Security Analysis of a Fingerprint-Protected USB Drive

Ben Rodes*, Xunhua Wang
Dept of Computer Science
James Madison University

benjaminrodes@gmail.com, wangxx@jmu.edu

*With Department of Computer Science, University of Virginia
Fingerprint-protected USB

No fingerprint, no access to data (on the private drive)
Our Work

Security analysis

1. Is the private data private?
2. Are the reference templates private?
   - Template/fingerprint privacy is debatable

Goals

1. Bypass fingerprint authentication (to access private data)
2. Recover fingerprint reference templates (i.e., retrieve and interpret)
Disclaimer

"Ladies and gentlemen: the story you are about to hear is true. Only the names have been changed to protect the innocent."

– Dragnet

?????? = Alice’s Fingerprint Drive (AliceFDrive)
Our Target: AliceFDrive

- Integrated fingerprint scanner
- 8 Gigabyte drive split over three partitions
  1. Public read-only
  2. Public
  3. Private: fingerprint-protected!
- Automatically installs required software; supports Windows only
- No security documentation
Fingerprint Characteristics

- Three levels of patterns and characteristics
  - Global
  - Local: most popular
  - Very-fine (require high-quality scanners)

Local Level Characteristics

Location($x, y$), angle($\theta$), type($t$), quality($q$)

Fingerprint Minutia Matching

Reference template

Database

Minutiae Extraction

Enrollment

Authentication

Minutiae Extraction

Fresh sample

Match?

"Is she Alice?"

Y/N?
Attack Strategies

Hardware Reversing

Potentially Destructive

Disk Dump

Didn’t Work

Software Analysis

Our Focus
Authentication Bypass

Enrollment

Authentication

Minutiae Extraction

Reference template

Database

Match?

Fresh sample

Reference template

Alice’s

“She is Alice!”

Y/N?
AliceFDrive Modules (DLLs & Executable)

- Tools: IDA Pro, OllyDbg

- One executable
  - AliceFDrive.exe

- Five DLLs
  - PasswordBank.dll
  - PSDK4_SS500A_PTFV.dll
  - PTFVLib.dll
  - LTTS1NDUT176.dll
  - LTTUSB.dll

- Stored on the local machine
Points of interest

From IDA Pro

Do not have parameter types: crucial parts missing

PTSDK4_SS500A_PTFV.dll Exports
PTSDK4_SS500A_PTFV.dll

SDK?
  PKSDK4_SS500A_PTFV.dll

API?
  bAPI4_

Documentation? Google!
  Only one related source returned by Google
  Not exactly same but similar module names
    PTKSDK_WISCMOS32_PTFV.dll
    PTFVLib.dll
### 3. SDK Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>bAPI4_OpenDevice</td>
<td>Open the fingerprint USB device (unused)</td>
</tr>
<tr>
<td>bAPI4_OpenSensor</td>
<td>Open fingerprint sensor</td>
</tr>
<tr>
<td>bAPI4_CloseSensor</td>
<td>Close fingerprint</td>
</tr>
<tr>
<td>bAPI4_Option</td>
<td>Options on the PC</td>
</tr>
<tr>
<td>bAPI4_GetImage</td>
<td>Get the gray fingerprint image</td>
</tr>
<tr>
<td>bAPI4_GetBinaryImage</td>
<td>Get the binary fingerprint image</td>
</tr>
<tr>
<td>bAPI4_StopImage</td>
<td>Stop the get image</td>
</tr>
<tr>
<td>bAPI4_CheckSensorStatus</td>
<td>Check sensor status</td>
</tr>
<tr>
<td>bAPI4_HMFVOpenLib</td>
<td>Enable the fingerprint algorithm functions</td>
</tr>
<tr>
<td>bAPI4_HMFVCloseLib</td>
<td>Disable the fingerprint algorithm functions</td>
</tr>
<tr>
<td>bAPI4_HMFVStartEnroll</td>
<td>Start the fingerprint enrollment</td>
</tr>
<tr>
<td><strong>bAPI4_HMFVEnroll</strong></td>
<td>Fingerprint enrollment</td>
</tr>
<tr>
<td><strong>bAPI4_HMFVVerify</strong></td>
<td>Fingerprint verification</td>
</tr>
<tr>
<td>bAPI4_HMFVSetParas</td>
<td>Set parameter to fingerprint algorithm</td>
</tr>
<tr>
<td>bAPI4_HMFVGetParas</td>
<td>Get parameter from fingerprint algorithm</td>
</tr>
</tbody>
</table>

*This document also has details about function parameters!*
Break #1 – Authentication Analysis

5.11 Fingerprint Verification

Synopsis: bAPI4_HMFVVerify(int iResolution, int iWidth, int iHeight,
BYTE *pFingerImage,
BYTE **ppEnrolledfeatures, int iEnrolledNum,
int *piMatchedID,
int *piStatus)

Description: Call this function to do fingerprint verification.

*piStatus (output parameter)

- HMFV_STS_VF_SUCCESS 2
- HMFV_STS_VF_FAIL 1
- HMFV_STS_VF_POORIMG -1
- HMFV_STS_VF_ERROR -2

Dynamically changing piStatus allows any fingerprint to pass authentication
Bypass Implementation Details

- Created a persistent change made using OllyDbg to PTFVLib.dll
  - JZ → JMP (0x74 → 0xEB)
  - TEST → SUB (0x85 → 0x2B)
- piMatchedID must be 0
  - We were lucky (initially)
- Easily disseminated and used by grandpas and script kiddies
Authentication Bypass

Enrollment
Authentication

Minutiae Extraction
Reference template
Database

Minutiae Extraction
Fresh sample

Why bother?
We have the data!

Match?
Y/N?

"She is Alice!"

Alice's Reference
ppEnrolledfeatures

5.11 Fingerprint Verification

Synopsis: bAPI4_HMFVVerify

- BYTE *pFingerImage,
- BYTE **ppEnrolledfeatures,
- int iEnrolledNum,
- int *piMatchedID,
- int *piStatus)

Description: Call this function to do fingerprint verification.

- ppEnrolledfeatures: contains the reference templates
- iEnrolledNum: number of enrolled features to be verified
- Observably altered by enrollment
Minutia Template Recovery

**Goals**

1. Retrieve the reference templates automatically from the device.
2. Determine the format of the reference templates.

**Storage location**

1. Follow AliceFDrive’s code to access reference templates (ppEnrolledFeatures)
2. Analyze the target code

**Minutia format**

1. Create/analyze custom templates
2. Analyze the enrollment code
The minutia reference templates are read into memory and then processed

- bAPI4_ReadSecureArea(…) returns encrypted reference templates
- Processing code likes AES T-box implementation
  - 16 byte AES key of all 0xFF
Custom Templates

Synopsis: bAPI4_HMFVEnroll(int iResolution, int iWidth, int iHeight, BYTE * pFingerImage,
BYTE **pEnrolledFeatures DWORD *pwEnRetSize,
int *piStatus)

Description: Call this function to do fingerprint enrollment.

- *pEnrolledFeatures: minutia reference template; our target
- *pFingerImage: fingerprint image to be enrolled

What format is this?

Trace code; trial and error
Image Format
Custom Template Generation

- Differential analysis
  - Create fingerprint variants: add/remove a minutia point
  - Study the corresponding changes

Fingerprint image source: Maio and Maltoni, Direct gray-scale minutiae detection in fingerprints. Reprint with permission from IEEE. © IEEE
Minutia Template Format

- No observed compliance to any standard
  - ISO, CDEFF, ANSI/NIST
- Our discovery
  - A fixed-size header
  - Six bytes for each minutia point

```
0x09
Still unknown
```

- A fixed-size header

```
Minutia point 1
(5 bytes)
```

- Six bytes for each minutia point

```
Minutia point n
(5 bytes)
```

- Footer for point 1

```
Footer for point 1
```

- Footer for point n

```
Footer for point n
```
Minutia Template Format

• Five-byte body: enrollment code analysis
  • 11 bits – \(x\) coordinate (relative to an origin)
  • 11 bits – \(y\) coordinate (relative to an origin)
  • 2 bits – type (ridge ending or bifurcation)
  • 8 bits – angle
  • 8 bits – quality (?)

• One-byte footer
  • 8 bits (the footer byte) – unknown

Is this interpretation correct?
Reference Template Mapping

Interpreted minutia template points

Enrolled fingerprint

Mapping

Fingerprint picture source: Maio and Maltoni, Direct gray-scale minutiae detection in fingerprints. Reprint with permission from IEEE. © IEEE
Stored Minutia Mapping

Minutia template recovered from AliceFDrive (encrypted on AliceFDrive)

Alice’s fingerprint
Summary of Vulnerabilities

- Verification function
  - Easy to bypass
  - Compromised private data
- Minutia reference templates
  - Have been recovered by our own code
Quick Fix with Fuzzy Vaults?

AliceFDrive has to use reference templates in cleartext
- Matching is close matching, not exact
- Reference templates cannot be hashed

Fuzzy vault schemes may offer a solution
- Set based (minutia are sets)
- Secure data and reference templates
Quick Fix with Fuzzy Vaults?

Extract (enrollment)

$E_k(m)$

USB stores $p$, $E_k(m)$, but **not** templates

$s'$

$s$

$p$

$k$
## Fuzzy Vault: Gap between Theory and Practice

<table>
<thead>
<tr>
<th>Theory</th>
<th>Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Juels and Sudan, 2002/2006</td>
<td>Nandakumar et al., 2007</td>
</tr>
<tr>
<td>Fingerprints are represented as a set of</td>
<td>Displacement and rotation causes some issues</td>
</tr>
<tr>
<td>minutiae points</td>
<td>Pre-alignment required</td>
</tr>
<tr>
<td>Existing fuzzy vault schemes</td>
<td>Need to store additional data (helper data) for</td>
</tr>
<tr>
<td></td>
<td>existing FE/FV schemes</td>
</tr>
<tr>
<td>① Set-wise comparison: fuzzy</td>
<td>May weaken security</td>
</tr>
<tr>
<td>② Element-wise comparison: exact or very</td>
<td>Heuristic security only</td>
</tr>
<tr>
<td>close</td>
<td></td>
</tr>
<tr>
<td>Provable security</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

We have shown

1. The fingerprint-protected device fails to protect the security of private data
2. We can recover significant data on enrolled fingerprints
   - AliceFDrive may be worse than regular USB drives
     - Not only lose your data but also fingerprints

All similar devices may be equally insecure

No quick fix: existing fuzzy vault schemes may not be adequately secure

Final thought: Obfuscate or use a different paradigm
Questions?