Security issues of Internet-based biometric authentication systems: risks of Man-in-the-Middle and BioPhishing on the example of BioWebAuth

Christian Zeitz[a], Tobias Scheidat[a], Jana Dittmann[a], Claus Vielhauer[a],
Elisardo González Agulla[b], Enrique Otero Muras[b], Carmen García Mateo[b] and José L. Alba Castro[b]
[a]Dpt. of Computer Science, Univ. of Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany
[b]Signal and Communications Processing Dpt., Univ. of Vigo, Campus Universitario, 36310 Vigo, Spain

ABSTRACT

Beside the optimization of biometric error rates the overall security system performance in respect to intentional security attacks plays an important role for biometric enabled authentication schemes. As traditionally most user authentication schemes are knowledge and/or possession based, firstly in this paper we present a methodology for a security analysis of Internet-based biometric authentication systems by enhancing known methodologies such as the CERT attack-taxonomy with a more detailed view on the OSI-Model. Secondly as proof of concept, the guidelines extracted from this methodology are strictly applied to an open source Internet-based biometric authentication system (BioWebAuth). As case studies, two exemplary attacks, based on the found security leaks, are investigated and the attack performance is presented to show that during the biometric authentication schemes beside biometric error performance tuning also security issues need to be addressed. Finally, some design recommendations are given in order to ensure a minimum security level.

Keywords: biometric authentication, security analysis, methodology

1. INTRODUCTION

Internet-based applications used for electronic transactions have crucial remote-operated user identification. Biometric-based authentication systems are an alternative to classical password-based ones and even personal possession in order to minimize the possibility of manipulated transactions. Biometrics uses automated mathematical methods to recognize a person based on physiological or behavioral characteristics, like fingerprints, face geometry or speech. The ability of an application that allows recognition based on more than one human characteristic is called multi-modal. Multi-modality is acknowledged as a powerful methodology to overcome some of the weakness of mono-modal systems by weighting the contribution of every biometric characteristic according to factors that can unevenly affect the acquisition and processing of the different biometric characteristics. An Internet-based biometric system will benefit from multi-modality because error-rates coming from affordable off-the-shelf acquisition devices, like standard microphones, webcams, handwriting tablets or fingerprint scanners, can be reduced to acceptable levels when properly combined. For the application of biometric multi-modalities system security needs to be considered. Here security aspects must be ensured, such as confidentiality, data integrity, data authenticity and entity authenticity, non-repudiation and availability. In this article the development and definition of a systematic methodology is presented for analyzing security gaps at any given generic biometric authentication system oriented to the Internet. For this purpose we have chosen a selection of known security tools common from [1] like Wireshark, Nmap and Webscarab, just to mention a few, which are integrated in the methodology (see section 2.4). In order to test these guidelines for a practical applicability, a security audit was carried out on the open source web-based system for biometric authentication proposed by Otero-Muras et al. in [2].

This paper is structured as follows: The next section describes the development of the analysis-methodology. Afterwards, the description of the Internet-based biometric system is shown. Further in section four we show the strictly appliance of the analysis methodology besides presenting the results. In section five, we discuss two exemplary attacks onto the case study. Finally, in the sixth section, conclusions and some security recommendations are given.

2. METHODOLOGY

Several security analysis tools could not give us satisfactory results regarding the distributed environment and the specifics for biometric authentication security. Either existing methods are not able to analyze all criteria or the appliance is much more time consuming. This section describes our approach of a new methodology derived from OSI-model and CERT attack-taxonomy by considering criteria to several aspects. To analyze the risk of a whole system, at first, we split it into its components. As an important component, the network can be analyzed with the OSI-model, from which we also derive our checklist layers with a skill level indication. With this basis we extend our checklist with details from the CERT attack-taxonomy. Finally, our checklist is provided with a classification for the attacker's knowledge and skills, the box view model. This methodology can be used for security analysis of distributed and biometric authentication systems.

2.1 Risk analysis

To analyze the security of an Internet-based biometric authentication system with complex structures, it is useful to break down the complete system structure into the main components with its related security aspects, which have to be ensured. Figure 1 shows on the right side the server component containing web application and database, on left side the client with client application and the communication between them. In Internet-based biometric systems, furthermore the client application does interact with the biometric sensors. Below each component the corresponding important security aspects are listed, which have to ensure.

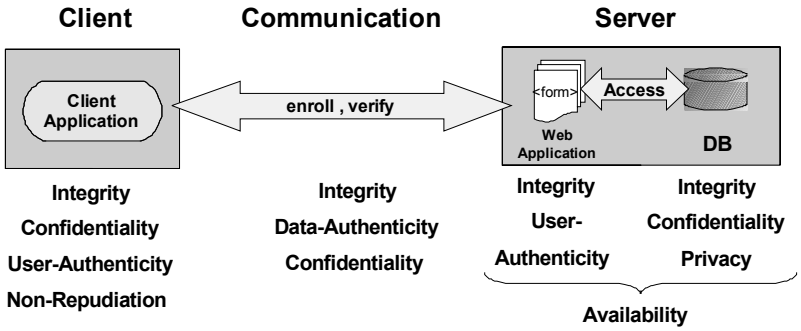

Fig. 1. Breakdown of system components.

With such a breakdown the risk analysis can be done component by component. For example: What is the purpose of the server; for Intranet or Internet? How many users have access to the database? And so on. In order to determine the risks, security audits are mostly done by simulation of different kinds of attacks. An attack is an illegal and unauthorized action to violate a system's security where the executor of such action is called attacker. In context of the study described here, attacks can be differed in five basic attack types [3], which can be passive or active. Due to their influence on the normal data flow these five attack types are interruption, modification, reading, removal and spoofing/creation. An *interruption* attack means, traffic between client and server is blocked. It is imaginable as physical broken connection. For example, a manipulated network-router can be configured in order to block data towards a certain IP address. During a *modification* the attacker manipulates the data stream nearing real time (e.g. changing contents of a web page, while it is downloaded by the client), or in non real time (i.e. manipulation of client applications as preparation for Replay or Phishing attacks). Contrary to the other four active attack types, *reading* is a passive attack where only the normal data flow is grabbed and this action is not recognizable. The *removal* (or stealing) is the most strongly noticeable attack by the client, maybe by missed acknowledgements. Mostly, for a successful stealing the attacker needs to knock out the server by a prior Denial of Service attack, in order to play the role as server. *Spoofing/creation* is the reversal of removal; here a wrong identity is pretended to the victim. For this purpose the connection between client system and server system must be also switched off as similar as removal-attack. In order to handle the huge amount of possible attacks on Internet-based authentication systems, a classification is carried out by using the seven layers (see Figure 2) of the OSI-model ([4]).

2.2 OSI-model: a new approach

Due to the specific structure of a distributed authentication system, the communication within any kind of network needs special attention during security analysis. Known as an abstract network-model, the OSI-model ([4]) can also be used to analyze data transfer to generate a kind of checklist. The OSI-model provides a layer separation by default and hence a

possibility to separate attacks, which affecting configuration, implementation or design of protocols on each layer. Also by default, the OSI-model is limited to network tests in its analysis spectrum. In order to involve any kind of application (as executable program) participated in the authentication process, whether on client or server side, we enhanced the OSI-model with a new *application layer*. To avoid confusions with the original application layer the session-, presentation- and original application layer (containing the used protocols) are merged to the new *penetration layer*. As next enhancement step a *biometric layer* was added to account for biometric tests. Figure 2 shows the enhanced OSI-model in relation to the original model.

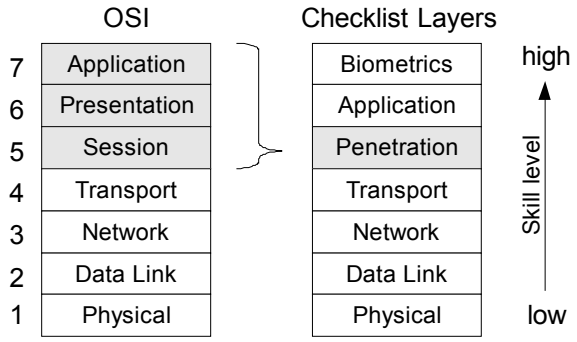

Fig. 2. OSI-model and the derived enhanced OSI-model used for checklist layers

The usage of the OSI-layers with the enhancements for applications and biometrics gives us a bottom-up rising specialization of required skills known by an attacker. The lower layers are attackable by any attacker, while the upper layers like application and biometrics needs specific knowledge about used applications and biometric techniques. A model for classifying such skill level is described in section 2.4.

2.3 CERT for more details

Another way to analyze the vulnerabilities of system components is provided by the CERT attack-taxonomy, as shown in Figure 3. The Computer Emergency Response Team (CERT [5]) is a centre of Internet expertise that analyses the Internet security. This taxonomy allows getting a detailed view of an attack referring to the actions on which component (target), aimed vulnerabilities and used tools. For classifying attacks, a combination of consideration on components with breakdown of communication layers is possible now.

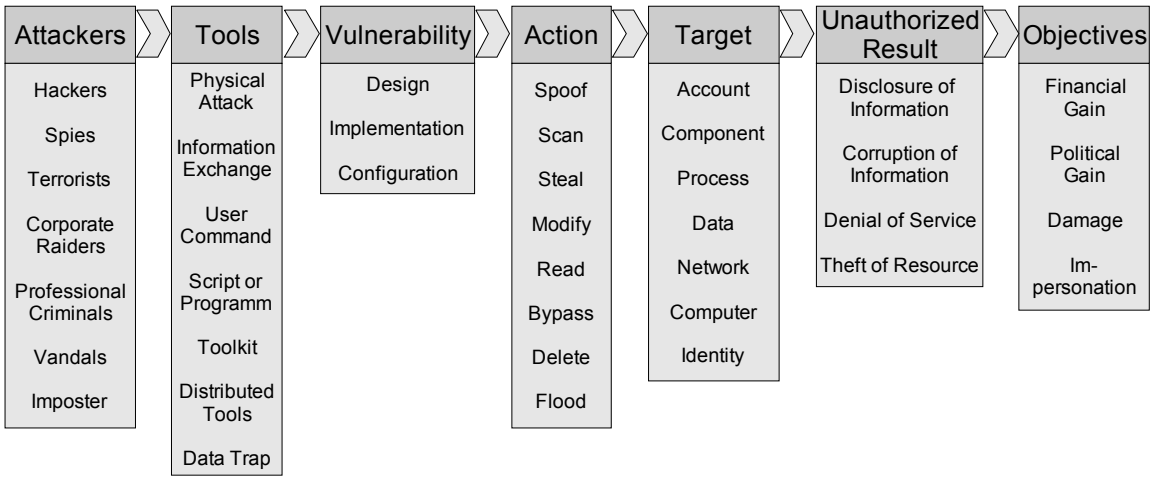

Fig. 3. CERT taxonomy including biometric attacks.

2.4 Box model views and presentation of the methodology

To refer to the a-priori knowledge and skills for running attacks, the next step is a classification by a box model view. Here it can be differed between *black*, *gray* and *white* box view ([6]). Black box means, a possible attacker needs and/or has no a-priori knowledge about the processes within the system. For example, if an attacker wants to perform a brute-

force attack to crack a password, he does not need to know the internal of the cryptographic method used for keeping the password secret. By contrast, the complete insider knowledge is covered by the white box view, while gray box describes the knowledge base placed between no a-priori and full knowledge, which can be gathered during the attack process. Finally, this box model provides a classification for the skill level of a possible attacker. It may be that on one layer different views exist, but in general it can be seen that the skill level rises over all layers (compare tables 2 to 6).

Based on the components breakdown, the enhanced OSI-model and the CERT attack-taxonomy, a more comprehensive analysis of potential attacks is derived, divided into checklist layer provided by the enhanced OSI-model (see Figure 4). As a first raw classification of a test, the box model view delivers an overview of expected skills. Furthermore, the analysis requires knowledge about the exploited vulnerability and the threatened security aspect, given by the CERT attack taxonomy. The extra information, target and action, are also known from the CERT, whereas the possible targets are taken from the components breakdown (see section 2.1), but extended by identity, which is used for biometric security analysis on biometric layer. Contrary to the CERT taxonomy, the tool column is listed beyond the action column to summarize which tools are appropriate from our tool selection and which results can be achieved shown in the last column. Contrary to the CERT attack taxonomy, the content of the columns are not always combinable, e.g. some tools are not usable on every layer or on some layers not every target is attackable.

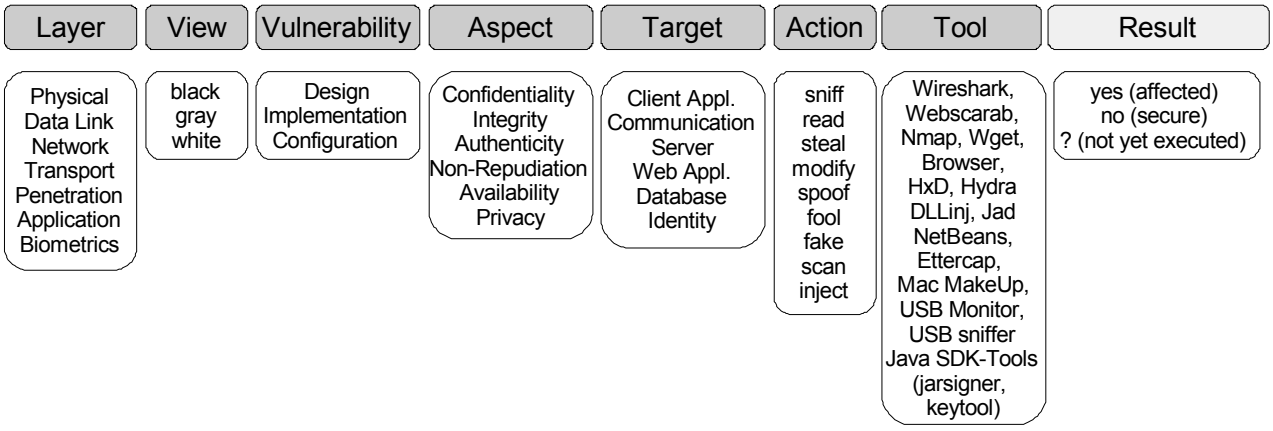

Fig. 4. Headers of the checklist with lists of values below each header component

3. CASE STUDY: REFERENCE SYSTEM BIOWEBAUTH

The Internet-based system for biometric authentication that has been audited by applying the methodology introduced in section 2.4 was originally presented by Otero-Muras et al. in [2]. The main objective on the development of this system, called BioWebAuth (Biometrics for Web Authentication, [7]), was to provide biometric authentication for single sign-on web authentication. For this purpose, this system integrates within a widely accepted Java-based open-source system for web authentication called Central Authentication Service (CAS) ([8]). The main idea behind this integration was to take advantage of the infrastructure provided by CAS to offer single sign-on web authentication, while improving security beyond basic mechanisms based on login and password, by adding biometrics. The design of the BioWebAuth system was focused on security, interoperability and usability issues. With this goal, some widely accepted standards in the field of biometrics were adopted, as summarized in the following:

- *Security*: In order to comply with the Core Security Requirements of the ANSI X9.84 standard for Biometric Information Management ([9]), SSL connections are used, and local disk writing of user samples is avoided.
- *Interoperability*: Attention has been paid to the design of client-server architecture capable of controlling any kind of biometric software or device compliant with the standard BioAPI ([10]).
- *Usability*: The user interacts with the system through a user-friendly graphical user interface. This interaction is driven by an easily configurable dialogue. Thus, verification tasks are modeled as human-machine dialogues specified by an XML document which describes the sample acquisition process and the biometric verification mode.

Figure 5 depicts a detailed block diagram of the biometric authentication system itself. Starting from a client enrolment or verification request, the successive actions and functionalities are explained as follows (see diagram numbering):

1. The biometric client application (T_{CA}) obtains from the biometric server application (T_{WA}) on the server (T_S), the XML dialogue that contains the enrolment or verification process description.
2. The client application interprets the protocol contained in the XML dialogue (T_{Com}), prompts the corresponding information to the user, acquires the biometric sample, and performs an enrolment or verification.
3. Each time a biometric sample is required, the sample is captured from the corresponding BioAPI-compliant module. For this purpose, the client application calls the *BioAPI_Capture* primitive using a Java Native Interface wrapper for the BioAPI framework. BioAPI-compliant modules are also called Biometric Service Providers or BSPs.
4. The result of the acquisition process is sent to the server bound to an enrolment or verification request.
5. The enrolment or verification process is executed in the server as a sequence of BioAPI calls.
6. The verification result or enrolment templates are stored in the server database (T_{DB}).
7. The database with the biometric verification results (T_{ID}) will be available to finally authenticate users for the web through the Central Authentication Service (CAS).

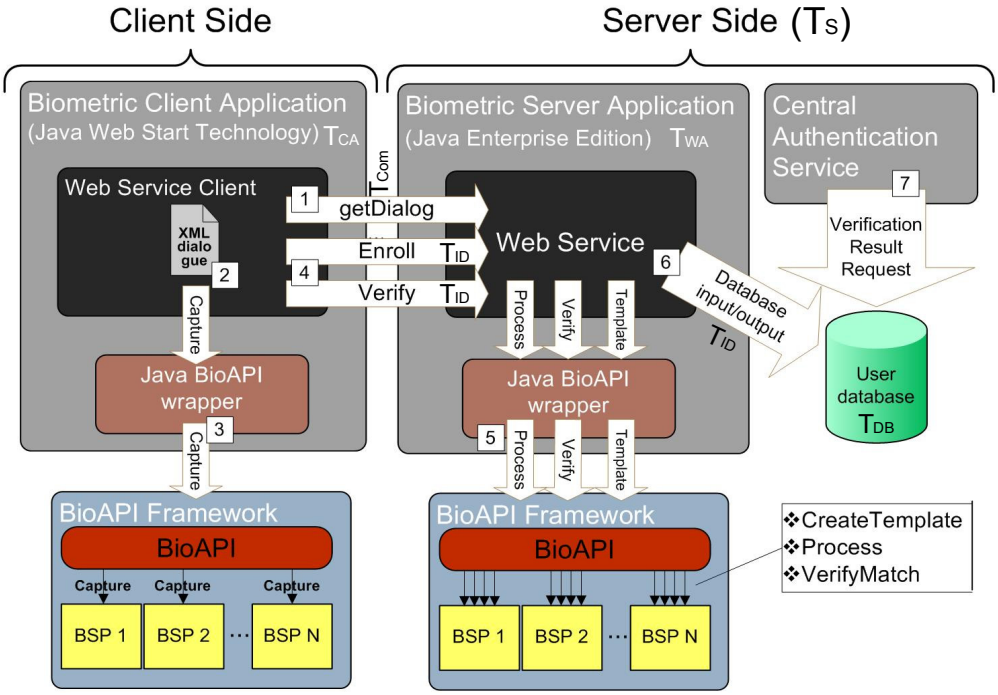

Fig. 5. Headers Building-blocks and functionality description of the biometric authentication system

As a result, any application prepared to use the original CAS for authenticating its users can also use the biometric extension for this purpose, supporting any BioAPI-compliant biometric software or devices.

4. CASE STUDY: ANALYSIS FOLLOWING THE METHODOLOGY

In this case study, the analysis was applied layer-by-layer as suggested and introduced in section 2.4, starting with the physical layer up to the biometric layer. With every layer chosen for security analysis, a more specialized skill level of an attacker is required. For example, on the physical layer the attacker's skill do not have to be very specialized by knowing details from the application or the used biometric technique. In this section we present the overall analysis results, while the fifth section selects some specific results in order to show two exemplary attacks.

In the following subsections the test results are summarized in tables by describing a test following the methodology. That means we give an overview of what view is necessary (e.g. skill level), what vulnerability is exploited, what security aspect is violated, what target is attacked with what action by using what tool. For each test the result is given as 'yes' or 'no', representing the existence of a security leak (compare Figure 4 in section 2.4).

In order to keep the result tables compact, the following syntax (see Table 1) is introduced, which provides some abbreviations for the CERT terms. The target column assigns different synonyms of components, used in section 2 and 3 of this paper, to one notation as T_x, where the x stands for a component. For example, client application and biometric client application is now assigned to T_{CA}. The vulnerabilities are abbreviated with V_x, where the x can stand for a design, implementation or configuration vulnerability. Security aspects are abbreviated with S_x, but here the x can be one of the security aspects such as integrity, availability or privacy.

Table 1. Abbreviation syntax for vulnerabilities, targets and security aspects.

Vulnerability (V)		Target (T)		Security Aspect (S)	
Design	V_D	(Biometric) Client Application	T_{CA}	Confidentiality	S_{Conf}
		Communication (getDialog, Enroll, Verifiy)	T_{Com}	Integrity	S_{Int}
Implementation	V_I	Server	T_S	Authenticity	S_{Auth}
		Biometric Server / Web Application	T_{WA}	Non-Repudiation	S_{NR}
Configuration	V_C	(User) Database	T_{DB}	Availability	S_{Avail}
		Identity	T_{ID}	Privacy	S_{Priv}

Note, in the following tables results are marked with symbols, giving additional information. The asterisk (*) indicates a different result based on tests executed on higher layers, which lead to a positive result in the actual layer. Question marks (?) indicate tests, which are not yet executed. If the tool column is marked with a minus (-), there is no special tool required, while N/A means a tool is not (yet) available for testing.

4.1 Analysis on physical layer

The physical layer is responsible for the bitwise transmitting of information, so only sniffing of bit streams is possible on this layer by executing passive wiretapping. Tools like *Wireshark* ([1]) provide a usable interpretation of such bit streams. Other tools like the *Webscarab*-tool ([1]) can manipulate the bit stream in order to flip some bits in encrypted data streams or can simulate a proxy. A proxy can access an unencrypted HTTP request and is able to redirect the request to another website. Table 1 shows the result overview on the physical layer.

Table 2. Results on physical layer.

Physical Layer Test	View	V	S	T	Action	Tool	Result
Sniffing	Black	V_C	S_{Conf}	T_{Com}	read, steal	Wireshark	yes*
Bit Flipping	Black	V_D	S_{Int} (Data)	T_{Com}	manipulate	Webscarab	?
Proxy	Black	V_C	S_{Int} (Data) + S_{Auth}	T_{Com}	modify	Webscarab	yes*

In the physical layer all tests are based on black box attacks, because there is no knowledge for any application or biometric technique necessary. Though the targets are all T_{Com}, biometric samples can be grabbed during sniffing, if they are not protected against unauthorized reading, e.g. by cryptographic methods like SSL-encryption. The two positive results are caused by a configuration vulnerability found at penetration layer (see section 4.3), which is exploited here.

4.2 Analysis on data link, network and transport layer

In this subsection the results of the data link, network and transport layer are presented together because the tests are aimed on network management like the MAC-spoofing ([11]) and the ARP-spoofing ([12]), which works with manipulation of MAC-addresses. In first test, a well-known and trusted MAC-address is assigned to an untrustworthy

network interface card (e.g. with Mac MakeUp [13] tool). In the ARP-spoofing attack an IP-address is assigned to an untrustworthy MAC-address, by feeding the network with faked information. As result of such manipulations, data packets designated for the victim can be stolen now, if the victim is already paralyzed by a DoS attack. Ettercap ([1]) is a tool, which can achieve both, a-priori DoS attack and poising of the network. However, the security analysis in this work is concentrating on distributed authentication systems and not on the configuration of a network. Under special circumstances configuration oriented tests can be useful to a security analysis, but to evaluate the configuration of the server providing biometric authentication is a study more interesting in our case. For this purpose a tool named *Nmap* ([1]) is used, which is able to find open ports and to identify the services running on these ports (Service detection). If *Nmap* is able to determine what Operating System (OS) is running on the server machine (OS detection) the attacker can tune his attack toward this OS. Table 3 summarizes the results for the 3 layer analysis.

Table 3. Results on data link, network and transportation layer.

Layer Test	Layer	View	V	S	T	Action	Tool	Result
MAC-spoofing	Data Link	Gray	V_D	S_{Auth}	T_{Com}	spoof	Mac MakeUp	?
ARP-spoofing	Data Link	Gray	V_D	S_{Int}, S_{Auth}	T_{Com}	spoof	Ettercap	?
IP-spoofing	Network	Gray	V_D	S_{Auth}	T_{Com}	spoof	-	?
Service detection (internal)	Transport	Black	V_C	S_{Conf}	T_S	scan	Nmap	yes
Service detection (external)	Transport	Black	V_C	S_{Conf}	T_S	scan	Nmap	no
OS detection	Transport	Black	V_C	S_{Conf}	T_S	scan	Nmap	yes

The first three tests are very similar. They are executed with spoofing and are still aimed on the communication component, but they require knowledge about the designed network architecture and management (gray box). The last three tests executed with *Nmap* are still black box attacks because these tests do not require any knowledge. On this layer with the shown tests specific biometric security violations are not involved, because the attacker has no possibility to access any kind of biometric data.

4.3 Analysis on penetration layer

On the penetration layer, those tests are collected, which can threat any server component in order to intrude into the server and inherit it, the so-called penetration. Some famous actions in conjunction with penetration are SQL-injection, Cross-Site-Scripting (XSS) and Buffer Overflows. During a SQL-injection the database is penetrated by hazardous SQL-queries. XSS is a method to embed a partially or fully foreign website into a known website. For more information on SQL-injection and XSS see [14] and [15]. Buffer Overflows are mostly caused by ill-treatment of oversized input values. Table 4 shows the results of the tests, executed in most cases with the *Webscarab*-tool ([1]). The SSL-bypass is a simple test to check if an unencrypted connection is possible and with that an unauthorized reading of communication could be carried out.

Table 4. Results on penetration layer.

Penetration Layer Test	View	V	S	T	Action	Tool	Result
SQL-injection	Gray	V_I	S_{Int}	T_{DB}	inject	Webscarab	no
XSS	Gray	V_I	S_{Auth}, S_{Int}, S_{Conf}	T_{WA}	inject, spoof	Webscarab	no
cmd-injection	Gray	V_I	S_{Int}	T_S	inject	Webscarab	?
Session-hijacking	Gray	V_D	S_{Conf}, S_{Priv}	T_{WA}	steal	Webscarab	yes
PW Cracking	Black	V_D	S_{Conf}, S_{Priv}	T_{DB} (user account)	crack	Hydra	yes
SSL-bypass	Black	V_I, V_C	S_{Auth}	T_{Com}	bypass	Browser	yes
Buffer Overflow	Gray	V_I	S_{Avail}	T_{WA}, T_{DB}	flood	Browser	no

The majority of tests are now carried out with gray box view, so it can be seen that the necessary skill level is raised. Seven tests are executed, four of them with the *Webscarab*-tool, two with a standard browser, and the password cracking is executed with a tool named hydra ([1]). It can be seen that the most tests listed in this layer are based on implementation vulnerabilities. Some of the tests mentioned in this layer can threaten the biometric security. For example the SQL-injection: If the attacker is able to inject hazardous code into the database he might be able to access biometric data stored in the database. Furthermore, the SSL-bypass can possibly lead to read out of biometric data during communication as discussed in section 4.1.

4.4 Analysis on application layer

Tests on the application layer (see Table 5) are focused on security leaks of applications involved in biometric authentication on the client side as well as on the server side. The type of application depends on used Application Programming Interface (API), which was in our case a Java framework. Java class files are easy to decompile (for example with *Jad* [16]) and thus, reverse engineering is possible. In combination with untrustworthy certificates a hazardous usage of a redesigned exemplar of the client application (T_{CA}) is possible. During the analysis, the tools *keytool* and *jarsigner*, provided by the Java Software Development Kit (Java SDK [17]) were able to fake such certificates. Other tests, which observe the application's memory, have shown that it is possible to read out access data like username and password. In case of our analyzed system the acquired image and/or speech sample used for biometric authentication also could be read out and even manipulated within the memory with a hex editor like *HxD* ([18]). Another way to get the samples is to hack the software drivers for the acquisition devices like webcams and microphones. In order to prevent positive results in the most of the tests of the penetration layer, the developers need to run a code review for server applications to find programming errors and bugs. This can be done with an Integrated Development Environment (IDE) like *NetBeans* ([19]). Table 5 shows the results of the tests mentioned above.

Table 5. Results on application layer.

Application Layer Test	View	V	S	T	Action	Tool	Result
Uncontrolled Download	Gray	V_C	S_{Conf}	T_{CA}	read	Wget	yes
Reverse Engineering	Gray, white	V_C	S_{Conf}	T_{CA}	decompile	Jad	yes
Faking Certificates	Gray	V_I	S_{Auth}	T_{CA}	fake	Java SDK-Tools	yes
Backtrack	Gray	V_C, V_I	S_{Conf}	T_{CA}	trace	-	yes
Memory Dumping	Black	V_I	S_{Conf}	T_{CA}	read out	HxD	yes
Memory manipulation	Black	V_C	S_{Auth}, S_{Int}	T_{CA}	modify	HxD	yes
Driver Hacking	Gray	V_D	S_{Int}	T_{CA}	replace	N/A	?
Code Review	White	V_I	S_{Avail}	T_{CA}, T_{WA}	bug tracking	NetBeans	?

Except of two tests, listed in this layer, the results show that the system is vulnerable although some of the tests require knowledge from a white box view, what means that the skill level is raised again. The main vulnerabilities on this layer are based on configuration and implementation. Furthermore, on the application layer all tests are focused either on client or on server applications. In almost every test we have great potential to threaten the biometric security. Especially the memory operations allow getting access to biometric samples stored in the memory, as presented in the next subsection.

4.5 Analysis on biometric layer

On the biometric layer (see Table 6) other multi-layered aspects should be taken into consideration like faking a biometric trait (physical or behavioral), to fool the sensors or to read the acquired sample directly from the sensor. Furthermore, the tests are more focused on the quality of the biometric algorithm. On biometric layer the typical target is not only a system component, it is the identity of a person. If an attacker wants to impersonate an identity, he only needs to get the sample of a person. The easiest way is to take a picture or a voice record of the chosen victim and simply to fake the sample. A second approach is to grab the sample at the acquisition device (sensor). The sensor tests may be easy for microphones but for webcams an USB Monitor [20] can be useful. Sniffing the sample directly from the sensor is not always unrecognizable. The usage of the *USB sniffer* tool [21] provoked a malfunction in the client application, so the positive result marked with [1] is valid only under special conditions. Possible attacks, which are focused on the biometric sample in order to impersonate an identity, are fake, theft or spoof of a sample. In our analysis the spoofing and theft are

realized by manipulating the memory of the application by the *DLLinj*-tool [22]. For the faking, a simple photo (face recognition) or voice record (voice recognition) is sufficient.

Presenting a photo to the sensor can be used as a liveness detection test for face recognition at the same time. The other two algorithm tests could not yet executed, because they require a white box view in order to fool the algorithm. The last test also requires a white box view, but no tool is available yet. While reverse sampling the biometric templates stored in the database are reverse engineered to the biometric sample in order to use it for authentication. A usable tool might be very specific for getting a positive result. In the previous subsection we have seen that the biometric samples are accessible within the memory.

Table 6. Results on biometric layer.

Biometric Layer Test	View	V	S	T	Action	Tool	Result
Sniffing (Sensor)	Gray	V_D	S_{Conf}	T_{ID}	sniff	USB Monitor/sniffer	yes[1]
Replay (Sensor)	Gray	V_D	S_{NR}	T_{ID}	spoof	USB Monitor	?
Fake (Sample)	Gray	V_D	S_{NR}, S_{Auth}	T_{ID}	fake	-	no
Theft (Sample)	Gray	V_I	S_{Conf}	T_{ID}	read	DLLinj	yes
Spoof (Sample)	Gray	V_I	S_{NR}, S_{Auth}	T_{ID}	inject	DLLinj	yes
Liveness-detection (Algorithm)	Gray	V_D	S_{NR}	T_{ID}	fool	-	no
Feature Extraction (Algorithm)	White	V_D	S_{NR}, S_{Auth}	T_{ID}	fool	-	?
Tolerance (Algorithm)	White	V_D	S_{NR}, S_{Auth}	T_{ID}	fool	-	?
Reverse Sampling (Template)	White	V_I	S_{NR}	T_{ID}	invert	N/A	?

In this layer six of nine biometric security tests require a gray box view and three imply insider knowledge. We also see that data transfer from sensor to application need special attention as well as the data (biometric data and access data) stored in the application's memory. The majority of the tests are based on design vulnerabilities with more weighting on design of the biometric recognition algorithm.

5. CASE STUDY: EXEMPLARY ATTACK

After presenting the results of our analysis of selected vulnerabilities and tools we have chosen selected successful attack results from sections 4.1, 4.3 and 4.4, see Table 7 ranging from physical, penetration to application layer tests in order to present two attack scenarios. Firstly, in the first subsection 5.1 a possible Man-in-the-Middle attack on the BioWebAuth system is shown, which threatens the five security aspects confidentiality, integrity, authenticity, non-repudiation and availability. Secondly, another attack scenario (Bio-Phishing) is presented which uses a manipulated biometric client application (T_{CA}) of a possible generic system for Phishing in order to theft biometric data by an attacker, see subsection 5.2.

Table 7. Summary of selected successful attack results taken from physical, penetration and application layer.

Test No.	Layer Test	Table	View	Vulnerability	Aspect	Target	Action	Tool
1	Sniffing	physical	Black	V_C	S_{Conf}	T_{Com}	read	Wireshark
2	Proxy	physical	Black	V_D	S_{Int}	T_{Com}	modify	Webscarab
3	SSL-bypass	penetration	Black	V_C	S_{Conf}	T_{Com}	bypass	Webscarab
4	Uncontrolled Download	penetration	Gray	V_C	S_{Conf}	T_{CA}	read	Wget
5	Reverse Engineering	application	Gray	V_D	S_{Int}	T_{CA}	decompile	Jad
6	Faking Certificates	application	Gray	V_I	S_{Auth}	T_{CA}	fake	Java SDK-Tools

The Man-in-the-Middle scenario shown in the following subsection 5.1 requires all partial results, while the Bio-Phishing scenario shown in section 5.2 requires the last three entries of table 7 (all on gray box level).

5.1 Data modification by Man-in-the-Middle (MitM)

By the strictly appliance of the checklist from section 4.1, initially no results in physical layer were found. But after running the tests on penetration layer, where SSL-bypass with a proxy (i.e. *Webscarab*) succeeded (Table 7 test 3), reading and modification of data stream with *Wireshark* and *Webscarab* on physical layer were successful (see Table 7 test 1 and 2). The next result found on penetration layer allows uncontrolled access with *Wget* (a Unix/Linux command line tool) to the client application binaries from BioWebAuth (test 4 in Table 7). On the application layer these binaries could be fully decompiled with *Jad* (java de-compiler that also handles obfuscated class-files), which allows a manipulation of the source code. After the re-compilation, individual certificates can be generated with *keytool* and *jarsigner*, two Java-tools integrated in the Java SDK.

These analysis results allow a possibility for an exemplary attack, which combines these vulnerabilities to a Man-in-the-Middle attack (MitM) that can violate all five security aspects.

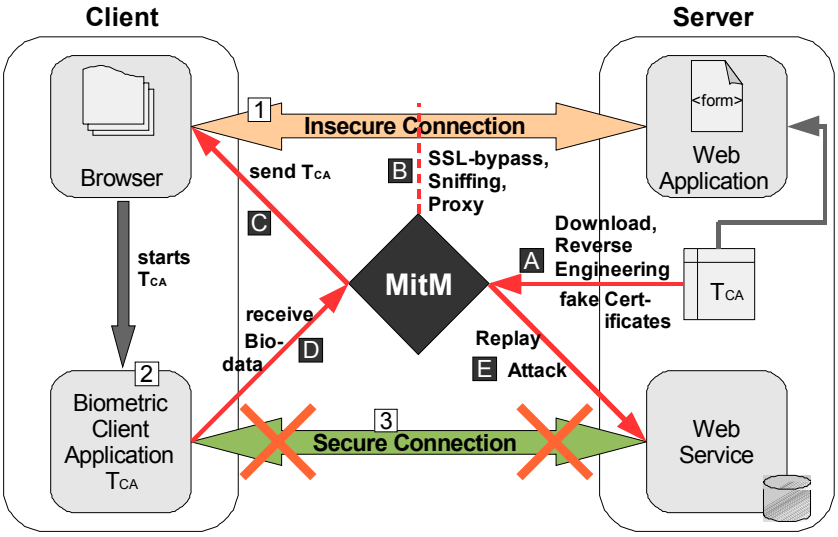

Fig. 6. Possible Man-in-the-Middle Attack scenario with modification of client application.

Figure 6 contains both, a standard sequence describing the normal data flow and a MitM sequence showing the possible attack. During the standard sequence (see numeration from 1 to 3) the browser on the client side requests the T_{CA} via a particular or even fully insecure connection (1) and executes it after the download (2). Afterwards the T_{CA} establishes a fully encrypted connection to the web service (3). A possible MitM attack could be executed similar to the literal numeration from A to E in Figure 6. The attacker can prior download and modify the T_{CA} by reverse-engineering as well by faked certificates (A), to violate integrity and authenticity of client application (see test 4 to 6 in Table 7). As given by test results 1 and 2 from Table 7, the attacker can simply sniff the unencrypted connection, interrupt the download request and redirect it by spoofing (B), affecting the availability of the server. According to this step the client machine should be infected with malware a-priori, in order to set up the hacker machine as proxy and to prevent SLL-connections (test 3 in Table 7) to the aimed biometric server. The redirection part is made simple because of the insecure connection, where plain text readings and manipulations are possible. To start the BioWebAuth client it is necessary to download an initialization file with information about the client's download location that can be manipulated within this file. If redirection achieved, the client loads the T_{CA} now from a different malicious server (C). After starting the modified T_{CA}, the behavior seems to be the same, but the biometric data will be sent to the MitM (D), who achieved his/her final goal now: bypassing the encrypted secure communication. Finally he/she is able to run a replay attack (E), compromising confidentiality and non-repudiation.

Originally, the BioWebAuth reference system was configured to use by default encrypted connections for downloading the T_{CA}. However, unencrypted connections were also allowed. The security analysis presented here has shown the vulnerability of allowing these unencrypted connections. Consequently, the use of unencrypted connections for

downloading the T_{CA} was disabled, in order to protect the BioWebAuth system against this kind of attack. As we see in the next subsection the Bio-Phishing opens another way to attack the system.

5.2 Bio-Phishing

The exemplary attack of Bio-Phishing is based on results for the target client application (T_{CA}) as listed in Table 7 (three last columns). Besides stealing of access data (username and password) Bio-Phishing is also aimed on identity in form of biometric samples. This technique mostly works with faked e-mails, which contain a trustworthy appearing but manipulated link to a faked website. A possible attack can proceed as described in Figure 7. At first, the attacker obtains the client application (A) as described in table 7 with test number 4. The test results 5 and 6 in Table 7 have shown that a reverse engineering is possible (B), e.g. by using *NetBeans* for the source code manipulation, in order to establish a direct connection to the hacker and to bypass the original connection. Furthermore, he sends (or bombs) spam mails toward the Internet containing the manipulated link to the Phishing-site, which has to be a perfect copy of the original one. If the victim gets such a faked e-mail (C), the manipulated link redirects him to the attacker's website (D) and he possibly gets spoofed by downloading a faked version of the T_{CA}. Via the established direct connection to the hacker's server the acquired biometric data are stolen and can be used for replay attacks (E), for example.

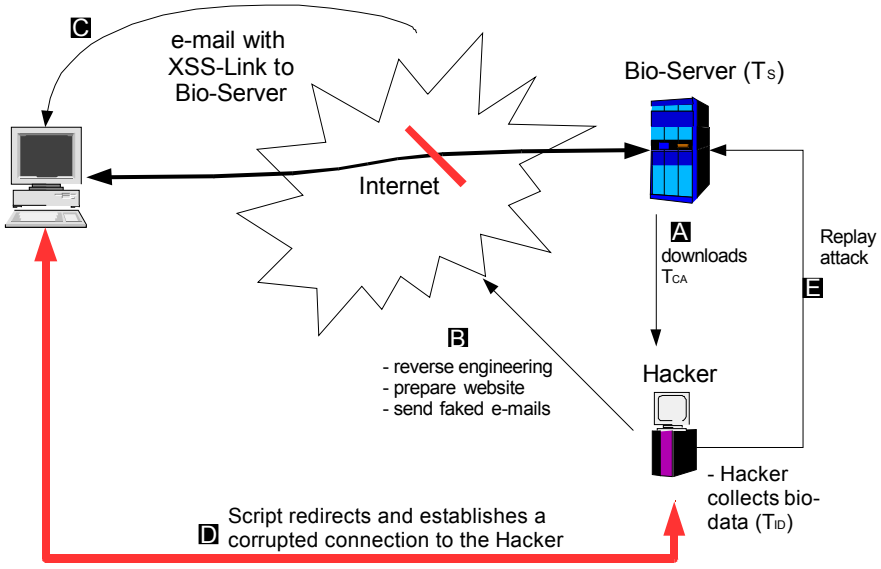

Fig. 7. Possible scenario for Bio-Phishing using the modified T_{CA}. Client receives an e-mail with manipulated URL.

The Phishing scenario provides a better cost-value-ratio than Man-in-the-Middle attack as it only requires the manipulation of T_{CA}, the creation of the faked web-site, and collecting a lot of biometric data. Otherwise, the Man-in-the-Middle scenario requires firstly that the attacker becomes the MitM and secondly theft of the data, person by person, which implies much more effort.

Both scenarios are mainly possible by the lack of security certificates signed from a trusted authority. In case of the BioWebAuth system, no trusted certificates are available yet, but only dummy certificates used for testing purposes.

6. CONCLUSIONS

This paper summarizes which security issues have to be taken into consideration with the integration of biometric authentication into web environment. For this purpose, a security checklist is developed, which is based on the OSI-model. Enhancements and modifications to this model by new layers and expansions based on CERT components allow setting up a checklist usable for a more detailed security analysis. Here we exemplary show how common and known tools can be applied for testing, where we selected overall 17 tools mostly from [1] to show exemplary our test results. The skill-level of an attacker is also considered based on the box model views. With the example of analyzing the BioWebAuth system, the vulnerabilities are evaluated by using the introduced checklist and two exemplary attacks (Man-in-the-Middle, Bio-Phishing) are further investigated by combining attack tools, which have resulted in a security weakness.

Following, some design recommendations structured by the break down of the system are given: Integrity and authenticity in client applications can be achieved by using official and trustworthy digital signatures and certificates issued only by a trusted certification authority. To protect confidential data against readout or manipulation by malware, data should be encrypted and even randomized during storage in memory. Moreover, saving data on local file system should be avoided completely. In order to protect biometric samples like images or sound samples against manipulation, they should be labeled by a hidden, fragile and encrypted watermark in order to ensure authenticity and non-repudiation of data sent to the server. Furthermore the communication's confidentiality also should be ensured by encryption methods, like the well known SSL or TLS protocols. The server itself must be protected by intrusion detection/prevention systems and should be always up to date in its security patches. The database configuration should be concentrated on user- and port-access.

As future work it would be interesting to check the open tests or to add more tests especially in the biometric layer. Furthermore, a software implementation of the checklist as a new security scanner tool may be possible. The modular architecture of the checklist, given by the layers, is an advantage for such a tool in order to investigate only specific parts, e.g. the biometric part in a stand-alone application or the network part for any kind of distributed applications.

7. ACKNOWLEDGEMENTS

The work described in this paper has been partially supported by the European Commission through the IST Programme under Contract IST-2002-507634 BIOSECURE, by Spanish MEC under the project PRESA TEC2005-07212, and Xunta de Galicia under the project PGIDIT05PXIC32202PM. The content of this publication is the sole responsibility of the University Magdeburg, University Vigo and can in no way be taken to reflect the views of the European Union.

8. REFERENCES

[1] Top 100 Network Security Tools, URL: http://sectools.org/index.html
[2] Otero-Muras, E., González-Agulla, E., Alba-Castro, J. L., García-Mateo, C. and Márquez-Flórez, O. W., "An Open Framework for Distributed Biometric Authentication in a Web Environment", Annals of Telecommunications. Special issue on multimodal biometrics, V. 62, N. ½, 177-192 (Jan/Feb 2007)
[3] Bishop, M., [Computer Security], Addison-Wesley, Boston, ISBN: 0-201-44099-7 (2003)
[4] OSI-Model, URL: http://www.sigcomm.org/standards/iso_stds/OSI_MODEL/ISO_IEC_7498-1.TXT
[5] Welcome to CERT, URL: http://www.cert.org/
[6] King, M., "EAGLES Evaluation of Natural Language Processing Systems: Final Report. EAGLES Document EAG-EWG-PR. 2", Center for Sprogteknologi, Copenhagen (1996)
[7] BioWebAuth: Biometrics for Web Authentication. Open source project, URL: http://sourceforge.net/projects/biowebauth
[8] JA-SIG (Java Architectures Special Interest Group) Central Authentication Service (CAS), URL: http://www.ja-sig.org/products/cas/
[9] ANSI X9.84-2003, Biometric information management and security for the financial services industry. *American National Standards Institute,* New-York (USA) (2003).
[10] BioAPI Consortium (ANSI/INCITS 358-2002), URL: http://www.bioapi.org
[11] Wright, J., "Detecting Wireless LAN MAC Address Spoofing", http://www.uninett.no/wlan/download/wlan-mac-spoof.pdf (2003).
[12] Whalen, S., "An Introduction to Arp Spoofing", http://www.node99.org/projects/arpspoof/arpspoof.pdf (2001).
[13] Mac MakeUp - MAC Address spoofing tool, URL: http://www.gorlani.com/publicprj/macmakeup/macmakeup.asp
[14] Sicherheit von Webanwendungen - heise Security, URL: http://www.heise.de/security/artikel/84149/
[15] SQL Injection Attacks by Example, URL: http://www.unixwiz.net/techtips/sql-injection.html
[16] Home Page of Jad - the fast Java decompiler, URL: http://www.kpdus.com/jad.html
[17] Java SE downloads, URL: http://java.sun.com/javaee/downloads/?intcmp=1282
[18] HxD - Freeware Hex Editor and Disk Editor, URL: http://mh-nexus.de/hxd/
[19] NetBeans IDE 6.0 Download, URL: http://download.netbeans.org/netbeans/6.0/final/
[20] PC Port Monitoring, Device Analyzing Software for Windows, URL: http://www.hhdsoftware.com/
[21] USB sniffer for Windows, URL: http://benoit.papillault.free.fr/usbsnoop/doc.php.en
[22] Hoppe, T. C., "Valuierung der Bedrohungssituation von Computerzecken", Diploma thesis, Otto-von-Guericke-University of Magdeburg, Department of Computer Science (2006). (in German)