A Touch of Pwn

Attacking Windows Hello Fingerprint Authentication

Jesse D'Aguanno
@0x30n

Timo Teräs
@TerasTimo

1

Background

Introduction

- Who are we?
  - Timo Teräs
    - Security Researcher – Blackwing Intelligence
    - Alpine Linux Core
    - Likes: Reverse Engineering, Applied Cryptography, Low Level Development, CTFs, Sauna, Cooking
  - Jesse D'Aguanno
    - Director of Research – Blackwing Intelligence
    - Likes: Reverse Engineering, Vulnerability Research, Applied Cryptography, Program Analysis, Long Walks on the Beach

Mission

- Mission: Windows Hello Fingerprint Authentication Vulnerability Research
- Targets: Top 3 Embedded Devices
  - Selected by Windows Hello Team
- Goal: Bypass Windows Hello Authentication

Mission

SECRET MISSION

MYTHBUSTERS

- Approach: Physical "presentation" attacks
  - Cloning fingerprints, etc.
  - 🚫 Not our focus today (but fun!)

Mission

SECRET MISSION

- Approach: Software / Hardware Attacks
  - Black box
    - No source code
    - Just three brand new laptops
    - Reverse Engineering, vulnerability research, exploit dev (potentially)
  - ✅ Today's focus

Threat Model

- Physical Access to Target
  - Stolen / Confiscated Device
  - Evil Maid
  - Etc.
- Bypass Windows Hello Authentication

- Not in Threat Model: Local Privilege Escalation

Targets

★ Targets

Targets

- Dell Inspiron 15
  - Sensor: Goodix

Targets

- Lenovo ThinkPad T14s
  - Sensor: Synaptics

Targets

- Microsoft Surface Pro X
  - Sensor: ELAN

Match on Chip

- Match on Chip (aka "Match on Sensor")
  - Biometric data never leaves the sensor
    - Sensor includes a microprocessor and memory
    - Storage & matching performed within sensor hardware
  - Can't just replay a captured image to the host
  - Windows Hello Enhanced Sign-in Security *only* supports match on chip

Enhanced Sign-in Security

- Windows Hello Enhanced Sign-in Security
  - Goals:
    - Protect biometric data storage
    - Secure communication channel
  - Hardware / Software Components
    - Virtualization Based Security (VBS)
    - Trusted Platform Module 2.0 (TPM)
    - Secure Device Connection Protocol (SDCP)

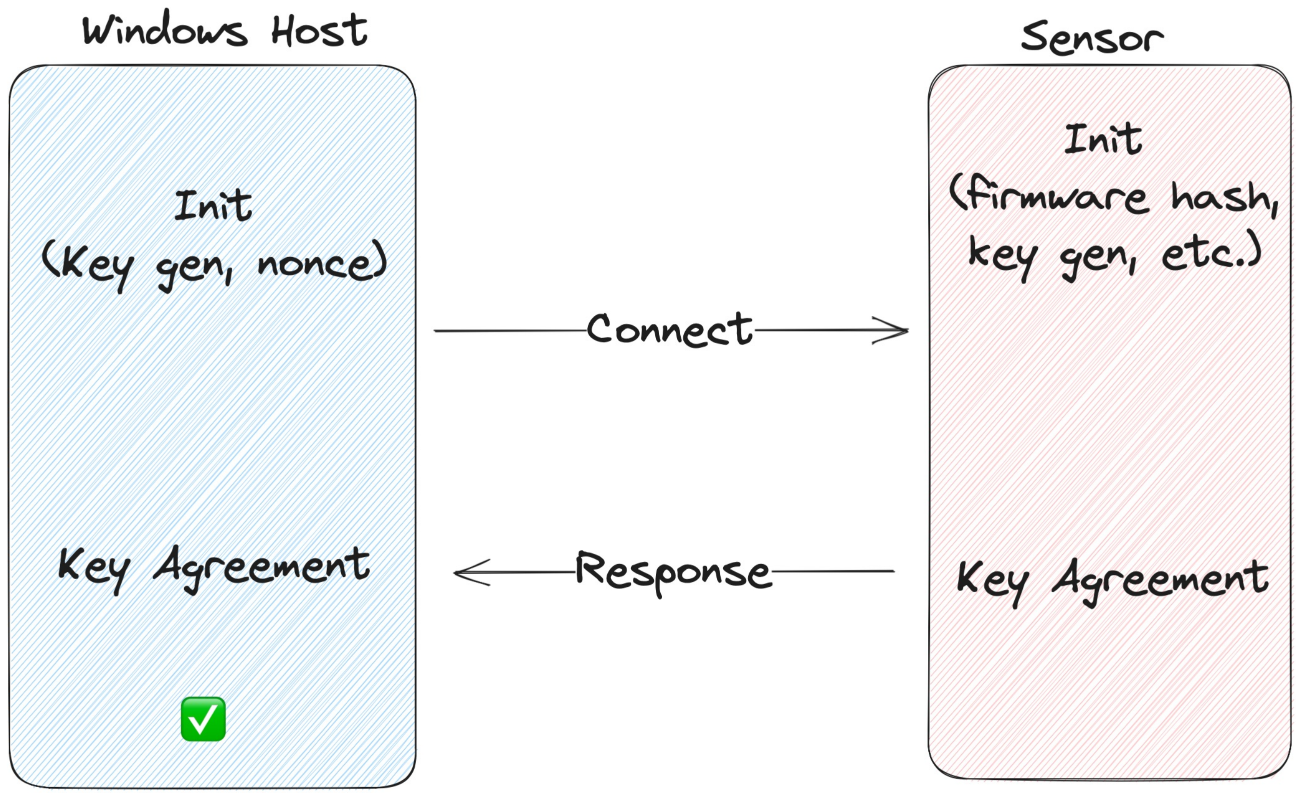
Secure Device Connection Protocol

- Secure Device Connection Protocol (SDCP)

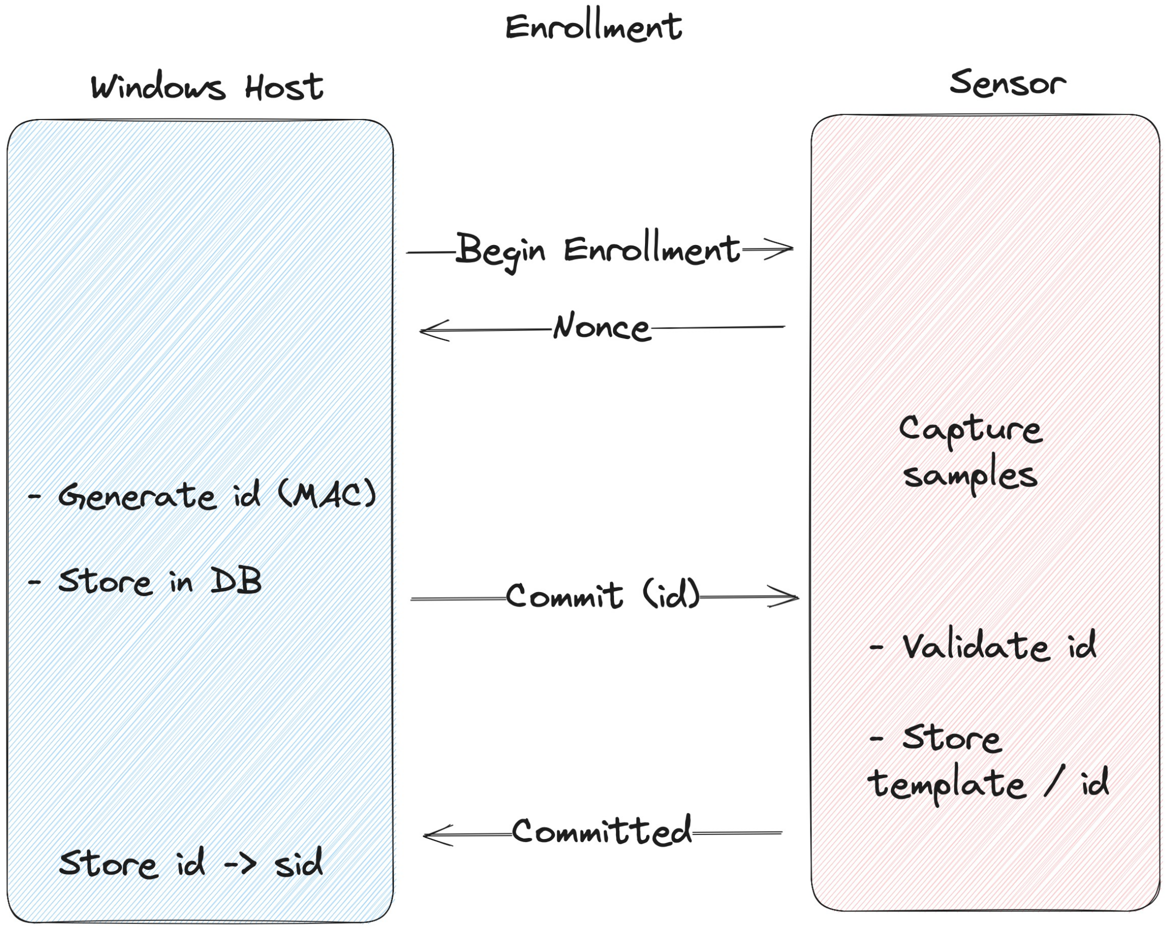
- Goal: End-to-End Secure Channel Between Host and Sensor
  - Attempts to ensure host is communicating with trusted / healthy sensor
    - Firmware validated against ROM root of trust
  - Biometric operations cryptographically secured
    - `Enroll()`
      - Authenticated (via MAC) nonce used as stored user identifier
    - `Identify()`
      - Authenticated to prevent MitM and replay

Secure Device Connection Protocol

SDCP Bootstrap

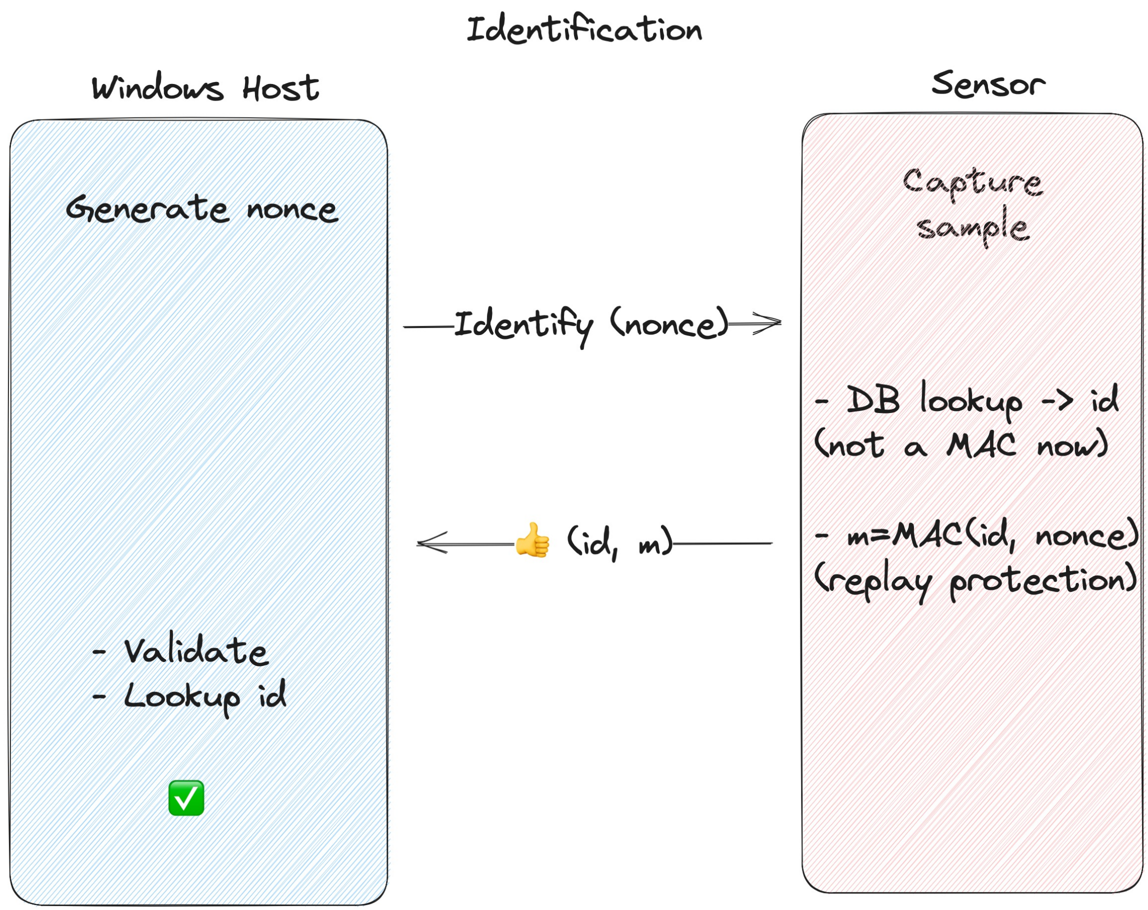


Secure Device Connection Protocol

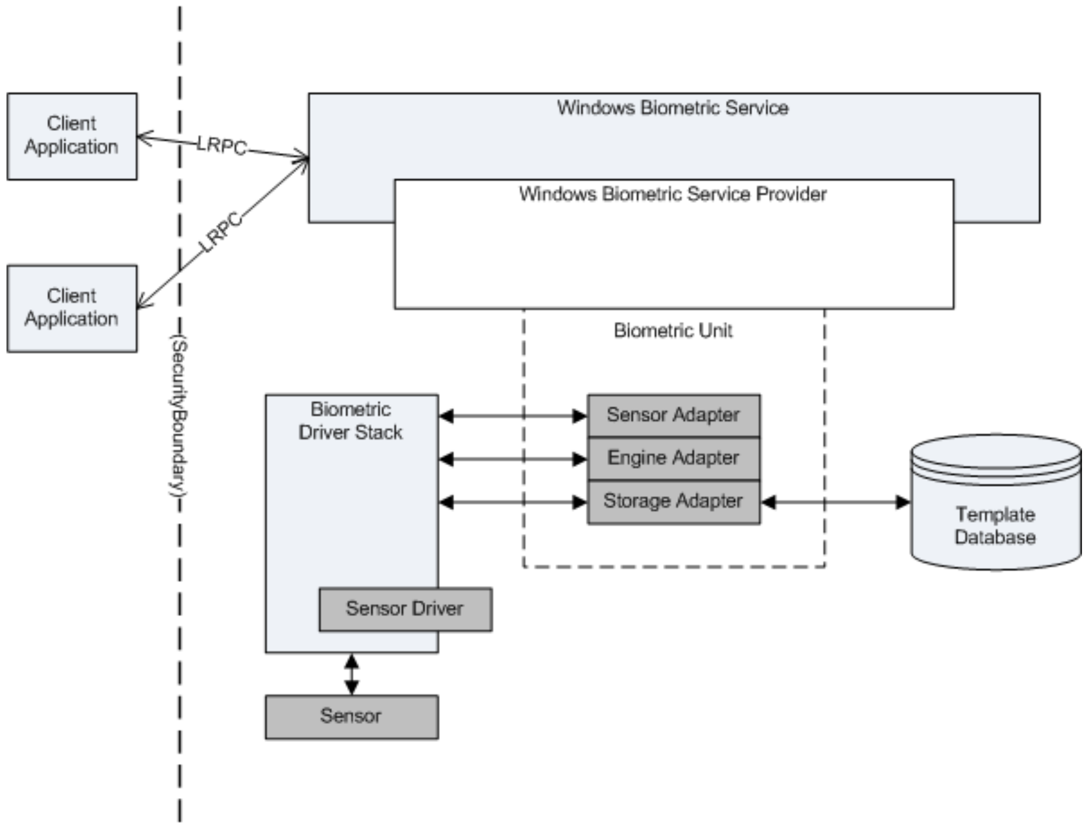


BLACKWING INTELLIGENCE

Secure Device Connection Protocol

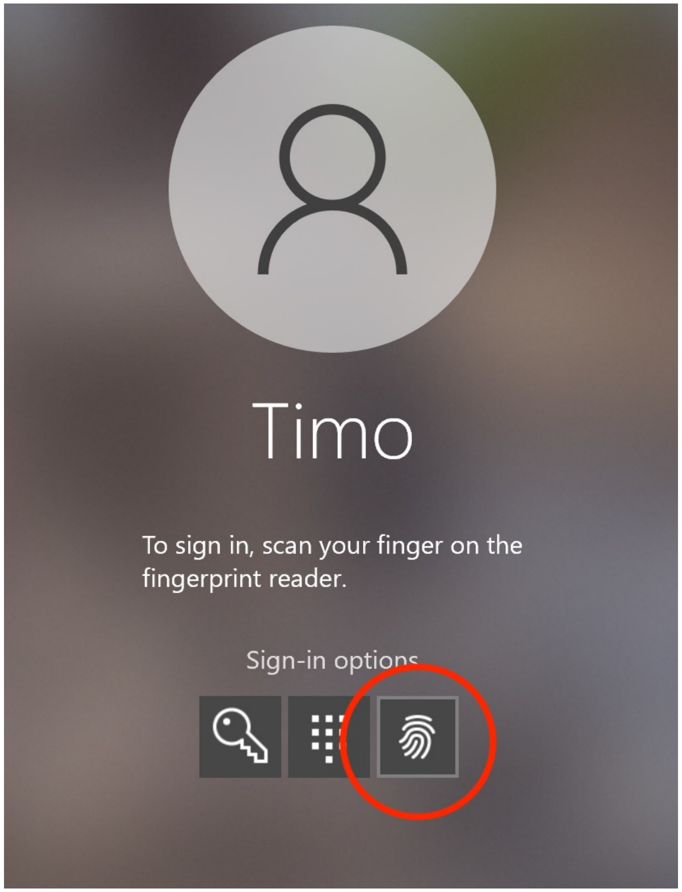


Windows Biometric Framework



https://learn.microsoft.com/en-us/windows/win32/secbiomet/framework-security

Windows Biometric Framework



Approach

- Initial Analysis
  - Initial RE
  - Attack Surface Assessment
  - Bus / Protocol Identification
  - System Configuration Review
  - Code Quality Assessment
- Target Prioritization
- Target Assessment & Exploitation
  - In-depth RE
  - Protocol dissection
  - Vulnerability Research
  - Exploit Development (PoCs)

Target Prioritization

- 1 - Goodix
  - Supports Secure Device Connection Protocol (SDCP)
  - Good support in Linux
  - Cleartext USB communication
  - Overall poor code quality
- 2 - Synaptics
  - Supports SDCP (kinda)
  - Limited Linux support
  - Encrypted USB communication
  - Better code quality
- 3 – ELAN
  - Doesn’t support SDCP
  - Limited tooling (Surface Pro X – Windows on Arm)

2

Targets

Goodix - Overview

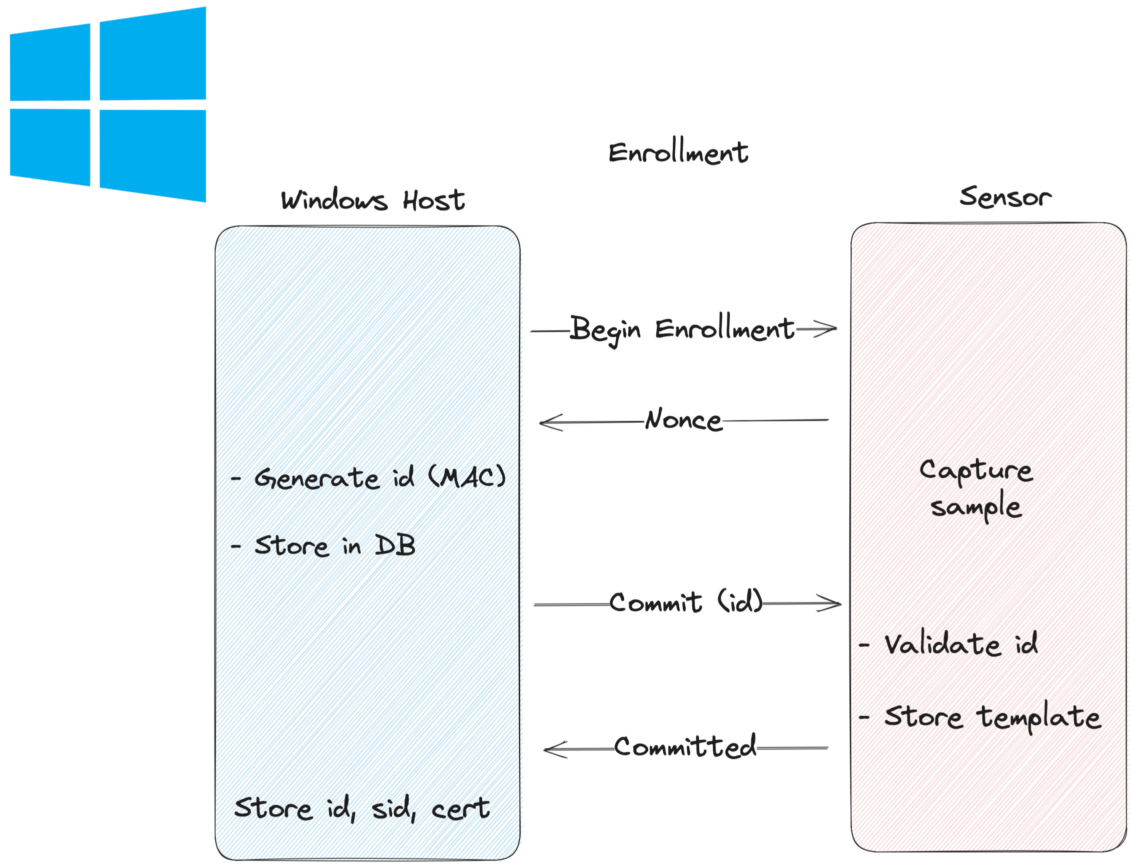
- Bus
  - Internal USB
- Type
  - Match on Chip (MOC)
- OS Support
  - Windows Hello
  - Linux

Goodix - Research

- Research Goals
  - *Examine Windows Configuration*
  - *Examine Linux drivers*
  - *Observe Host-Sensor Communications*
  - *Reverse Engineer Drivers*
  - *Reverse Engineer Firmware*

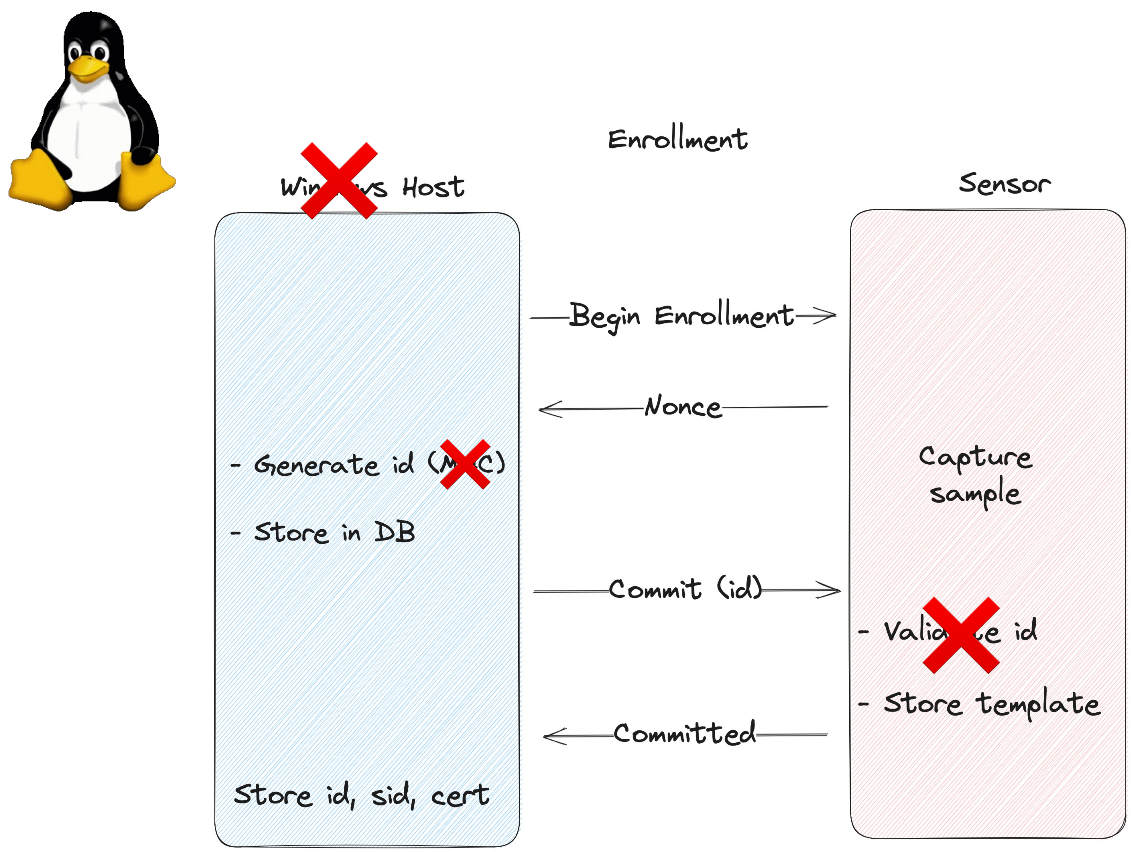
Goodix - Findings

Goodix - Findings



BLACKWINGINTELLIGENCE

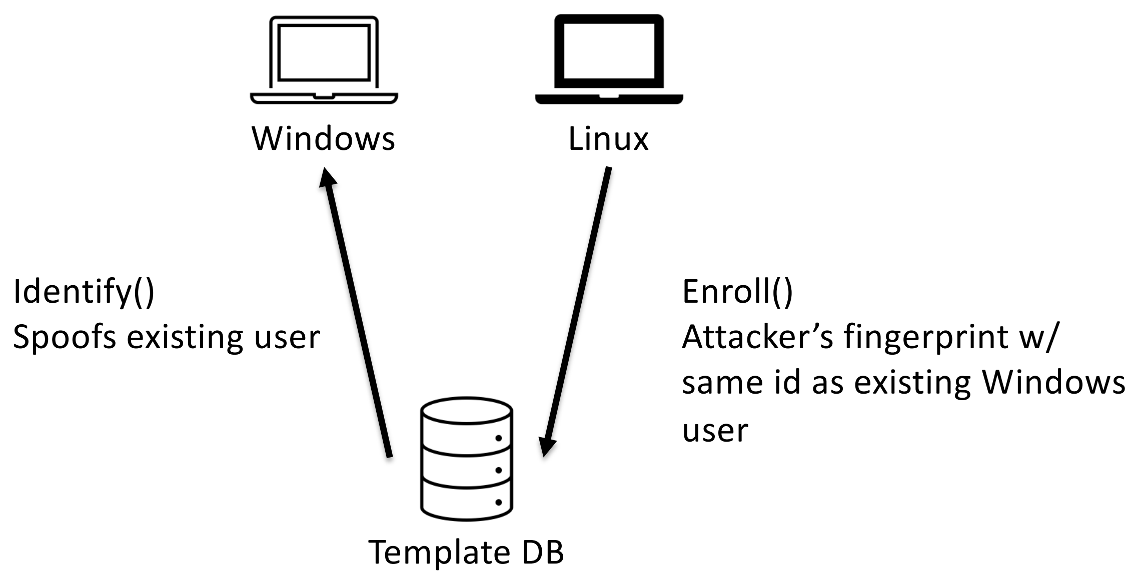
Goodix - Findings



Goodix - Findings

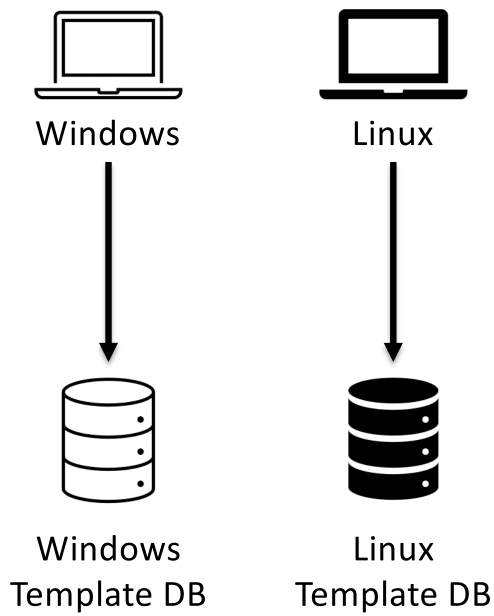
Goodix - Findings

Goodix - Overview



Goodix - Findings

Goodix - Overview

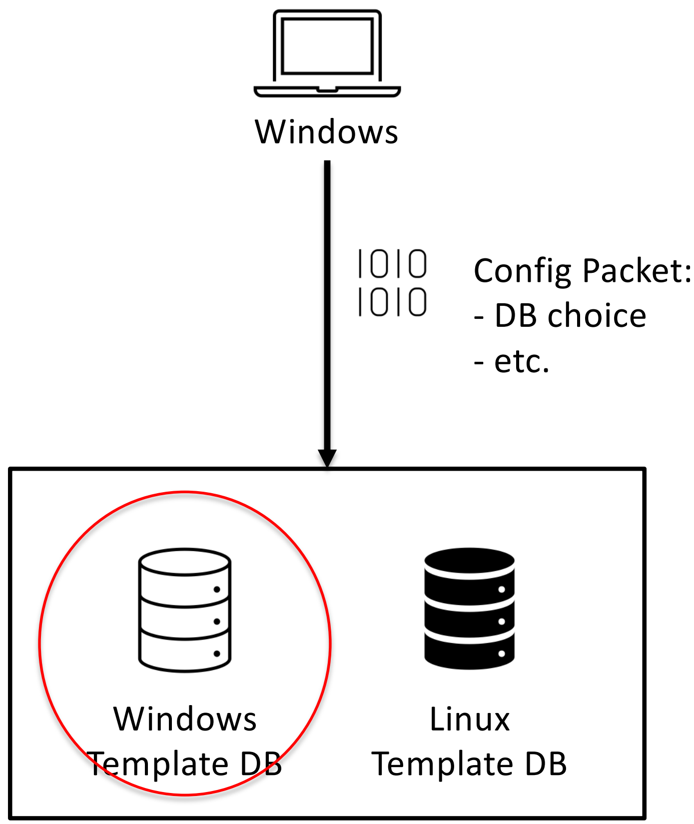


Goodix - Overview

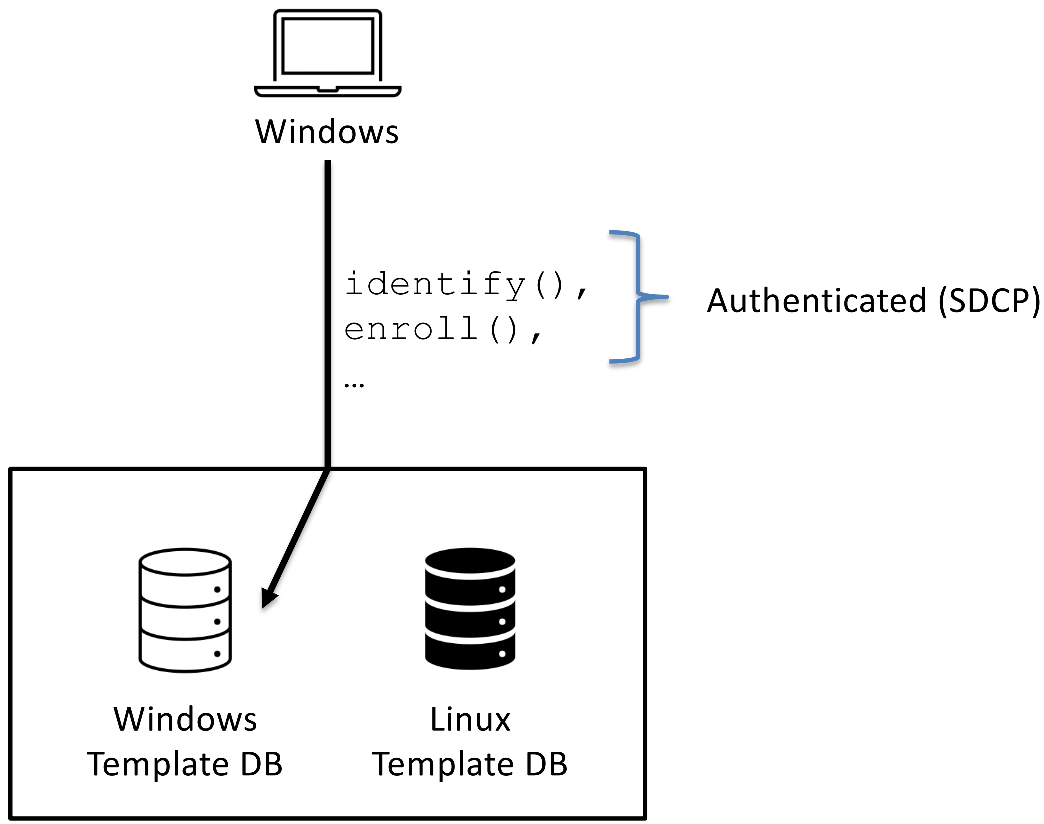
- How does the sensor know what database to use?

- Driver sends configuration packet to sensor on device initialization
  - Sets various interesting config data **- including what DB to use!**
  - Maintains configuration state until next configuration received
  - ***Unauthenticated***

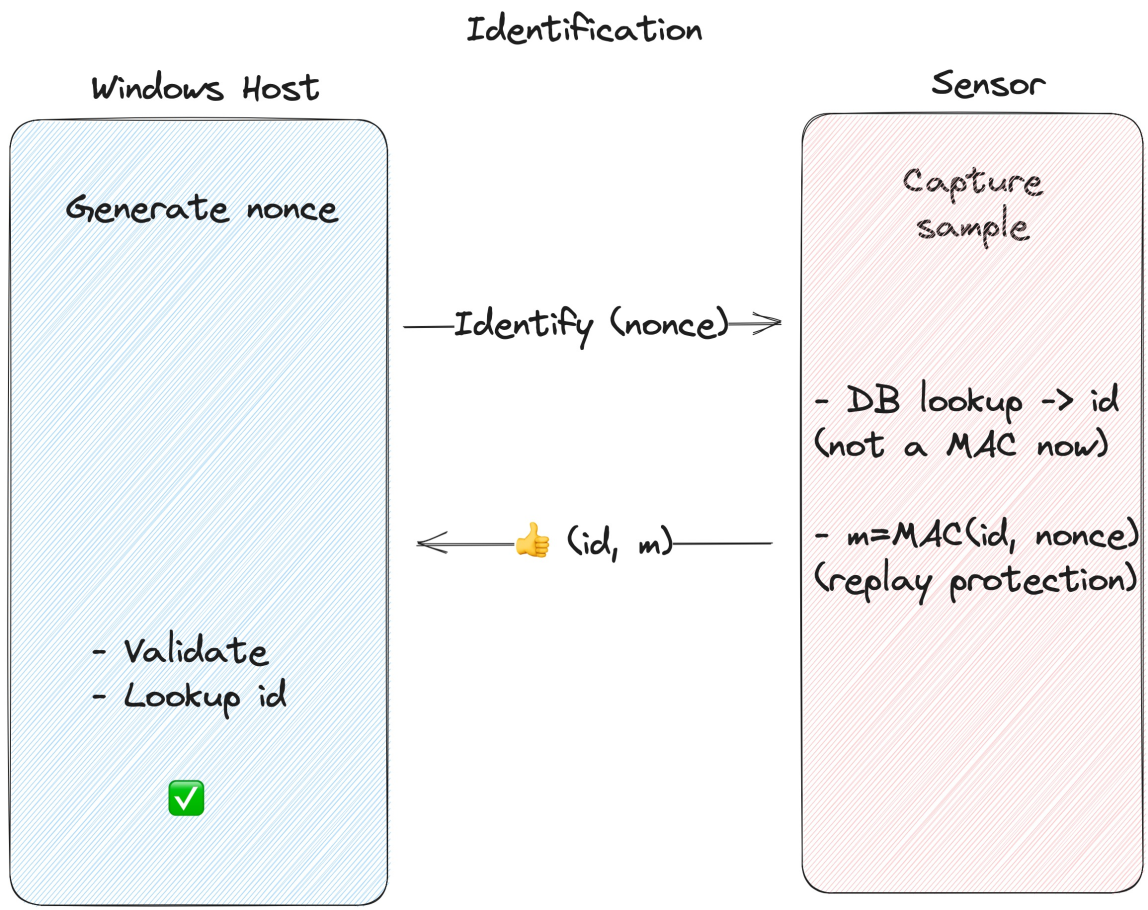
Goodix - Normal



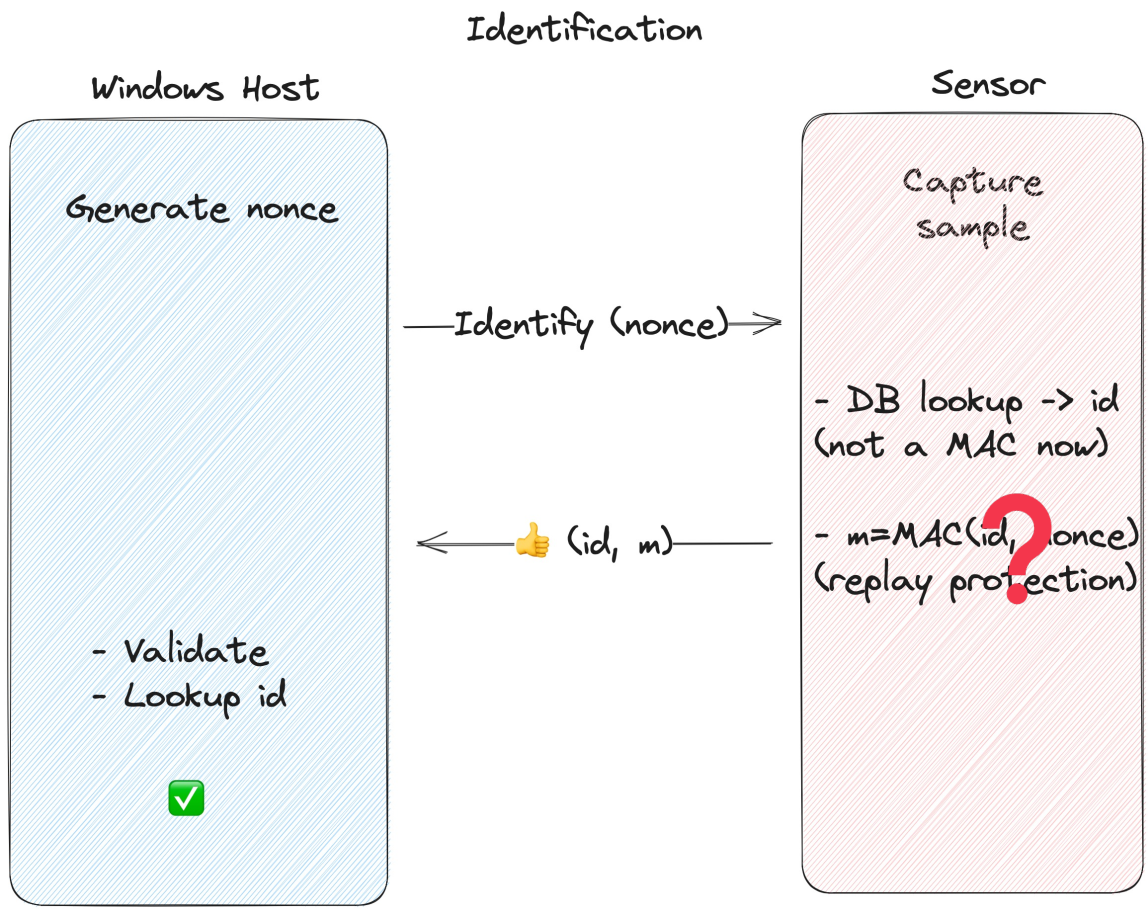
Goodix - Normal



Goodix ID – Windows DB



Goodix ID – Linux DB



Goodix – Findings Recap

- Windows Support
  - Windows driver uses SDCP
  - Dedicated template database on sensor
    - Cannot write arbitrary entries
- Linux Support
  - Linux driver does not support SDCP
  - Separate template database on sensor
    - Can write arbitrary entries
    - Does not enforce SDCP enrollment
      - *Does* calculate `identify()` response

Goodix - Vulnerabilities

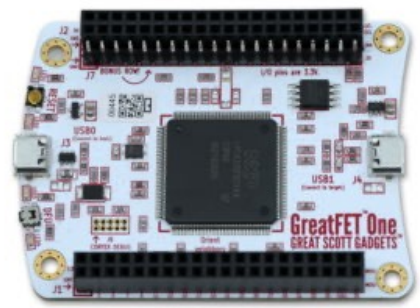
- Vulnerability Chain:
  - Info Leak: Template DB enumeration
  - Arbitrary Write to Linux Template Database
  - Unauthenticated Sensor Configuration

Goodix - Vulnerabilities

- Exploitation
  - Enumerate sensor's Windows template DB
  - Enroll attacker fingerprint using secure ID (SDCP MAC'd nonce) of existing user into Linux template DB
  - MitM USB and boot Windows
  - Rewrite configuration packet to point sensor to Linux template DB
  - Login with attacker's fingerprint

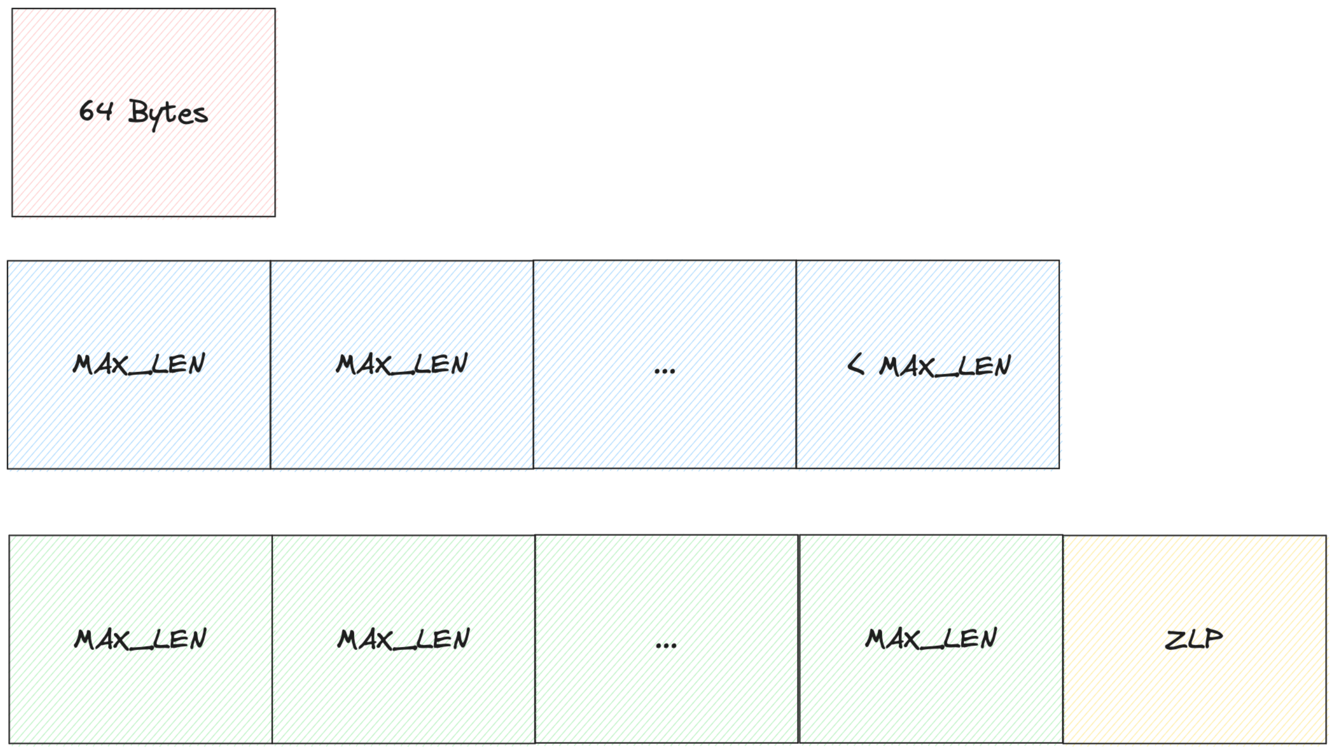
USB MitM

- USB Research in 2023, a word…
  - USBProxy
    - Not maintained (now USBProxy-legacy…)
  - GreatFET One + Facedancer

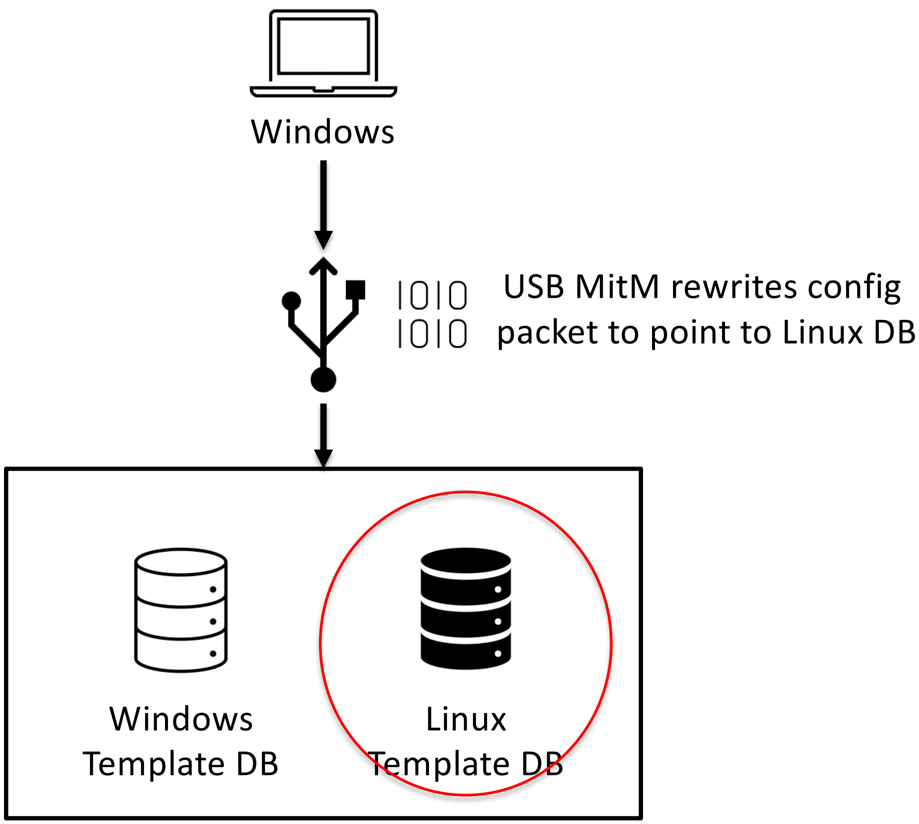


USB Bulk Transfers

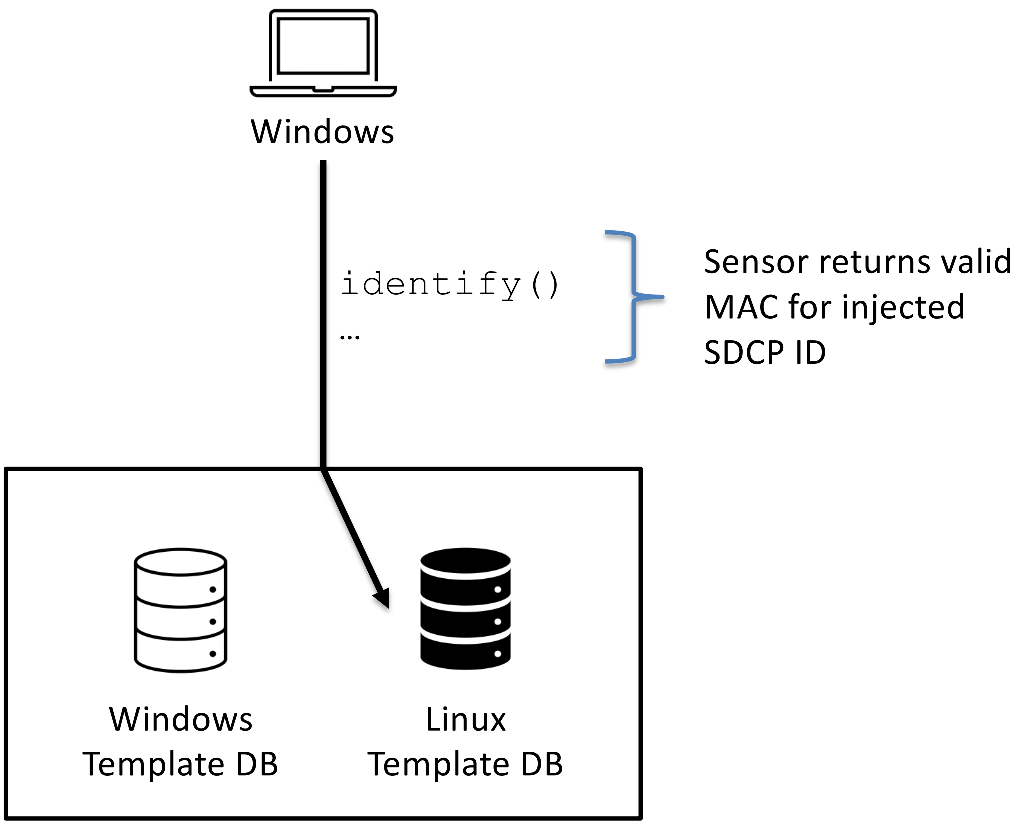
- Zero Length Packets (ZLP)



Goodix - Overview



Goodix - Overview



Goodix - Demo

Synaptics - Overview

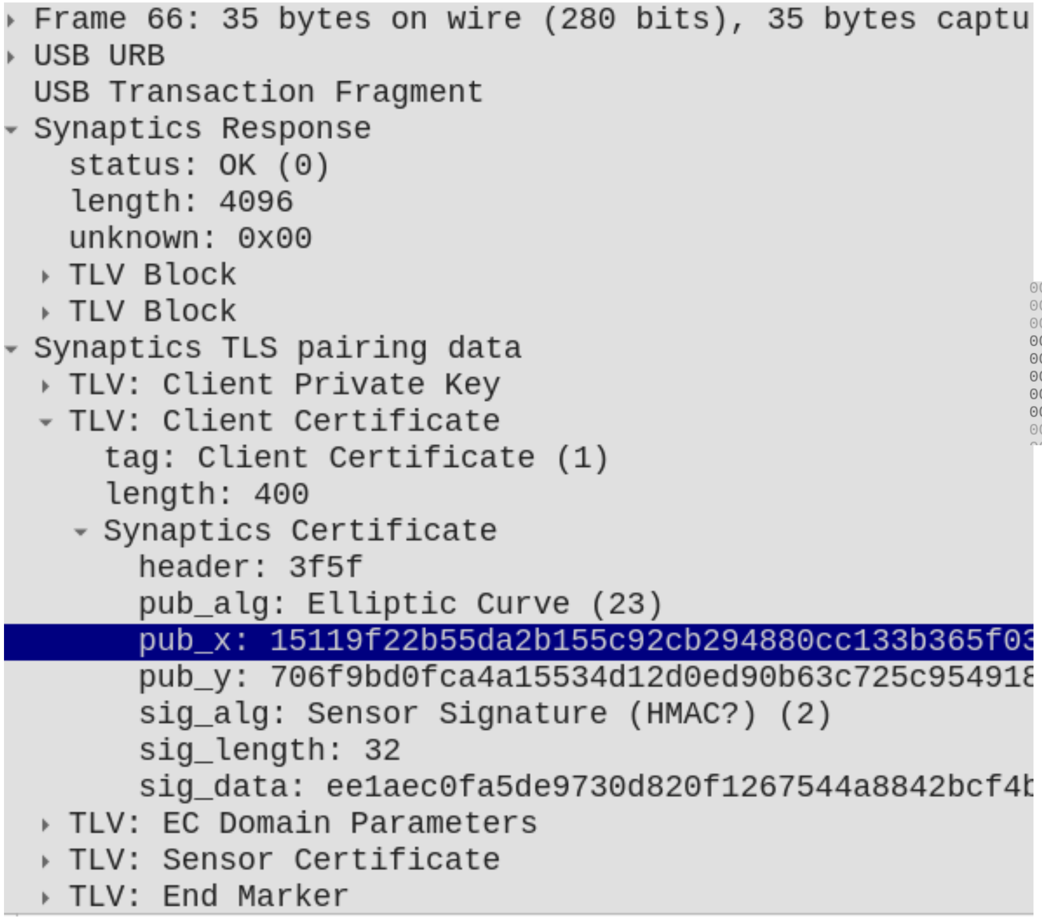
- Match on Chip (MOC)
- Limited Linux Support
- Encrypted Communication
- Encrypted Firmware

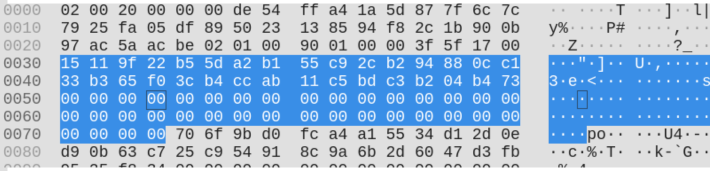
Synaptics – TLS Implementation

- USB traffic encrypted by TLS makes on the wire protocol analysis more difficult…
- Few early packets unencrypted, but most important functionality requires TLS channel
- Need to break TLS!

- Which Required
  - Lots of RE
  - Extracting TLS session keys with DBI/debugging
  - Reimplement broken TLS implementation
  - Custom protocol dissection

Synaptics – TLS Paring





Synaptics – TLS Paring

```
▸ Frame 83: 643 bytes on wire (5144 bits), 643 bytes captu
▸ USB URB
▾ Synaptics Request
    cmd: START_TLS (0x44)
▾ Synaptics TLS
  ▾ Record: Handshake
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 556
    ▾ Handshake: Certificate
        Type: Certificate (11)
        Length: 408
      ▾ Certificates
          Chain length (broken): 400
          Certificate length (broken): 400
          Unknown: 0x7719
        ▾ Synaptics Certificate
            header: 3f5f
            pub_alg: Elliptic Curve (23)
            pub_x: 15119f22b55da2b155c92cb294880cc133b365f0
            pub_y: 706f9bd0fca4a15534d12d0ed90b63c725c95491
            sig_alg: Sensor Signature (HMAC?) (2)
            sig_length: 32
            sig_data: ee1aec0fa5de9730d820f1267544a8842bcf4
    ▸ Handshake: Client Key Exchange
```

BLACKWINGINTELLIGENCE

Synaptics – TLS Paring

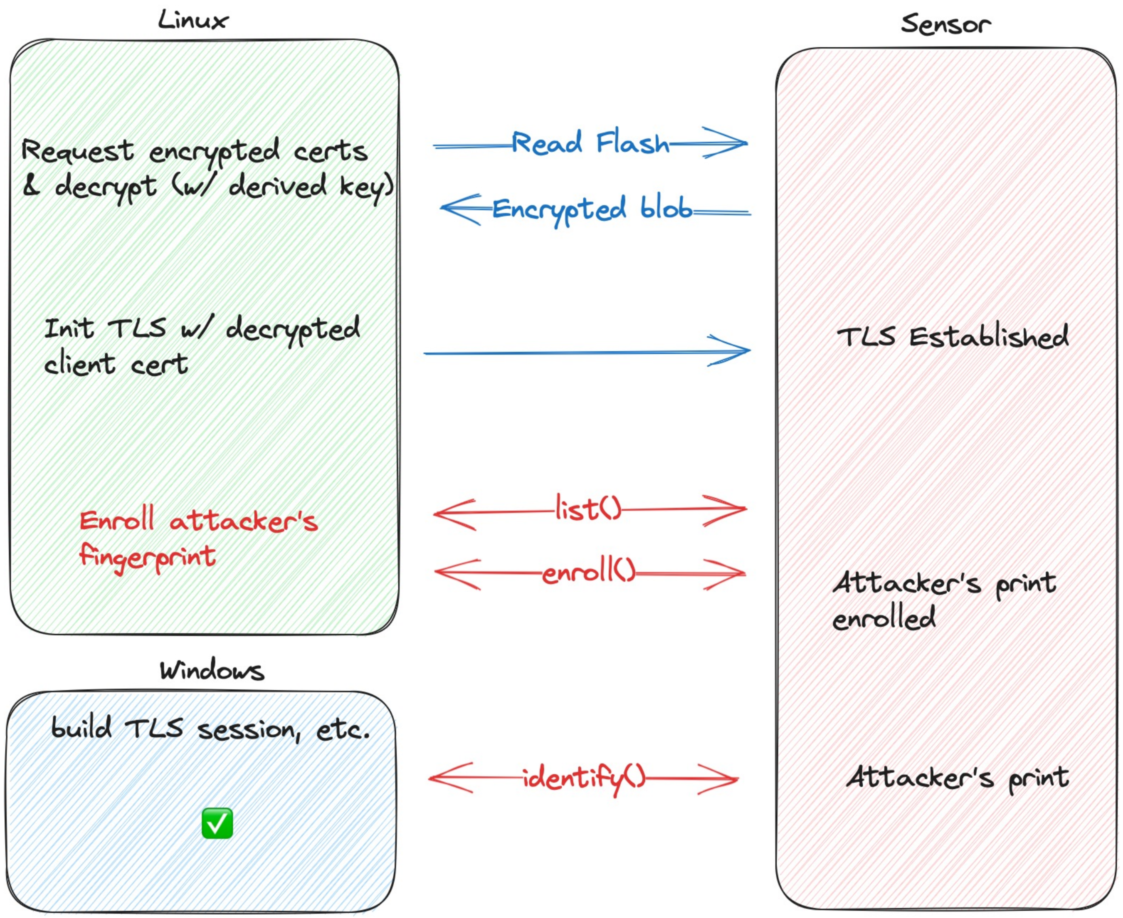
```
▸ USB URB
▾ Synaptics TLS
  ▸ Record: Application Data
▾ Synaptics Response
    status: OK (0)
    recid: 91569585ff0142b4ad1f0e85b55a3213
    response length: 36
    sdcp length: 0
    dbrecord length: 111
    identify data: 7c080000000000000000000000000100000000000000000000
  ▾ DB Record
    ▾ DB TLV - Parent record id
        tag: 0 (Parent record id)
        length: 16
        data: 91569585ff0142b4ad1f0e85b55a3213
    ▾ DB TLV - OS ID
        tag: 1 (OS ID)
        length: 76
        identity type: WINBIO_ID_TYPE_SID (0x00000003)
        identity sid length: 28
        identity sid: S-1-5-21-4100066640-3526454011-486358429-1001
    ▾ DB TLV - Finger pos
        tag: 2 (Finger pos)
        length: 1
        fingerpos: WINBIO_FINGER_UNSPECIFIED_POS_01 (0xf5)
```

Synaptics - Vulnerabilities

- Vulnerability Chain
  - Insecure Default Configuration
    - SDCP is off by default – would mitigate this attack
  - Host Derived Cryptographic Keys
    - Derived from machine BIOS/ACPI product name and serial number
  - Sensor Database Enumeration
    - Once inside TLS
  - Arbitrary Fingerprint Enrollment
    - Once inside TLS

BLACKWING INTELLIGENCE

Synaptics – TLS Attack



Synaptics - Demo

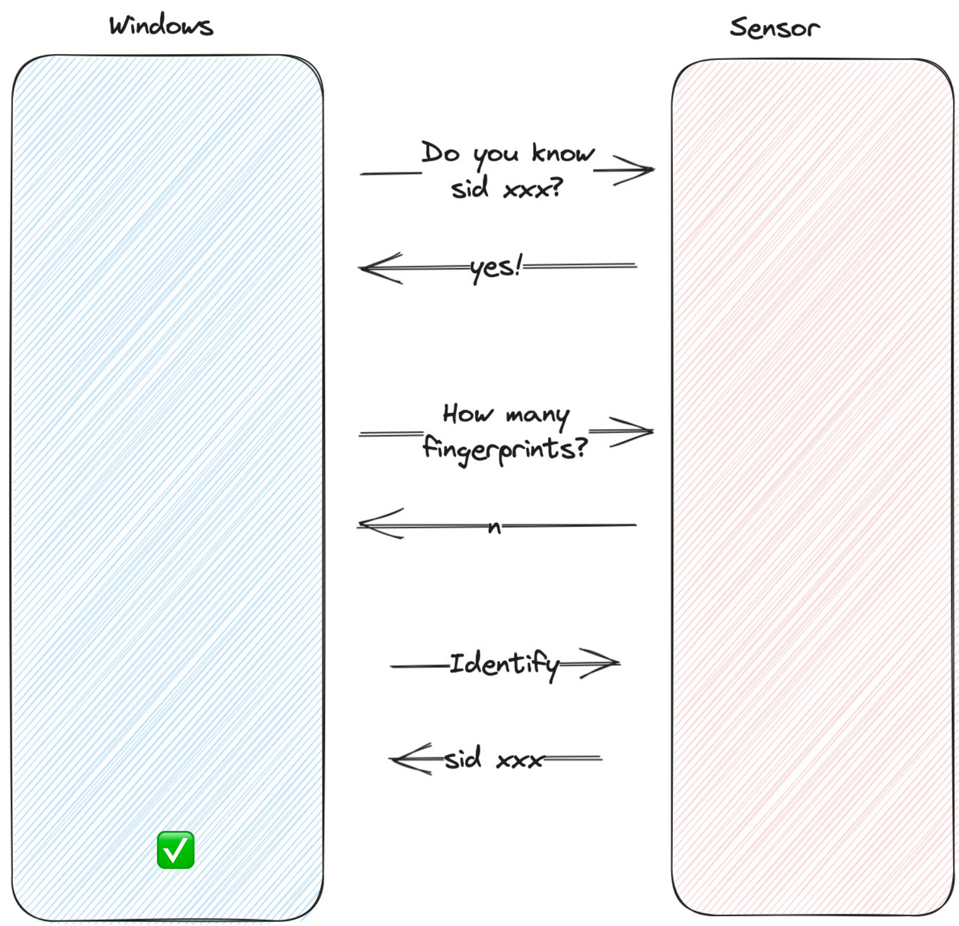
ELAN - Overview

- Match on Chip (MOC)
- Unknown custom bus & hardware connector
- Microsoft branded
- HARD! 😬

ELAN - Vulnerabilities

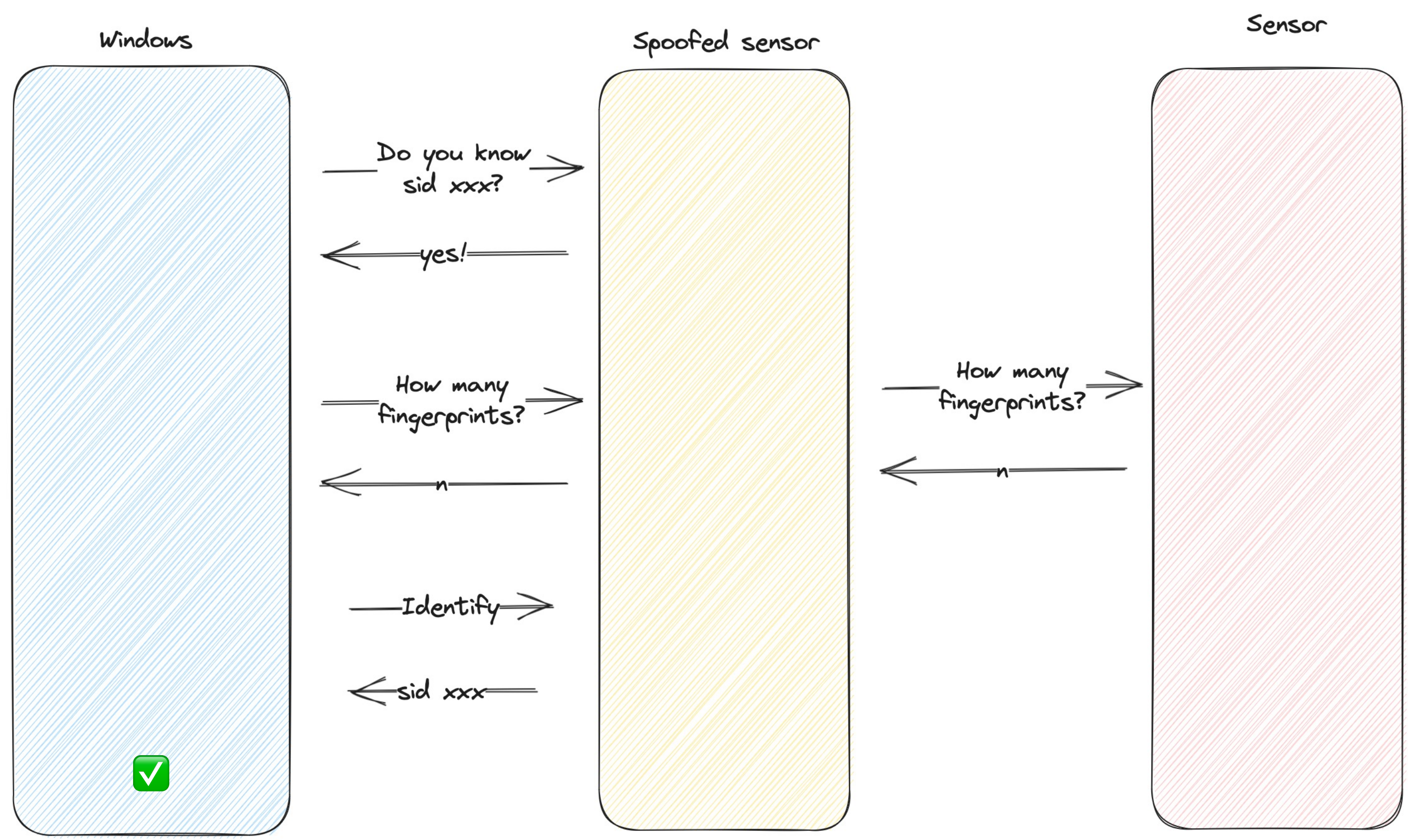
- Vulnerability Chain
  - Sensor Spoofing
  - Info Leak – Valid SID

ELAN - Vulnerabilities



ELAN - Vulnerabilities

ELAN - Vulnerabilities



ELAN - Vulnerabilities

- Why is number of fingerprints queried?
  - Unclear, but possibly a (weak) protection against someone plugging in a Type Cover that's configured with the same SID
  - If the number of fingerprints reported by the sensor doesn't match the host's expectations, the driver will erase the chip
  - Trivially bypassable by simply querying the actual sensor

ELAN - Vulnerabilities

- Exploitation:
  - Plug-in spoofed device
    - Advertise sensor PID / VID
  - Observe valid SID from Windows driver
  - Return number of fingerprints (PoC hardcoded to one)
  - Initiate Fingerprint Login on Windows
  - Send Valid Login Response From Spoofed Device

ELAN - Demo

3 Conclusion

Conclusion

- **Full authentication bypass on all three targets**
- Inconsistency between vendors
  - Common issues:
    - Code quality
    - Misunderstanding of SDCP by developers
    - Logic issues – especially non spec'd commands / support for other OSs
    - Unauthenticated attack surface
    - Info leak by design
      - All sensors leak valid IDs to support fingerprint option on login screen

Conclusion

- Recommendations
  - Make sure SDCP is enabled
  - Enhance SDCP to prevent this class of vulnerability
  - Vendors – Have a qualified 3rd party audit your implementation!

Conclusion

- Future work
  - Additional Hardware and Firmware Research
    - Firmware Security
      - Potential memory corruption for code execution
      - Hidden functionality
    - HW vulns put secrets at risk
      - JTAG, decapping, storage access, glitching, power analysis, etc.
  - Other targets
    - Linux, Android, Apple

Thanks

- MORSE
- Windows Hello Team
- Blackwing Team
  - Chris Williams
  - Ricardo Lobo