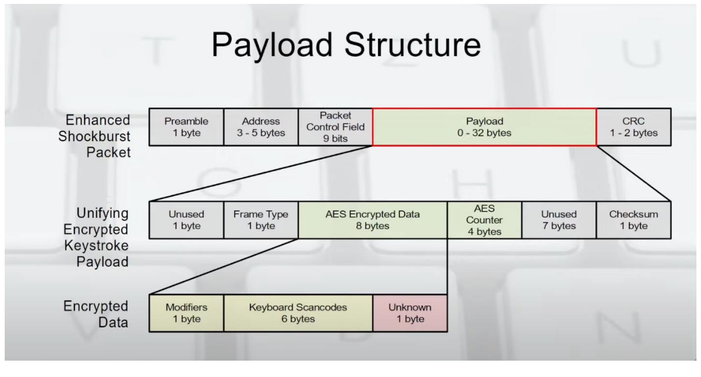
Relevance to Lecture

This project combines wireless security research with in-depth reverse-engineering of software implementing vulnerable wireless stacks. Like Bluetooth, the Unifying is flawed at the protocol level and any patches to it face the same challenges of remaining compatible with older devices while providing improved data security with newer versions. As a wireless protocol, it is subject to threats unique to this medium.
The lecture goes into detail for a range of wireless attacks and how to defend against some of them. It also mentions fuzzing as a means of rapid exploit development and how open tools for interfacing with wireless hardware have boosted the number of reported vulnerabilities in Bluetooth.
This project aims to replicate a portion of this success by building on prior research on the Unifying protocol and fits well as an extension of the presented topics.

- Security of wireless devices
- Vulnerable protocol seeing wide-spread use
- Fuzzing used for bug-finding
- Cryptographic attacks possible, though unlikely



Pairing process of Unifying protocol. Keys are exchanged in plain text[1].



Structure of a packet. Keypresses are encrypted with a pairing key[1].

Quellen: [1] *Hacking Logitech Unifying*, Talk at DC612; [2] *MouseJack*, Bastille Wireless Threat Intelligence; [3] *Logitech-Lücken: Angriff mit 10-Euro-Hardware möglich, jetzt handeln!*, Heise;
[4] Image: reichelt.de product page; [5] Image: https://github.com/decrazyo/unifying;