WHITE PAPER FOR PUBLIC DISTRIBUTION

InformationRiskManagement

Biologger – A Biometric Keylogger

An IRM Research White Paper by

Matthew Lewis

Information Risk Management Plc 8th Floor Kings Building Smith Square London SW1P 3JJ
Tel: +44 (0)20 7808 6420 Fax: +44 (0)20 7808 6421 info@irmplc.com http://www.irmplc.com

IRM Research

Information technology constantly changes and advances. IRM is dedicated to keeping pace with new technology and continuing to innovate in the field of information security. This ensures that we are well informed of new issues and technologies, expanding our knowledge and providing world class services to our clients.

If you would like further information about the subject discussed in this paper then please send a request via the following link:

http://www.irmplc.com/index.php/164-Enquiry

Biologger – A Biometric Keylogger

Introduction

Biometric systems comprise electronic devices, and as such can utilise common electronic transports for the transmission of biometric related data. With biometric access control and identification systems, users will typically present their biometric to a sensing device, which in turn may transmit data pertaining to that biometric to a server or secondary processing unit to perform biometric comparisons and auditing functions. Following this matching process, further electronic signals will be generated, perhaps to open a door, or to issue a message to a terminal to inform whether or not a user has been identified/verified.

In this paper we realise a proof-of-concept implementation of a biometric keylogger, or "Biologger". While conventional keyloggers are typically used to obtain passwords or encryption keys to circumvent specific security measures, our Biologger will aim to capture biometric-related data between a biometric device and other processing units, to be used and exploited in a number potential attack vectors against the biometric system, such as manipulation of biometric data and control signals, as per traditional man-in-the middle attacks.

Organisations across a number of different sectors are beginning to implement biometric systems as part of their physical and logical access controls, while a number of these systems and devices are configured to integrate with existing infrastructures for ease of deployment, such as through the use of IP protocols. It is properties such as this that we seek to explore and exploit as part of a proof of concept construction of a Biologger.

While the research surrounding this paper focused on a specific fingerprint access control device, the overall aim of this paper is to highlight the possible attacks and risks within this domain. We postulate that without adequate protection of biometric data and control signals, similar techniques to those described in this paper may be successfully applied to other biometric modes, such as face and iris recognition access control systems. We assume the reader holds a basic understanding of the core components and principles of biometric systems. For further information in this area the reader is referred to [1].

Scenario Configuration

The configuration examined by IRM is illustrated in Figure 1. The fingerprint access control device tested utilises UDP as the main data transport between itself and management software running on a remote server. The management software is used to maintain control over a network of fingerprint devices, each of which is allocated a static IP address. The management software is also used to maintain a backup database of the system users and their biometric data, and to maintain an access control audit log of authentication attempts.

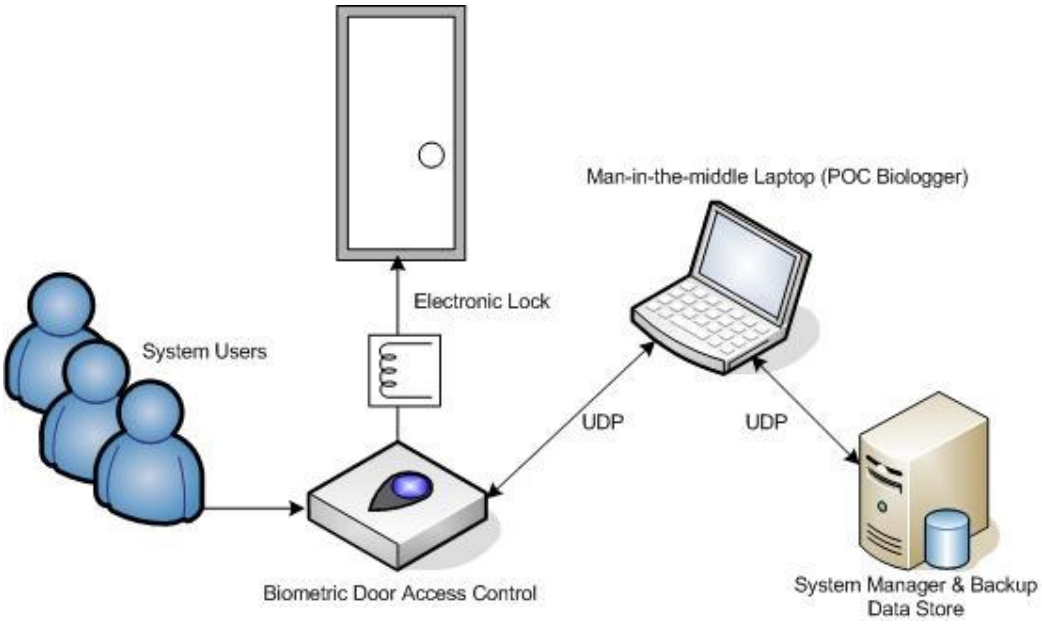


Figure 1. Fingerprint door access control scenario configuration

To emulate the possibilities of an in-line Biologger, we chose to proxy all traffic between the fingerprint device and management software through a third man-in-the-middle device which was used to examine this data exchange. We note that although the ability to do this would require privileged access on the device or server, this was sufficient as a proof of concept Biologger. Similarly, we note that truly motivated attackers might conduct similar research on access control devices that they have acquired within their own environments.

Initial investigation of the configuration described above revealed that all biometric matching was conducted within the access control device. This differed from our expectations of biometric samples being transmitted to a backend server for input to a matching algorithm, which would subsequently issue appropriate accept or reject signals to the device and door lock. The management software however incorporated functionality to upload and download user templates to specific access control fingerprint devices within the network, and so inspection of all such data flows between the device and management software would form the root of this research.

Protocol Investigation

Our Biologger was configured to store all data passing to and from the biometric device and management software. A number of user actions were performed on the system such as user enrolments, deletions, and upload/download of templates between the biometric device and management server. The captured data was then analysed offline in order to inspect the protocol used between device and server. Out initial aim here was to distinguish between biometric, user and control data.

It was determined that upon each new issuance of a command or data from/to the biometric device, a non-variable 6-byte message is sent as a form of "wake-up" message, indicating that further data is to follow:

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

The device/management server then replies with a message that contains two variable bytes, which appear in all subsequent messages relating to the current data exchange – a primitive form of session identifier. Two further variable bytes were also identified in all subsequent messages; however it was not possible to determine the

derivation of these byte values. It appeared that they represented a form of checksum over the current message; however we were unable to determine the checksum calculation during this research. Since the session identifier changed for each new data transfer, the 2-byte checksum would therefore also change even if sending the same message as previously sent. We were unable to identify the checksum computation, rendering the ability to manipulate data on-the fly difficult. This would be investigated further in a later stage of this research.

While inspecting the various messages captured with our Biologger, messages of 300-350 bytes in length were deemed likely to contain biometric template data. After further inspection of these messages we were able to determine the fields identified in Figure 2 with a fair degree of certainty.

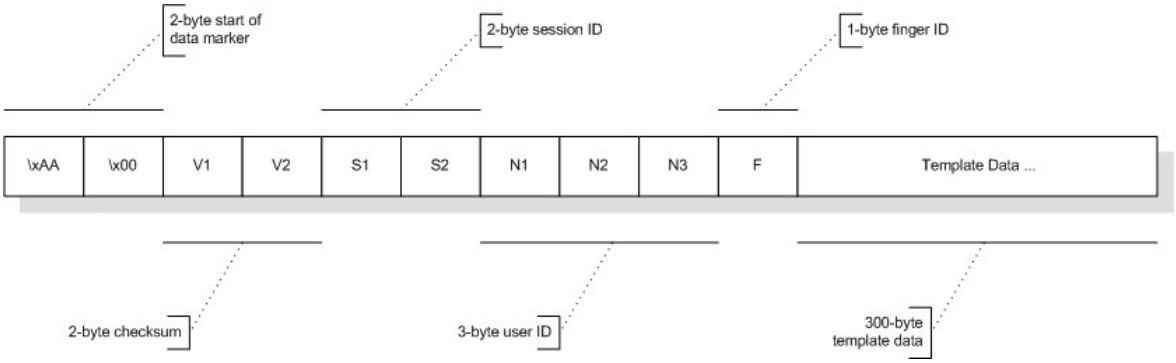


Figure 2. Identifying data fields within the protocol between device and management server

We note that no reverse engineering of software or firmware was conducted during this research. Protocol data formats were identified from observations of data captured with our Biologger only.

All data transmitted to the fingerprint device was found to begin with the same 2-byte start of data marker. The 2-byte checksum identified above then follows, with 3 bytes reserved for the user ID, and one byte for the finger relating to the template - the device provides the ability for users to enrol up to all 10 fingers as part of their user ID. By observing various upload and download messages we were able to identify the finger data values shown in Figure 3 below:

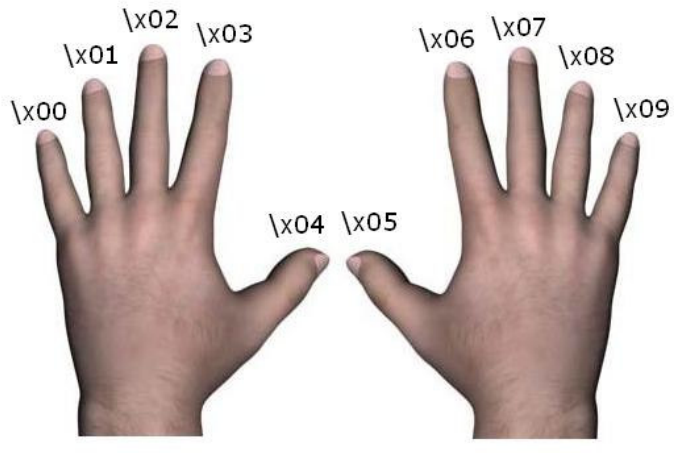


Figure 3. Finger enumeration within templates

The observations made above constitute information leakage issues. It is clear that the data transfer between the device and management server investigated is in plaintext, and that it is possible to ascertain a fair amount of information pertaining to the users and their biometrics registered within the system. In sophisticated spoofing attacks against fingerprint access control systems an attacker may be able to generate a working 3D fake finger from latent fingerprint images of a victim to be used in attempts at bypassing such access controls [2]. One of the main problems in this field is identifying the correct fingerprints to acquire for the development of spoof fingerprints. The ability to identify user IDs and specific fingers registered within the system can heavily assist motivated attackers in this area.

Similarly, certain fingers are sometimes known to contain more detail than others. For example, our thumbs can usually generate richer templates with a greater number of minutiae points than say the little finger. It is possible (depending on the matching algorithm) that a false match may be more likely against those fingers with more detail. The ability to identify those users with registered thumbs might allow attackers to therefore target those users in exploitation of the system's False Acceptance Rate (FAR).

Identifying Template Format

Having identified the 300-byte fingerprint template data from captured traffic within our Biologger, we proceeded to analyse this data offline in order to identify the file format of those templates, and the possible data and types that they contained. A brief open source investigation revealed some hints as to the template format of the system investigated. It was apparent that fingerprints within the system are categorised in terms of their fingerprint type:

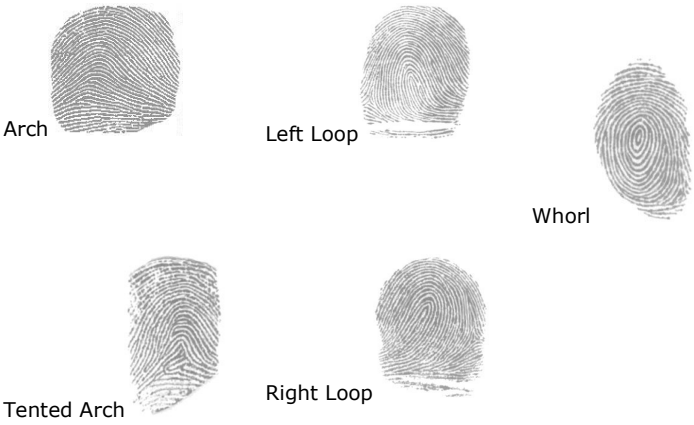


Figure 4. Examples of different fingerprint types

Categorisation can be used to speed up matching time, since checks within the matching algorithm are only made against those templates with the same finger type as identified in the newly acquired sample. Inspection of numerous template exchanges of various enrolled users between device and server from our Biologger revealed patterns that appeared consistent. The first two bytes of the 300-byte template remained constant, denoting a start-of-template marker. The third byte appeared to differ slightly in values, which upon closer inspection looked likely to represent the fingerprint type. E.g:

```
\x00 – Left Loop

\x01 – Right Loop

\x02 – Arch
```

If the type of fingerprint contained within a template can be determined, e.g. whorl, then attackers can identify those registered users that are best candidates as targets in attempts at exploiting the FAR of a system. This point is particularly pertinent for those systems operating in identification mode as opposed to verification mode where an additional identifier or authentication factor is required.

Identifying Image Data

The system under investigation during this research was seen to provide a function of backing up enrolment fingerprint images within the management server. User enrolments can be conducted on any device within a network of devices, and if requested by the management server, those enrolment images are transmitted over UDP to a backend database. Inspection of data captured with our Biologger during a request for backup storage of enrolment images revealed a stream of large UDP data packets that when combined totalled around 10,000 bytes. This data was deemed likely to contain fingerprint image data, which from our earlier open source investigation was likely to represent a greyscale image.

Without knowing any further information of the type of image or its resolution, we proceeded to input the suspected image data captured with our Biologger into an algorithm that seeks to graphically identify image data and resolution. By plotting the data values as greyscale pixels in different coordinates, it was possible to graphically inspect different resolutions of this data in order to identify any possible image data contained within. The Python code extract in Figure 5 below demonstrates this principle:

```
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
        # save the plotted image
        newIm.save("fingerprint" + str(wrap) + ".jpg")
```

Figure 5. Python code for graphically plotting different dimensions of possible image data

Following execution of this code, a number of image files are generated of different plots with varying wraparound values in the x-axis. Figure 6 below shows some of the plots generated:

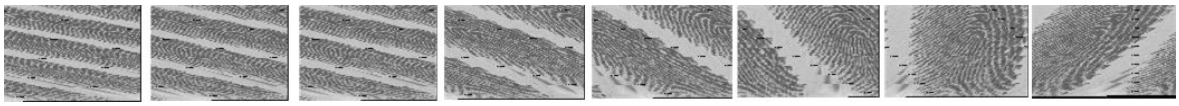
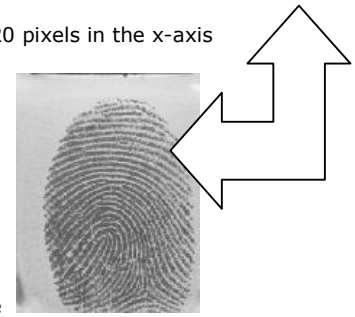


Image revealed with a wraparound of approximately 120 pixels in the x-axis



Actual image acquired by device and stored in backend database

Figure 6. Successful identification of fingerprint image data from exchange between reader and management server

The best match image above is not perfect as it is likely to contain elements of control data that we have not properly isolated and hence plotted as image data. However, this clearly demonstrates that it is possible to identify fingerprint image data with such techniques. The result of such a finding to attackers could be significant. If a good quality image can be reconstructed, then it is conceivable that techniques described in [2] could again be used to generate a 3D spoof finger of users that have obviously been registered with the system at some point.

Identifying and Replaying Control Data

Further investigation of other messages identified within the protocol between device and management server identified a number of control messages. As with template and finger image data; control data within the system investigated appeared to be transmitted in plaintext. It was observed that it was possible to add new users to access control devices within the network, without fingerprint data. I.e. the access control device could be used to provide door access based only on a PIN or password. It was straightforward to identify the following message format for adding a new user:

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

By observing the addition of various privileged users, the following information was ascertained:

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

A door unlock message was also identified, which can be issued by the management server to open specific doors within a building. This was a simple 8-byte message, which among other elements contained the device number to which the unlock signal should be issued.

As described earlier in this paper, a 2-byte checksum is used for issuance of each message, the computation of which we were unable to determine. A 2-byte session ID is also used during each main data transfer, which we were able to determine by issuing the "wake-up" message also identified earlier. Without the ability to compute the checksum and therefore generate such control signals in replay attacks, we chose to use the brute-force approach,

by trying every possible combination of 2-byte checksums with other data parts which we could determine. The Perl code fragment in Figure 7 demonstrates the principles involved in these attempts, specifically for brute forcing the "open door" command:

```
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);
                }
        }
}
```

Figure 7. Perl code for brute-forcing a checksum within an open door command

Using this technique it was possible to successfully issue a valid "open door" command to the device which in turn issued the appropriate signal to the electronic lock. It was also possible to use the subroutine above with the "add user" message in Figure 8 below, which based on the information gathered in this phase enabled us to successfully add a user "Hacker" with ID "65534" to device "0" with a PIN of "1234" , and administrator privileges on the access control device:

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

Figure 8. Perl code for brute-forcing a checksum with an add user command with administrator privileges

We note that owing to the UDP nature of the traffic involved with the system investigated, the above brute-force attempts were not 100% reliable. We also note that a multi-threaded version of the algorithm in Figure 7 could be used to improve on the time required to brute force all possible checksum values (circa 6 minutes on a 1Ghz laptop with the current single-threaded solution). While not optimum and slightly inelegant, the technique derived in this scenario serves as a proof of concept of the possibilities within this domain.

A delete user message of similar format was also identified during this research, which with the above techniques and enough time, would ultimately allow an attacker to delete all users registered within the access control device, effectively invoking a denial of service or door access to valid and registered users.

Mitigation

Most of the issues identified throughout this paper could be mitigated through proper encryption of all biometric, user and control data transmitted between devices and management servers. Robust authenticated sessions between devices and management servers would also go some way in improving the security of such

implementations. Biometrics are not secret, however biometric systems should endeavour to maintain confidentiality and integrity of biometric-related data (templates and images) at all times. This will assist in reducing the information leakage issues that might allow attackers to develop workable spoof objects or identify those users with weak or strong biometrics to be targeted in exploits against a system's FAR.

Regular log auditing of biometric access control systems should be conducted in order to identify situations of increased levels of rejections for example, which will allow administrators to identify possible issues with the system threshold levels or suspected unauthorised access attempts.

Further security can be achieved through the implementation of challenge and response mechanisms. For example, a fingerprint access control device that requests a randomly selected finger of all possible 10 registered fingers.

Conclusion

The aim of this whitepaper is not to discourage the use of biometric access control systems, but to encourage security by design with such products and their deployments, and to highlight the possibilities open to attackers or malicious employees with no more than the ability to intercept traffic between such devices and other processing units. Biometric device manufacturers and system integrators cannot rely on security through obscurity alone for the overall security of their devices and systems. Deployment of biometric access control systems within existing infrastructures such as IP networks should involve careful identification of the network traffic routing and the accessibility to biometric-related data on those networks. Without adequate protection of the confidentiality, integrity and availability of biometric access control devices and their data, the threat of "Biologging" activities within those enterprises employing such access controls is real.

References

[1] Biometrics Demystified – IRM Whitepaper, www.irmplc.com

[2] Impact of Artificial "Gummy" fingers on Fingerprint Systems – T Matsumoto, H Matsumoto, K Yamada, S Hoshino, Yokohama National University, 2002.

About the Author

Matthew Lewis is a Security Consultant at Information Risk Management Plc (IRM) where he performs a range of consultancy services including providing advice to clients about the use of biometrics. Prior to working at IRM, Matthew spent three years at CESG (the UK Government's Information Assurance arm) researching the security capabilities of biometric systems and advising Government about their use. Matthew has presented at many international conferences on the subject of biometrics and co-administered the UK Biometrics Working Group.

About IRM

Information Risk Management Plc (IRM) is a vendor independent information risk consultancy, founded in 1998. IRM has become a leader in client side risk assessment, technical level auditing and in the research and development of security vulnerabilities and tools. IRM is headquartered in London with Technical Centres in Europe and Asia as well as Regional Offices in the Far East and North America. Please visit our website at www.irmplc.com for further information.