

Vulnerability Analysis of Biometric Systems Using Attack Trees

Denis Speicher

Problem Report submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Computer Science

Bojan Cukic, Ph.D., Chair
Natalia Schmid, Ph.D.
Arun Ross, Ph.D.

Lane Department of Computer Science and Electrical Engineering
Morgantown, West Virginia
2006

Keywords: Biometrics, Attack Trees, Attack Modeling, Vulnerability,
Vulnerability Analysis, Security, Authentication.

Abstract

Vulnerability Analysis of Biometric Systems Using Attack Trees

Denis Speicher

Designing a secure authentication system is a challenging task, as many factors must be taken into consideration. Applications of biometric systems typically require robust system security, the compromise of which may have dire consequences. In 2002, Japanese cryptographer Tsutomu Matsumoto exploited a common vulnerability in 11 commercially available fingerprint biometric authentication systems. This suggests that a better method of vulnerability identification and analysis is needed. Frameworks provide a general view of biometric system structure, which may be used to simplify the task of vulnerability identification. Attack trees offer a means to simplify the task of vulnerability analysis. Combining the use of frameworks, attack trees, and risk assessment enables system engineers to more effectively identify and analyze vulnerabilities as well as determine the necessary countermeasures to eliminate them.

Contents

1. Introduction	1
1.1 Biometrics as a Security Enhancing Technology	3
1.2 Drawbacks: Vulnerabilities and Threats	6
1.3 Objective of the Report	8
1.4 Structure of the Report	9
2. Related Work	10
2.1 Attack Tree Methodology	10
2.1.1 Overview	10
2.1.2 Attacker Concerns.....	13
2.1.3 System Defense Concerns	18
2.1.4 Analysis	23
2.2 Biometric System Vulnerabilities and Attack Points.....	25
2.2.1 Ratha's Framework.....	25
2.2.2 Wayman's Framework.....	27
2.2.3 Bartlow and Cukic's Framework.....	28
2.3 Known Vulnerabilities	31
2.3.1 Forged Biometric	32
2.3.2 Replay Attack	33
2.4 Countermeasures for Known Vulnerabilities.....	33
2.4.1 Liveness Detection.....	34
2.4.2 Challenge Response.....	34
3. Application of Attack Trees to Biometric Systems.....	37
3.1 "Gummy" Finger Attack	37
3.1.1 Attack Tree Representation	39
3.1.2 Analysis	39
3.2 Replay Attack.....	40
3.2.1 Attack Tree Representation	42
3.2.2 Analysis	42
4. Summary and Future Work	44
4.1 Summary	44
4.2 Suggestions for Future Work	45

List of Figures

2.1 Simple Attack Tree Example	11
2.2 Risk Assessment Matrix.....	21
2.3 Cost of Attack	24
2.4 Ratha's Framework with Potential Attack Points	26
2.5 Wayman's Framework.....	28
2.6 Bartlow and Cukic's Framework.....	30
2.7 Ratha's Challenge Response Scheme	35
3.1 "Gummy" Finger Attack Tree Representation.....	39
3.2 Signal Replay Attack Tree Representation	42

List of Tables

- 2.1 Educational Complexity Levels14
- 2.2 Financial Cost Levels.....16
- 2.3 Temporal Cost Levels17
- 2.4 Consequence Severity19
- 2.5 Occurrence Likelihood.....20

Chapter 1

Introduction

Designing a secure authentication system can be quite a challenging task. Many factors must be taken into consideration during the design process. Historically, security breaches have often been the result of an exploited vulnerability, the likes of which system designers had either not considered plausible or simply not taken into consideration. If the existence of a vulnerability and likelihood of its exploitation had been apparent during system design one would expect that a countermeasure be in place. The absence of such countermeasures, hence the existence of such vulnerabilities, supports the previous statement. The problem of identifying vulnerabilities and successfully implementing countermeasures is a challenging one. It is apparent that a better method for analyzing vulnerabilities during the design process is needed so that system engineers will be less likely to overlook vulnerabilities which may later be exploited. This becomes especially important when dealing with an authentication system in which unauthorized access of system resources may have dire consequences.

Engineers in other disciplines have often relied on past failures to avoid design flaws. The collapse of a bridge, crash of an airplane, or sinking of a vessel, although tragic, enables future engineers to learn from past mistakes, thus improving future design. Generally, engineers can learn from past engineering failures because they have been publicized, usually along with the details necessary for such flaws to be easily avoided in the future. [15]

System engineers are typically reluctant to publicize data related to any system design flaw, vulnerabilities in particular. Furthermore, when a vulnerability of such a system is exploited and thus an attack has occurred, the owners and designers of the system fear that revealing details of the attack will provoke similar attacks, ruin their reputation, and affect customer confidence. [15]

In 2002 Japanese cryptographer Tsutomu Matsumoto exploited vulnerabilities in 11 commercially available fingerprint biometric authentication systems using a bit of ingenuity and \$10 worth of readily available household supplies. Matsumoto created artificial fingers from

gelatin bearing the fingerprints of authorized users and used them to fool the commercial systems [6]. The companies selling these products have long claimed that the systems were very secure with reliable liveness detection, a false claim that Matsumoto was able to exploit with little difficulty. So why did so many commercially available biometric systems possess the same vulnerability?

Is it because this particular vulnerability hadn't been considered? This is unlikely since most system vendors claimed that they implemented secure and reliable liveness detection. It can be assumed that the results of testing the liveness detection schemes yielded favorable results, but if this is true why did they fail during Matsumoto's experiments? Perhaps they only tested the liveness detection with artificial fingers made from silicon rubber or perhaps the method of submitting the artificial finger to the sensor was different than Matsumoto's method. While given the limited knowledge of system implementation details it may be unclear why the implemented countermeasures failed, it is clear that the method of vulnerability analysis used during the design process didn't catch the vulnerabilities.

Is it because of organizational reluctance of vulnerability disclosure? Despite this tendency there has been an increase in security related failure data in recent years. Rising public interest in the security of the Internet has fostered an increase in the publication of attack and vulnerability related data in journal articles, books, and security advisories [15]. This would suggest that organizational disclosure reluctance is not a primary contributor to system design flaws. However, anecdotal evidence suggests that system engineers are not learning from these documented attacks [15], [16]. While it is likely that there are many reasons system security is improving at a less than impressive rate, the security of a system is established during the design process so whatever the contributing factors are, this is where they are likely to occur. Moore and Ellison suggest that the lack of practical methods for improving system design, given historical attack data, is a primary contributor to the security trend [15].

While it is key that system engineers learn from past mistakes, this in itself is not enough to ensure the absence of vulnerabilities. With new technologies come new vulnerabilities. Even with current technology there lie vulnerabilities that attacker and system engineer alike have yet to discover. It is practically impossible to rule out the existence of system vulnerabilities using today's methods. However, given the right vulnerability analysis methodology and tools it becomes

easier for the system engineer identify and analyze potential points of attack and implement the appropriate countermeasures for each.

While the compromise of some systems such as an online gaming or chat server may not prove catastrophic, it's not hard to imagine a scenario where regional security may be threatened. Security is of the utmost importance when dealing with an authentication system in which an exploited vulnerability could result in an attacker gaining access to classified information or being allowed to enter a restricted area. Controlled access to system resources is the key function of an authentication system. Without this function the system is flawed. Therefore it is vital that extensive vulnerability analysis is performed during system design. Furthermore, the methods used should greatly simplify the designer's task of analyzing vulnerabilities and identifying possible means of exploitation. The designer can then implement appropriate countermeasures to negate the vulnerabilities.

A formal methodology for analyzing the security of systems and subsystems called attack trees was first introduced by Bruce Schneier in the December 1999 issue of Dr. Dobbs's Journal. According to Schneier, attack trees "represent attacks against a system in a tree structure, with the goal as the root node and different ways of achieving that goal as leaf nodes." This methodology helps the designer understand the different ways in which the system may be attacked as well as who the attackers may be, including their abilities, motivation, and goals. Understanding the attack and the attacker sheds light on the countermeasures necessary to thwart such attacks. [10]

1.1 Biometrics as a Security Enhancing Technology

Authentication can be defined as the process of verifying one's claimed identity. Many authentication systems in use today rely on either a token that a user possesses such as an ID card or the user's prior knowledge of something such as a password or personal identification number (PIN). Both have significant drawbacks.

Neither a token nor a password can ensure the integrity of the user's claim with a high level of certainty. A token can be stolen or forged by an imposter, who can then present the token to the authentication system and gain access with little or no trouble at all. The system cannot distinguish between a legitimate user's presentation of the token and that of an imposter. Similarly, a password can be obtained from a legitimate user with or without the user's cooperation. The user can

verbally disclose the password voluntarily or be forced to do so. The user may write the password on a small piece of paper and attach it to a monitor or the bottom of a keyboard so that it is not forgotten. If this is the case anyone passing by could potentially obtain the password and use it to claim the victim's identity. Passwords are often encrypted with a one-way hash function and stored in a file. If an imposter can obtain this file, they can perform a dictionary attack on it that will likely break any weak passwords in a short period of time, assuming they know the hash function. [25]

Another drawback of tokens and passwords is that they can be inconvenient for the users of authentication systems. In a token-based system the user must always carry the token in order to use the system. If the token is lost, the user must request a new token and the lost token must be invalidated in the system. Passwords can often be more of a burden than tokens. There is a security to convenience tradeoff in password-based systems. The more secure the password requirements the more inconvenient the passwords become. Lengthy passwords that include upper and lower case letters, numbers, and special characters are immune to easy guessing and dictionary attacks but are awkward to type and difficult to remember. Some systems require the user to change passwords periodically. Although this enhances the security of the password the user is likely to forget the password. To compensate for awkward or hard to remember passwords users often write down their passwords on paper and post them in a convenient location (*as described previously*), which may lead to the compromise of the password, and ultimately the security of the system.

Biometrics is a technology that aims to solve these problems, not with a token or password, but by using a physical characteristic of the user to validate his or her identity. Biometrics can be defined as identifying individuals based on a biological or behavioral characteristic [25]. Unlike a token or password, a biometric cannot be lost, forgotten, or changed. The user will always have possession of the biometric and in many cases authentication simply requires claiming an identity (*possibly typing a username*) and presenting the biometric to a sensor. Furthermore, a biometric system can differentiate between a legitimate user and imposter's identity claim with high confidence. Future applications of biometric systems may help eliminate identity theft, prevent credit card fraud, simplify border operations, and stop cellular bandwidth theft just to name a few [25].

Fingerprints are the oldest and most widely used form of biometric identification, dating back to just before the turn of the 20th century [13]. However, in recent years there has been an

increase in the research and use of other biometrics for identification including face, iris, retina, hand geometry, keystroke patterns, gait, and odor. Each of these has its strengths and limitations. Choosing the right biometric may depend largely on the application.

A biometric system is a pattern recognition system that establishes the authenticity of a presented biometric [19]. They are typically used to perform one of two main functions: to validate a claimed identity (verification) and to identify an individual out of a set of known identities (identification) [25]. In the context of this document we are primarily concerned with the verification function. Biometric systems operate in two modes: enrollment mode and identification mode.

The enrollment mode involves the presentation of a user's biometric to the system in order to train the system to recognize that individual. The user presents his or her biometric to the sensor, which creates a digital representation. The biometric data is then sent to the feature extractor where it is processed into a compact and expressive form called a template. If the quality of the capture fails to meet a predefined standard the user must present the biometric to system again. This may continue until a satisfactory capture is obtained. Once a template is created from a satisfactory capture it is stored in a database along with the templates of other users. It is not uncommon for multiple captures to be required, even if they all meet the quality standard. Some systems combine multiple captures to create a template although this may vary depending on the implementation and biometric being used. [19]

When operating in identification mode a user claims an identity and presents his or her biometric to the sensor. The sensor then creates a digitized representation of the biometric and sends the data to the feature extractor where a template is created in the same manner as it was in the enrollment mode. The template is then sent to a matcher, which retrieves the template of the claimed identity and compares it with the claimant's template. A matching algorithm determines the similarity of the two templates and produces a match score, typically between 0 and 1. If the match score is greater than a predefined threshold the templates are said to match and the claimant is successfully authenticated. Otherwise the templates are said not to match and the claimant is denied access to the system. [19]

1.2 Drawbacks: Vulnerabilities and Threats

Biometric systems solve many of the problems associated with traditional authentication systems. However, unless careful consideration is taken into the security of the system design it may be vulnerable to attack. Vulnerabilities and threats must be met with appropriate countermeasures to ensure the security of the system. It may be helpful to define some terms in order to minimize ambiguity, as each may have multiple definitions depending on the context. A vulnerability is a weakness of protection. An attack is the exploitation of a vulnerability, with undesirable effects on the system or its security. An attacker is an individual who attempts an attack on the system. A threat is a circumstance which makes an attack likely to occur, such as the exposure of a system vulnerability. Countermeasures are controls and preventative measures used to avert successful attacks. [23]

The existence of vulnerabilities in a biometric system is largely dependent on system design and structure, the type of biometric or biometrics used (i.e. fingerprint, iris, etc...), and managerial policies. Each of these areas encounters its own blend of vulnerabilities and must be analyzed in order to establish the appropriate countermeasures.

System design and structure refers to the hardware and software components of the system; component modules including their relative location, function, and communication protocols; and interaction with other systems or subsystems. The security of system hardware is vital. Most attacks on information systems happen remotely via network connections but if system hardware does not reside at a secure location with proper monitoring an attacker can simply damage or alter it. The nature of biometric systems requires that users interact with hardware, mainly the sensor. An unmonitored sensor may be viewed as a vulnerability, although other countermeasures (*besides monitoring*) may exist for dealing with sensor security.

All biometric systems can be broken down into component modules, each performing a vital function. These modules are common to all biometric systems regardless of which biometric or biometrics are used in identification. However, specific implementations differ in that some modules may be combined in function. For instance, the feature extractor and matcher modules are often combined and inseparable [1]. An attack which relies on these two modules communicating with each other over an open network would be impossible on this type of system and thus need not be considered. This brings up an important issue: modules may reside at different locations. Since

the modules must communicate with each other for the system to function properly, the communication protocol may raise important security concerns. Is the communication channel open? Are the data encrypted? How are data integrity ensured? If the sensor resides in a remote location, how does the system know that the received data came from a sensor capture of a living person's biometric? The answers to these questions may shed light on potential vulnerabilities.

Biometric data is acquired, processed, sent over a communication channel, stored, and retrieved as part of the normal operation of a biometric system. The integrity and confidentiality of the data must be maintained at every phase of operation. At the time of acquisition, it must be ensured that the biometric data is obtained from a living person and not from an artificially manufactured biometric or the mimic of an imposter. The processing of data is secure if the system software has been thoroughly tested resulting in correct operation and it cannot be maliciously altered. When the data is traveling over a communication channel it must not be viewed except by the intended destination module. Also, the system must ensure that received data was sent by a legitimate sending module and not by an attacker. Furthermore, it must be ensured that the data has not been altered in any way during transmission. Biometric templates are stored in a database. The database as well as the system on which it resides must be secure against attacks. The unauthorized insertion, alteration, retrieval, or deletion of biometric templates would greatly compromise the security of the system. Strong countermeasures should be taken to prevent this from happening.

If the biometric system interacts with other systems or subsystems, the security of each will affect the other. Vulnerabilities in the other systems could potentially compromise the security of the biometric system. Any such vulnerability should be met with a countermeasure to ensure that an attacker does not use the system in question as a vehicle to perform an attack on the biometric system.

The type of biometric used in the authentication system may carry with it a set of vulnerabilities unique to that biometric. For instance, some biometrics are less unique than others. Voice is on the low end of the uniqueness spectrum. It is possible for an imposter to mimic the voice of a legitimate user and be accepted by the system. Similarly, an attacker may play a recording of a user's voice to the sensor and fool the system into accepting the recording as the voice of the legitimate user. Voice is not the only biometric with its own set of vulnerabilities. Consider face recognition. While it may not be possible for an attacker to mimic a face like they would a voice, they may hold up a photograph of a legitimate user's face to the sensor, which may

be good enough to fool the system. Regardless of the biometric type, the unique set of vulnerabilities that come along with it must be taken into consideration.

Another area in which the existence of vulnerabilities may depend is system managerial policies. This may include policies regarding the enrollment process, system upgrades, supervision, employee background checks, employee and user training, and site security among others. It's not hard to imagine a scenario in which a poorly written policy in any one of these areas may introduce a vulnerability to the system. Consider a weak enrollment process where the user's only requirement to enroll is knowledge of his or her social security number. An imposter could easily obtain the information and enroll into the system with no problem. Similarly, if policy does not require background checks when hiring employees, an individual with malicious intent may be hired by the organization. He could then enroll himself in the system and wreak havoc. Looking at these examples it is easy to see how important managerial policies are to the security of a biometric system.

Considering the vast number of potential vulnerability areas, the importance of analyzing each becomes apparent. Having the best security in one of these areas is useless if security is lacking in another. For instance, secure communication protocols cannot stop the compromise of biometric data if an attacker can simply hack into the database and steal it. Likewise, strong managerial policies cannot protect the integrity of biometric data if an attacker can perform a replay attack and send a fraudulent signal to the feature extractor module. System security begins with system design. It is a difficult task to identify all potential vulnerabilities. Furthermore, identifying all possible means of exploitation for each is even more difficult. As for the attacker, motivation and abilities are both factors that determine the likeliness of a successful attack. This may factor into weighing the benefit of implementing a particular countermeasure versus the risk of not implementing one.

1.3 Objective of the Report

The objective of this report is to address the difficulty of vulnerability analysis in biometric systems and suggest using attack trees as a tool to simplify this task. By incorporating the attack tree methodology into the system design process, vulnerabilities may be analyzed more easily and the probability of exploitation can be reduced by implementing countermeasures.

1.4 Structure of the Report

This document is divided into four main sections. The introduction section presents the problem at hand, which is the difficulty of designing a secure system. The importance of security in an authentication system is addressed and the advantage of biometric authentication over traditional authentication systems is presented. Next, potential areas of vulnerability and threat are explored and the objective of using the attack tree methodology to simplify the task of vulnerability analysis is stated.

The section on related work describes work relevant to the attack tree methodology and biometric system security. First, attack tree methodology is described in detail including an overview, attacker concerns, system defense concerns, and a section on the analysis of attack trees. Next, three frameworks describing biometric system structure and potential attack points are explored. Three known vulnerabilities are then presented followed by the description of a countermeasure for each.

The section on the application of attack trees to biometric systems is the core of this report. Attack trees are applied to two different attack scenarios. Each scenario is described in words followed by an attack tree representation, an analysis of the attack tree, and a summary. It is the analysis of the attack tree from which means of vulnerability exploitation are explored and countermeasures developed to thwart the attack.

The final section summarizes the report, presents the conclusions drawn from the research, and provides suggestions for future work on the subject.

Chapter 2

Related Work

2.1 Attack Tree Methodology

Attack trees are a formal methodology that aims to simplify the task of vulnerability analysis. They allow the designer to understand the different ways in which a system can be attacked, thus making it easier to develop countermeasures to thwart those attacks [10]. They can also tell us a great deal about the factors that make the occurrence of an attack likely or even possible. We can include attacker attributes in the tree description such as education level or financial abilities to help determine the probability of an attack. Attack trees enable an attack to be viewed graphically, as opposed to trying to digest the complexity of an attack mentally. The latter may be much more difficult considering an attack may be carried out in many different ways. Furthermore, attack trees are reusable. Once a tree with a particular goal is completed it can be reused any time that goal needs to be analyzed. Also, they facilitate collaboration quite well, making it possible for many individuals to work on attack trees simultaneously, thus saving time.

2.1.1 Overview

An attack tree is the representation of an attack in a tree structure. The root node is the goal of the attack and the leaf nodes are ways of achieving that goal [10]. Any node that does not have a child is a leaf node. Leaf nodes describe events that must happen in order for the attacker to achieve the goal, either in conjunction or as alternatives. Non-leaf nodes can either be OR nodes or AND nodes. Each non-leaf node is a sub-goal of the main goal represented by the root. An OR sub-goal is satisfied if any of its children are satisfied. An AND sub-goal is satisfied only if all of its children are satisfied. A leaf node is satisfied if the event it describes is possible. An attack which is possible can be made impossible by making one or more leaf nodes impossible. This can be

achieved by implementing one or more countermeasures. If the root is an AND node the attack can be thwarted by countering at least one sub-goal. On the other hand if the root is an OR node all sub-goals must be countered for the attack to become impossible.

Consider the simple example in Figure 2.1 representing an attack in which the goal is to open a safe (*taken from [10]*). Leaf nodes possess a Boolean attribute that indicates whether or not the event is possible. The attribute values of the sub-goals are derived from the logical AND/OR of its children’s attribute values. Four alternatives are presented as a means of opening the safe. Since

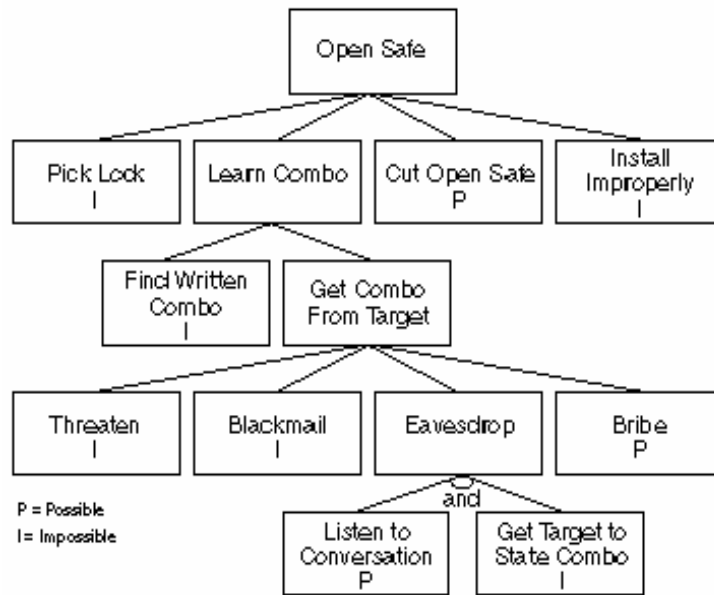


Figure 2.1: Simple Attack Tree Example [10]

the root is an OR node, accomplishing just one of these alternatives is sufficient to accomplish the attack. The safe can be opened by picking the lock, learning the safe combination, cutting open the safe, or installing the safe improperly. The safe combination can be learned by either finding a piece of paper with the combination written on it or obtaining the combination from the target (*an individual with the knowledge of the safe combination*). Obtaining the combination from the target can be accomplished in one of four ways. The attacker can threaten the target, blackmail the target, eavesdrop on the target, or bribe the target. In order to eavesdrop on the target two events must happen: the attacker must listen to a conversation of the target and the target must state the combination to the safe. Notice the notation that distinguishes an AND node from an OR node. An AND node is denoted by the curved line that connects the lines drawn from the node to its children

under which the word “and” is written (*the word “and” is optional and will not be used in the future examples of this work*). Any non-leaf node without this notation is an OR node.

A simple analysis of this attack tree reveals that two means of attack are possible. The attack may be carried out by either learning the combination or cutting open the safe. The other alternatives, picking the lock and installing the safe improperly, are impossible according to this attack tree. Cutting open the safe is fairly straightforward because it is a leaf node and its Boolean attribute is deemed possible. However, learning the combination is a sub-goal and the Boolean attribute associated with it must be calculated using the attributes of its child nodes. Since it is an OR node only one of its children must be possible for the sub-goal to be satisfied. Finding the written combination will not satisfy the sub-goal since it is deemed impossible. Getting the combination from the target is the only alternative. Only one “possible” attribute value of its children is needed being that it is an OR node. Threatening and blackmailing the target are not possible since they are leaf nodes with attribute values of “impossible.” Eavesdropping on a conversation of the target isn’t possible since it is an AND node and only one of its children has an attribute value of “possible”. However, the leaf node labeled “Bribe” will satisfy the “Get Combo from Target” sub-goal, which will satisfy the “Learn Combo” sub-goal, which satisfies the main goal of the attack tree.

Deciding whether an event is possible or impossible is left solely to the analyst. In the attack tree of Figure 2.1 the leaf node labeled “Find Written Combo” is considered impossible. While it is entirely possible for the safe combination to be written on a piece of paper and an attacker may stumble upon that piece of paper and learn the combination, the likeliness of this event is so low that the author of this attack tree has decided that a value of “impossible” was appropriate. Remember, in the case of this example the node attributes are Boolean, although other attribute types are possible.

In this example three of the root’s children are leaf nodes. Each of them may be further elaborated by converting it into a sub-goal and either determining the steps necessary to achieve each, in which case the node would become an AND node and the necessary steps would become its children, or determining alternative ways of achieving the sub-goal, in which case the node would become an OR node and the alternatives would become its children. The level of abstraction in which the nodes are described is ultimately up to the analyst. For instance, picking a lock requires a series of events to happen in order for it to be successful. First the attacker must learn

how to pick a lock, which in itself can become a sub-goal with a set of child nodes. For the sake of simplicity, we'll assume it's a leaf node. Then, the lock picking tools must be obtained. Next, the attacker must gain physical access to the locking mechanism. Again, this may be further elaborated but this is just an illustrative example and the completeness of this tree is not the issue. Finally, the attacker must physically pick the lock. As you can see the level of abstraction in the attack affects how easy it is to analyze the tree. It also makes it easier to develop an effective countermeasure to thwart the sub-goal.

It is unclear in the original example why the "Pick Lock" node is considered impossible, but as the level of abstraction lessens it becomes clearer what needs to be done in order to thwart this sub-goal. Stopping the attacker from learning how to pick a lock is not a viable option. It would be impossible to stop every potential attacker from learning how to pick a lock. This is simply out of the system designer's control. Likewise, there is no possible way to stop someone from obtaining the tools necessary to pick a lock. However, the attacker can be prevented from gaining physical access to the locking mechanism, and the lock cannot be picked otherwise. It becomes quite clear that buying a robust safe, the locking mechanism of which is behind half an inch of steel, makes it impossible for the attacker to gain access to it. Now it is clear why the sub-goal is impossible and what countermeasure will prevent it.

2.1.2 Attacker Concerns

In addition to offering an understanding of ways in which a system may be attacked, attack trees provide information on the individual performing the attack. There are certain requirements an attacker must meet in order to be capable of performing attacks. For instance, an analysis of the attack tree of Figure 2.1 reveals that two means of attack may result in the goal of opening the safe. One is to learn the safe combination by bribing someone who knows it and the other is cutting open the safe. Each alternative requires the attacker to possess a certain set of abilities and assets. Bribery requires only that the attacker possess a certain amount of money. Cutting open the safe, on the other hand, requires special equipment and the knowledge of how to use it. Like bribery, obtaining special equipment requires a certain amount of money. Attacker concerns can be thought of as the assets and abilities necessary to perform the attack. Each concern may be placed in one of three categories: Educational Complexity, Financial Cost, and Temporal Cost.

2.1.2.1 Educational Complexity

The education level of the attacker will determine his or her capabilities when carrying out the attack. Certain tasks require specific knowledge. An educational complexity categorization scheme is presented in [11] and shown in Table 2.1. Each level is identified by a number from 1-13, 1 being the lowest educational complexity level and 13 the highest. The qualifications for each level are presented and an alias is associated with levels 5-13. It is important to note that the academic degree references are meant only as a reference. It is not required that an attacker have a certain degree to be associated with that complexity level, often they will not.

Level 1 is the lowest and most basic level. An individual with this level of educational complexity has no computing education whatsoever. A level 2 attacker is familiar with the term “computer” and may be aware of what they are used for, but has never used a computer. At level 3 an attacker knows how to use a computer. An individual at level 4 can use a computer and has a basic knowledge of operating systems and the Internet. Level 5 includes all knowledge and skills

Educational Complexity	Qualifications	Alias
1	None - No computing education	
2	Primary education – Familiar with term “computer”	
3	Secondary education – Computer user	
4	GCSE – User, Basic knowledge of operating systems (OS) & Internet	
5	A level – User, Basic knowledge of OS & Internet, Basic programming skills	Script Kiddie
6	University Level 1 – Power user, Knowledge of OS	Amateur
7	University Level 2 – Power user, Medium knowledge of OS, Internet, programming and networking, Familiar with Linux	Amateur
8	University Level 3 / BSc – Administrator, Advanced internet, programming, networking & OS, Basic scripting, Basic Linux	Amateur
9	MSc – Root, Expert internet, programming, networking, scripting, Medium Linux	Hacker
10	Post MSc – Root, Expert internet, programming, networking, scripting, Advanced Linux, active hacking	Expert Hacker
11 (A)	PhD – all the above, known entity to hacking Community	Professional Hacker
12 (B)	Post PhD – all the above, criminal record	Computer Criminal
13 (C)		Supreme Being

Table 2.1: Educational Complexity Levels [11]

of levels 1-4 plus basic programming skills. An attacker at this level is sometimes referred to as a “Script Kiddie”. The 6th level or University Level 1 is a power user – someone with level 1-5

knowledge plus additional knowledge of the operating system. Level 7 (*University Level 2*) attackers have networking skills and are familiar with Linux. A Level 8 attacker (*University Level 3*) is comparable to someone with a Bachelor of Science degree. An individual at this level could be a system administrator with advanced networking knowledge, basic scripting, and basic Linux knowledge. Attackers at levels 6-8 may be referred to as “amateurs.” Level 9 is comparable to someone with a Master of Science degree. This level includes expert internet, programming, networking, and scripting skills as well as medium Linux knowledge. A level 9 attacker is sometimes referred to as a “hacker.” Level 10 attackers, or “expert hackers”, are active in hacking and have advanced Linux knowledge. At level 11 an attacker is a known entity in the hacking community, a professional hacker. Their knowledge and skills are comparable to that of someone with a Ph.D. Level 12 attackers (*computer criminals*) have all previous levels of knowledge plus a criminal record. Level 13 is the highest educational complexity level, known as the “supreme being.” There’s no telling what someone of this status is capable of.

2.1.2.2 Financial Cost

An attacker will almost certainly have to spend some money in order to obtain the tools and services necessary to carry out the attack. The cost may be incurred during or prior to the planning of the attack. For instance, any remote attack requires a computer, modem, and access to the Internet. The attacker likely had these possessions prior to the planning of the attack but the financial cost remains the same. Any tools or services that aid the attack can be figured into the financial cost since the attacker incurred a charge in obtaining them. Also, certain levels of financial cost may require the collaboration of a pair or group of individuals. A financial cost categorization scheme (*presented in [23]*) is illustrated in the table of Table 2.2. The degree of cost is divided into 10 levels (*0-9*) and ranges from no cost to billions of dollars. The collaboration required for funding can be an individual person, a pair of individuals, a group, a country, or multiple countries. Each level represents a degree of magnitude, or roughly 10 times, more than the previous level. For instance, level 3 represents thousands of dollars, which is roughly ten times more than the hundreds of dollars represented in level 2.

Cost Level	Financial Costs of Attack (\$)	Collaboration For Funding
0	0	Individual
1	10's	Individual
2	100's	Individual
3	1,000's	Individual
4	10,000's	Individual / Pair
5	100,000's	Individual / Pair
6	1,000,000's	Pair / Group
7	10,000,000's	Group / Country
8	100,000,000's	Group / Country
9	1,000,000,000's	Country / Multiple Countries

Table 2.2: Financial Cost Levels [23]

The cost level associated with a particular attack dictates what an attacker is capable of. Just because the attacker is not financially capable of a particular means of exploitation doesn't mean that he or she cannot perform the attack. A vulnerability may be exploited several different ways (*indicated the root being an OR node*), each having a cost level associated with it. If an attacker cannot produce the funds for a particular exploit at cost level 5, there may be another exploit for the same vulnerability at cost level 4. Furthermore, the attacker will be likely to choose the exploit with the lowest cost level, assuming he or she meets the educational complexity requirement.

2.1.2.3 Temporal Cost

The temporal cost refers to the amount of time it will take the attacker to exploit the vulnerability. Like the financial cost, the temporal cost may affect the attacker's ability to perform an exploit as well as which means of exploitation are used for a particular vulnerability.

Consider an attack scenario where a third party (*attacker*) attempts to read the contents of a file send over a network using an SFTP protocol. Let's suppose that the protocol uses the AES encryption algorithm with a 256 bit key length. One possible approach is brute force (*attempting to decrypt with every possible key until the correct decryption is reached*). Let's assume that it takes 250 milliseconds to decrypt the file with a particular key and check to see if it has been correctly decrypted. 2^{256} possible keys exist, so trying them all would take 2^{254} seconds, assuming no other

latencies are encountered. On average the attacker will reach the correct key after trying about half of the key combinations. Taking that into consideration, it would take the attacker 2^{253} seconds to find the correct key, on average. That’s approximately 4.59×10^{68} years! Even if the attacker had 1,000,000 machines available to run the decryption in parallel it would still take 4.59×10^{62} years to find the correct key, on average. The number of available machines is irreverent because it will take many lifetimes to decrypt the message using brute force, on average. The attacker simply does not have the time for this means of exploitation. It would be much easier for the attacker to hack into the system on either end of the communication and read the file in an unencrypted form. If the files are stored in encrypted form, the attacker is more likely to obtain the root password than to break the encryption. This may be accomplished by exploiting a buffer overflow, installing a key logger, or simply bribing someone. Whatever the method, the temporal cost would be far less than that of brute force and therefore within the temporal means of the attacker. Whether or not it would be within the attacker’s financial means is another question.

A temporal cost categorization scheme is presented in Table 2.3. It is divided into 9 levels (0-8). These temporal cost levels can be associated with the individual events (*leaf nodes*) of the attack. Adding up the temporal costs of the individual steps of a means of exploitation will result in the total temporal cost for that particular means.

Cost Level	Temporal Cost of Attack
0	Less than 1 second
1	Seconds
2	Minutes
3	Hours
4	Days
5	Weeks
6	Years
7	Decades
8	Lifetimes

Table 2.3: Temporal Cost Levels

2.1.3 System Defense Concerns

System defense concerns are factors that should be taken into consideration while developing countermeasures to thwart attacks. Choosing whether or not to implement a countermeasure is not as straightforward as you may think. The obvious mentality may be to implement a countermeasure for every possible vulnerability exploit you can think of. While this seems logical enough, several factors should be taken into consideration. Among these are financial cost, risk, and image and customer confidence.

2.1.3.1 Financial Cost

Each countermeasure comes with a financial cost of implementation. Similarly, every attack brings with it a financial cost to the organization; although this may not be the only type of cost incurred. Possible outcomes may include injury, loss of human life, reputation damage, pain and suffering, loss of time, and loss of productivity among others. Some of these losses do not carry a price tag while others carry a fairly large one. For instance, is it worth risking human life to save a few dollars? The answer to this question may seem trivial, but consider this: what if the chance of this happening is practically impossible? Would it then be worth a considerable fiscal investment? These examples may seem a bit extreme but they are not impossible, especially when you consider some of the high security applications biometric systems may be used for.

2.1.3.2 Risk

So how can one determine whether or not a countermeasure cost effective? Remember, we are not simply talking about money. There may be much more at stake. One way is to assess the risk of exploitation. Risk is defined as the chance of something going wrong – the danger that injury, damage, or loss will occur (*Encarta Dictionary*). Now that we know what risk is, how can it be assessed? Furthermore, how do we represent it?

Risk assessment can be broken down into three basic steps: determine the consequence severity, determine the occurrence likelihood, and assess the risk based on these two factors [22]. Assuming the two factors are determined accurately, this method of risk assessment can help in determining whether or not to implement countermeasures to thwart specific attacks.

Category	Description
I	Death, loss of critical proprietary information, system disruption, or severe environmental damage
II	Severe injury, loss of proprietary information, severe occupational illness, or major system or environmental damage
III	Minor injury, minor occupational illness, or minor system or environmental damage
IV	Less than minor injury, occupational illness, or less than minor system or environmental damage

Table 2.4: Consequence Severity [22]

A scheme for categorizing consequence severity is presented in [22] and illustrated above in Table 2.4. Severity is divided into four categories (*identified by roman numerals*), I being the highest severity and IV being the lowest. A category I consequence may result in death, loss of proprietary information, system disruption, or severe environmental damage [22]. Preventing a consequence of this severity should receive the highest priority unless it is very unlikely to occur, although it should still be addressed. A category II consequence may result in severe injury, loss of proprietary information, severe occupational illness, or major system or environmental damage [22]. Although not as harsh as a category I, prevention should take a high priority, as a consequence may be quite damaging to the operation and morale of an organization. A category III consequence may result in minor injury, minor occupational illness, or minor system or environmental damage [22]. While not terribly damaging to the operation of an organization, prevention still should be addressed. The priority of a category III consequence may be determined by likelihood of occurrence. A category IV consequence has the least severity and may result in less than minor injury, occupational illness, or less than minor system or environmental damage [22]. Prevention may or may not be necessary based on likeliness of occurrence and the discretion of the analyst.

Likelihood of occurrence is also categorized in [22], as illustrated below in Table 2.5. Occurrence likelihood is divided into 5 categories labeled A-E, A being the most likely to occur and E the least. The descriptions in Table 2.5 are meant as a reference. It is up to the analyst to determine the likeliness of an event.

Category	Description
A	Frequent - Possibility of repeated incidents
B	Probable - Possibility of isolated incidents
C	Occasional - Possibility of occurring sometime
D	Remote - Not likely to occur
E	Improbable - Practically impossible

Table 2.5: Occurrence Likelihood [22]

The likeliness of occurrence will depend on several factors and may be somewhat subjective. Among these are educational complexity, financial cost, temporal cost, motivation, and the attacker's risk and gain. Each will affect the likeliness of occurrence and some will affect others. For instance, the higher the educational complexity of an attack the less likely it will occur. Financial and temporal costs affect the likeliness in a similar way, although temporal cost is unique in that if it is high enough, say lifetimes, no one would have enough time to complete the attack. Motivation is the attacker's reasoning for performing the attack. In some cases this may be easy to determine, but in others there may be a plethora of motivations. Many are motivated by financial gain. Some are motivated by achieving a status and recognition. Others gain neither money nor status and are motivated by revenge or anger towards an organization. Gain may fall under the category of motivation, but risk (*in terms of the attacker*) is different altogether. The risk an attacker takes in performing an attack refers to the consequences he or she will face if caught. These may include fines, jail time, or in extreme cases (*such as treason*) death. The risk one is willing to take is directly related to the motivation and the likelihood of getting caught. No one in their right mind would risk heavy fines and jail time for a few hundred dollars, unless the likeliness of getting caught is low. On the other hand, if the likeliness of getting caught is medium but the gains are substantial it may be worth the risk.

These factors are primarily attacker concerns. After all, the attacker is the one who benefits from the success of an attack. However, I can think of at least one factor which is not an attribute of the attacker: public knowledge of the vulnerability. This factor is typically controlled by the research and development community and the media. For example, attacks often occur to systems

immediately following the vendor’s disclosure of vulnerability and the release of a patch to fix it. Systems, especially those of large scale, are not patched instantaneously. It takes time for the system administrator to learn of the vulnerability and download and install the patch. Depending on the system administrator’s competency level, the system may not be patched for quite some time. In that time attacks are more likely to occur.

It is important to note that I am in no way implying that the disclosure of vulnerabilities is altogether a bad thing. Sure, to an organization with an un-patched system it creates a threat, but pulling the wool over everyone’s eyes will not make the vulnerability go away. Consequently it is often publicized research on vulnerability exploits that offers countermeasures to thwart them.

Once the consequence severity and the occurrence likelihood have been properly categorized the final step in the risk assessment may commence. The risk assessment matrix illustrated in Figure 2.2 offers a means to categorize a risk based on the consequence severity and occurrence likelihood levels obtained in the first two steps of the methodology. Using the matrix, risk is categorized into one of four levels based on the results of steps 1 and 2. The original context of this matrix assumes that the system is in operation but it may be used in a design context just as well.

Severity level	Probability of occurrence				
	(A) Frequent	(B) Probable	(C) Occasional	(D) Remote	(E) Improbable
I (High)	1	1	1	2	3
II	1	1	2	2	3
III	1	2	2	3	3
IV (Low)	3	3	4	4	4

- = Risk 1 (undesirable and requires immediate corrective action)
- = Risk 2 (undesirable and requires corrective action, but some management discretion allowed)
- = Risk 3 (acceptable with review by management)
- = Risk 4 (acceptable without review by management)

Figure 2.2: Risk Assessment Matrix [22]

Levels 1 and 2 represent high risks. It would be extremely undesirable not to have an effective countermeasure to mitigate a risk of this magnitude. Level 3 risks are not as serious as that

of levels 1 or 2, only arising when the likeliness of occurrence is remote or improbable or when the consequence severity is very low. Cost may often be the deciding factor when considering whether or not to implement a countermeasure for this type of risk. Finally, level 4 represents the lowest risk level. Cost will likely be even more of an issue here since this type of risk only occurs for the lowest consequence severity at a likeliness of occasional, remote or improbable.

This methodology presents a clear and concise approach to risk assessment. Although only four categories are presented for the consequence severity, likelihood of occurrence, and risk levels; more may be added at the discretion of the analyst. It may be desirable to represent risk more precisely in six or eight levels, which would likely require the number of categories for steps 1 and 2 to increase as well. Furthermore, the analyst may want to set a threshold for risk levels that justify a countermeasure. A cost threshold may also be desirable for lower risk levels. The details of the methodology may (*and probably should*) be tailored to system, but the approach remains the same.

2.1.3.3 Image and Customer Confidence

The exploitation of a vulnerability may cause an organization to experience some type of loss, as described in Table 2.4. However, there may be even more at stake, which is not initially apparent. An organization's loss may continue well after the occurrence of the attack. Consider a system in which access to sensitive personal information is controlled via a biometric authentication system. The system is utilized by an organization which relies on customer loyalty to stay in business, as competition is fierce. Suppose the system experiences an attack in which sensitive customer information is compromised as a result. When word of the attack gets out customers will lose confidence in the organization's competence. The organization will lose customers to its competition and likely see a decrease in acquiring new ones. The attack will affect the image of the organization for a period of time. From the time the attack is disclosed to the time the image is recovered, the organization is likely to experience a significant loss in capital. This loss is in addition to any financial loss that initially occurred at the time of the attack. This example brings to light the fact that the consequences of an attack may be long term. This should be taken into consideration when assessing risks.

2.1.4 Analysis

The construction of attack trees alone is not enough to determine whether or not certain security measures are necessary. Although attack tree construction offers an understanding of specific means of exploitation and the countermeasures to stop them, it is the analysis in terms of risk and cost that will determine if they should be implemented. Once the attack tree representation of a particular threat is constructed, the attributes associated with the nodes in that tree as well as the attributes associated with any countermeasures can be quantified. The quantification may be in terms of the attacker or system defense, both of which are needed.

Quantification is the combining of leaf node attribute values for a particular means of exploitation, resulting in an attribute value for the specific attack. Consider the example attack tree in Figure 2.1 where the goal of the attack is to open a safe. Each leaf node has an associated Boolean value representing whether or not the event described by that node is possible. Leaf nodes may have multiple attribute values of varying types. In addition to type Boolean, attributes can be numerical or user-defined. Some possible user-defined attribute types are educational complexity level, financial cost level, temporal cost level, consequence severity, likelihood of occurrence, and risk level. Figure 2.3 illustrates the quantification of a simple numerical attribute representing the financial cost of performing the attack originally described in Figure 2.1.

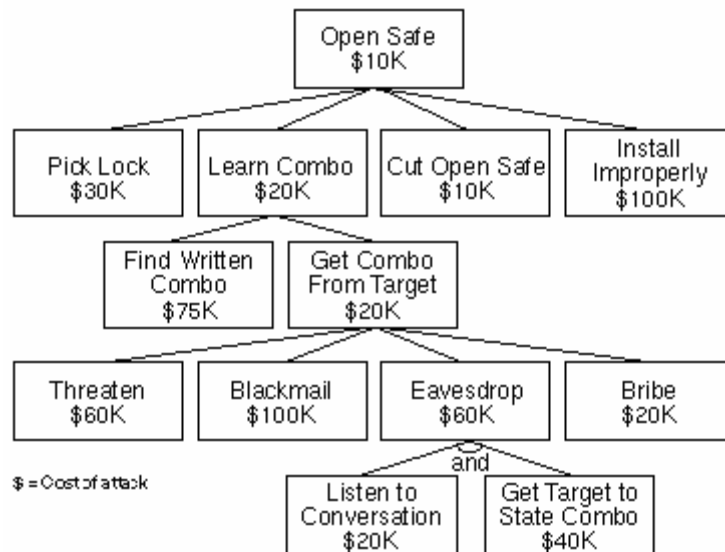


Figure 2.3: Cost of Attack [10]

When analyzing a numerical attribute like cost, we can learn a great deal about the likeliness of certain means of exploitation. For instance, looking at the four alternative ways of achieving the root goal it is evident that cutting open the safe is the least expensive. If an attacker only has \$10,000 to spend on the attack this is the only option because the attacker isn't financially capable of choosing another means. Alternatively, suppose the attacker has \$100,000 to spend on the attack but the contents of the safe are only worth \$30,000. All other factors aside, it wouldn't be beneficial for the attacker to choose a means costing \$30,000 or above. Furthermore, it would be more likely for the attacker to favor the \$10,000 attack over the one costing twice as much. While other factors must be taken into consideration when determining the likeliness of a particular attack, the cost certainly helps us understand the attacker's motive when choosing one.

When quantifying a numerical value it is important to understand what is desired. For instance, in many cases it is desirable to know what the least expensive attack would be. Since "Pick Lock", "Cut Open Safe", and "Install Improperly" are leaf nodes we already know their associated attribute values. However, the node labeled "Learn Combo" must be quantified to obtain an attribute value. It is an OR node therefore the child with the least expensive attribute becomes its value. The left child has a value of \$75,000 but the right child is an OR node itself and will need quantified. Three of its children have values of \$60,000, \$100,000, and \$20,000 but the fourth is an AND node and must be quantified. Remember that all events described by the child nodes of an AND node must happen in order to satisfy the sub-goal. Knowing this, it becomes apparent that all child node attribute values must be summed to get the value of the parent node. The fourth node has two children with values of \$20,000 and \$40,000 giving the parent node a value of \$60,000. Now we can quantify the "Get Combo From Target" node at \$20,000, the value of the child with the least expensive attribute. Now we can finally quantify the "Learn Combo" node at \$20,000 since this value is less than that of its other child value of \$75,000. Now that we have values for all the root's children we see that "Cut Open Safe" is the cheapest at \$10,000 and consequently becomes the value of our attack. From here on we will assume to be looking for the least expensive attack when dealing with financial cost.

The example in Figure 2.3 is a simple one meant for illustration. In practice, there would be several attributes associated with each leaf node. We can look at the attributes as being either attack attributes or defense attributes, as described in sections 2.1.2 and 2.1.3. Attack attributes provide an understanding of the attacker and help determine the likeliness of a particular means of

exploitation. These include attributes such as educational complexity level, financial cost level, and temporal cost level. On the other hand, defense attributes deal with the risk associated with the attack and help determine whether or not a countermeasure should be implemented to thwart a particular means of exploitation. These include attributes such as consequence severity, likelihood of occurrence, and risk level (*the latter is computed from the former two*).

Analysis of attack trees is a complex process that requires good judgment and may be somewhat subjective. It will take time for a security analyst or system designer to become proficient in the methodology. Simply knowing the theory is not equivalent to being a good attack tree analyst. Practice and hands on experience are the best ways to master the subject. Furthermore, collaboration can assist in the goal of completeness and timely development by dividing the workload and fostering the mesh of ideas which may fill in the gaps that may otherwise be present had non-collaborative development been used.

2.2 Biometric System Vulnerabilities and Attack Points

All biometric systems are similar in they all possess certain component modules necessary to their operation. Though implementations vary in how they function and communicate with each other the general framework remains the same. Because they all possess the same component modules which rely on communication with each other, we can use the framework to identify possible attack points. Narrowing the possible points of attack helps the analyst in identifying the means of exploitation. A biometric system may be attacked at points of data storage, processing, communication channels, or outside the system completely (*as in social engineering*). The frameworks described in the next three sections offer a means of describing the structure of biometric systems and identifying potential points of attack.

2.2.1 Ratha's Framework

According to Ratha et. al., a biometric systems can be cast in the framework of a pattern recognition system [1]. At the initial stage of operation the biometric signal is acquired from the user via a biometric sensor and transformed into a digital image. The image is then sent to the feature extractor where it is processed to extract important features and allow a more compressed representation known as a template. The next step depends on the mode in which the system is

operating. If the system is operating in enrollment mode (*and the quality of the image is satisfactory*) the template is sent to storage, which may be a database or smartcard. If the system is operating in identification mode the template is sent to the matcher. The matcher retrieves the template of the claimed identity from storage and compares it with the recently acquired template resulting in a match score. If the match score is higher than some predetermined threshold the templates are said to match and a final decision of yes is output. Otherwise if the match score is lower than the threshold the templates are said to not match and a final decision of no is output. Figure 2.4 illustrates the structure of such a system along with possible attack points.

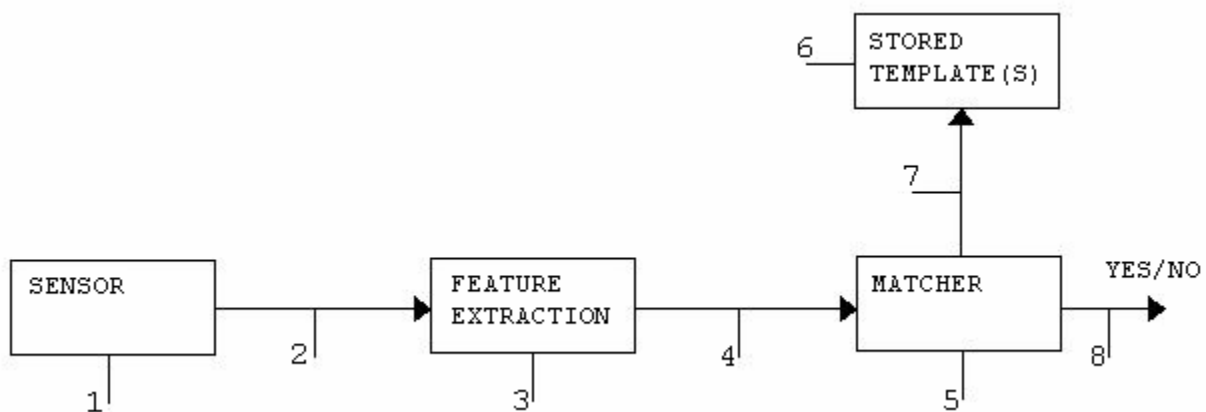


Figure 2.4: Ratha's Framework with Potential Attack Points [1]

Ratha et. al. identifies eight possible points of attack in this framework [1]. Labeled 1-8 in Figure 2.4 they are as follows:

1. Presenting a fake biometric to the sensor: This may be as simple as holding a picture up to the camera of a face recognition system or as complicated as making an artificial finger from a latent fingerprint left behind on a piece of glass.
2. Submitting a previously recorded biometric to the feature extractor: This attack involves recording the signal of biometric acquisition from a sensor then bypassing the sensor by submitting the recorded signal to the feature extractor.
3. Overriding the feature extraction process: The feature extractor is maliciously altered (*possibly with a Trojan horse*) and forced to produce templates with pre-selected features.
4. Tampering with the feature representation: Although the feature extractor and matcher components are often combined, making this type of attack extremely difficult, if they reside

- at separate locations and communicate via a network it is possible for a third party to snoop the communications channel and alter the templates before they reach the matcher.
5. Corrupting the matcher: In the type of attack the matcher is attacked (*again, possibly with a Trojan horse*) and forced to output predetermined match scores.
 6. Tampering with stored templates: The database is attacked and the biometric templates are altered, resulting in either a legitimate user being denied access to the system or an imposter gaining access to the system. Note that this attack may occur to a database on an information system or a smartcard carrying a biometric template.
 7. Attacking the communication channel between the template storage and the matcher: Here the data traveling through this channel are intercepted and modified before they reach the matcher.
 8. Overriding the final decision: If the final decision can be overridden by a malicious hacker the absence of vulnerabilities in all previous attack points will do the system no good.

Ratha's framework is good place to start when identifying vulnerabilities. Like any large problem identifying all possible vulnerabilities of a biometric system is a complex task when it is viewed holistically. The framework allows the analyst to partition the task, focusing on each potential attack point individually. This simplifies the task, facilitates collaboration, and increases the chance that all vulnerabilities will be identified.

2.2.2 Wayman's Framework

Ratha's framework may be useful in identifying potential attack points but it is too abstract to compensate for the intricacy of today's biometric systems. With the vast amount of activity taking place in the system the analyst needs a way to further partition the task of vulnerability identification while maintaining a holistic view. Wayman proposed a framework in [3] which allows the system to be viewed at both the micro and macro levels. While it includes the attack points from the previous framework it also adds numerous others. Figure 2.5 illustrates this framework.

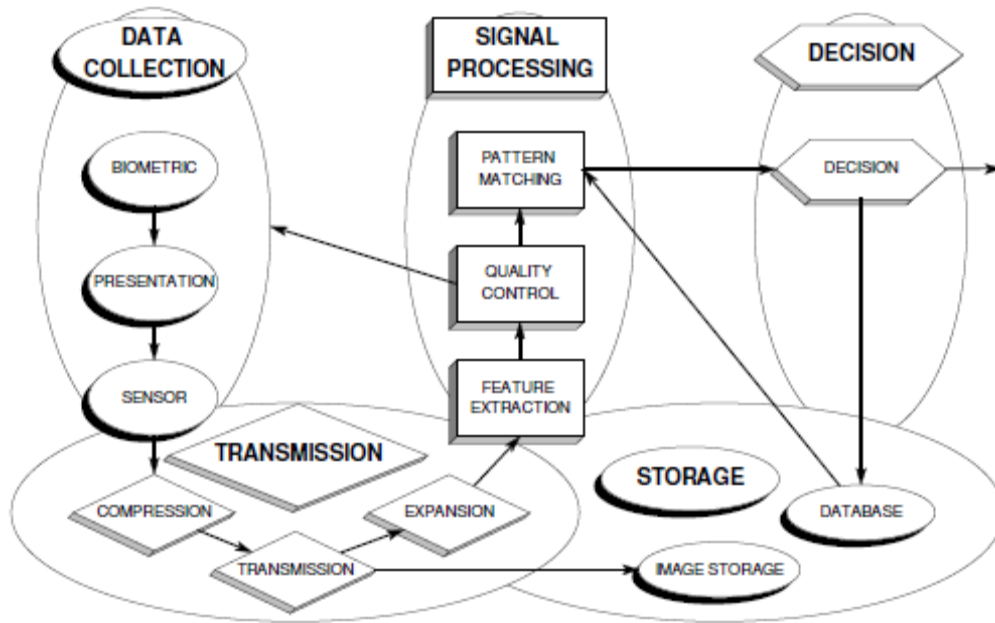


Figure 2.5: Wayman’s Framework [3]

Five broad areas (or subsystems, as Wayman refers to them) are represented, each with the potential for multiple attack points. Data collection includes the imaging of a biometric pattern presented to the sensor. Transmission includes the travel of data over a communication medium, which may or may not include compression and expansion. Signal processing includes feature extraction, quality control, and pattern matching. The storage of biometric templates may be in a database or smartcard. The decision is the Boolean output of “accept” or “reject” based on a match score and a threshold. As you can see this framework broadens the possibilities for use in vulnerability identification. Attack points will not be addressed in this framework, as this framework is a subset of the framework in section 2.2.2.

2.2.3 Bartlow and Cukic’s Framework

Wayman’s framework is undoubtedly an improvement from the one proposed by Ratha. It allows the analyst to further partition the task of identifying vulnerabilities. However, certain factors are not taken into consideration. Bartlow and Cukic address these issues in the framework proposed in [24]. This new framework, based on Wayman’s, adds three new areas of consideration: administrative supervision, underlying IT environment, and token presentation.

All biometric systems require some degree of administrative supervision. Although the level of supervision varies from system to system it is not hard to imagine the vulnerabilities that may be associated with it. Administrators are human, making them less predictable and harder to analyze. Vulnerabilities in this area will undoubtedly compromise the security of the even the most well designed system. A biometric system may or may not be associated with an underlying IT environment (*the latter is referred to as a stand alone system*). The IT environment is a larger system which the biometric system is a part of (*subsystem*). Interaction with the IT environment may introduce vulnerabilities not present in the previous frameworks. Token presentation is required in some biometric systems. These types of systems base their final decision on the presented biometric and the information in the token, which may introduce another potential point of attack to the system. A smartcard containing biometric data is an example of a token used in systems of this type. [24]

Figure 2.6 shows the new framework along with potential attack points (*labeled 1-20*) and the types of attacks that may occur at these points. 22 different types of vulnerabilities are described in [24]. Each is given a title and listed in Figure 2.6 near the attack points. “BAD ADMIN” refers to intentional or unintentional administrative errors, resulting in illegitimate users gaining access to the system or legitimate users being denied access to the system. This may be an outcome of a manual override or bad supervision practices. “FAIL SECURE” is a result of the biometric system or underlying IT environment being exposed to abnormal operating conditions. This may have been caused by a buffer overflow or a denial of service attack (*among others*). “POWER” refers to a power outage, which may render the system useless. “BAD USER” occurs when a legitimate user attempts to upgrade his or her privileges to that of an administrator. “UNDETECT”, although not a specific vulnerability, refers to attacks which are carried out without being detected by the system, which may encourage future attacks. “BYPASS” refers to an attack in which the attacker finds

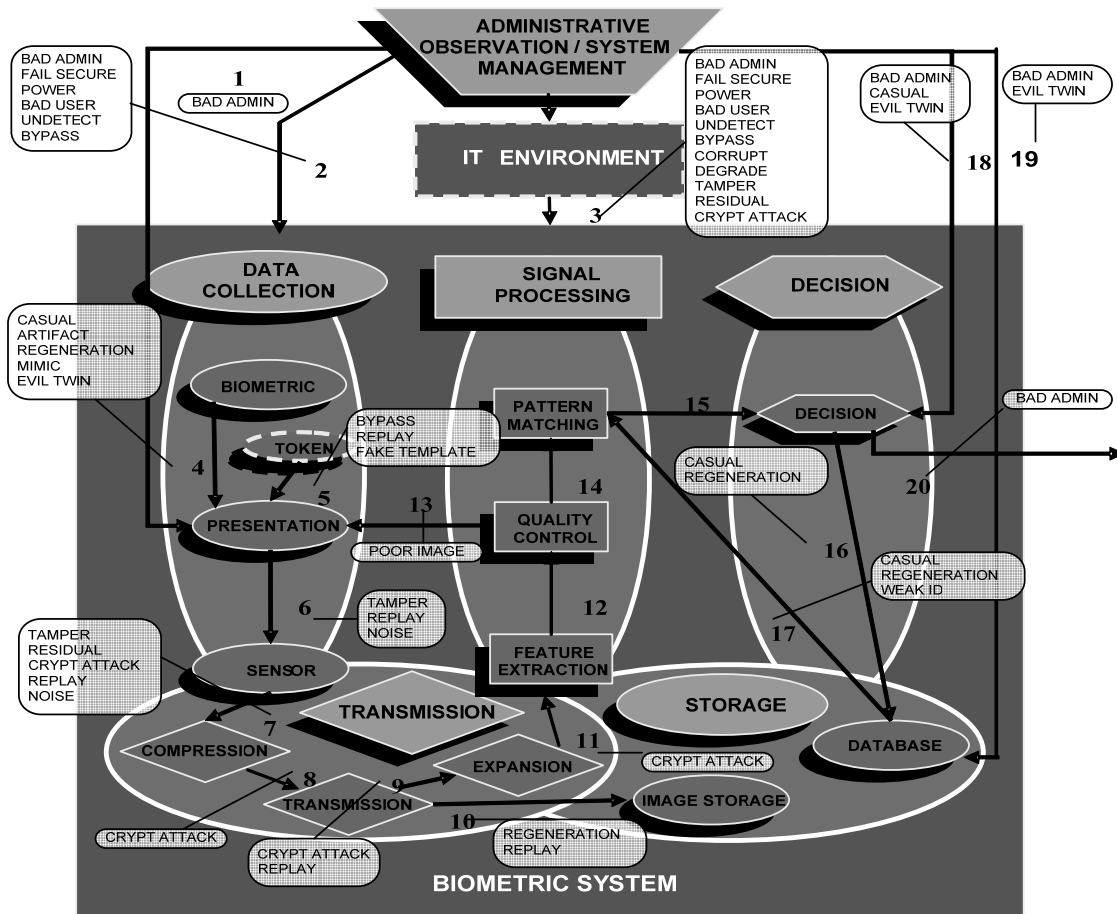


Figure 2.6: Bartlow and Cukic’s Framework [24]

another means of gaining access, bypassing the biometric system altogether. This may involve breaking down physical barriers, forcing a legitimate user to present his or her biometric to the sensor, or with the help of an accomplice, who is a legitimate user. A “CORRUPT” attack involves an attacker making modifications to the biometric system or the IT environment, putting the system in a vulnerable state. This may involve modifying system software or changing system parameters. “DEGRADE” vulnerabilities occur when certain software is installed in the underlying IT environment that degrades the security of the system. “TAMPER” refers to the modification to system hardware which may erroneously allow an individual into the system. “RESIDUAL” refers to latent fingerprints on a sensor being used to make artificial fingers. The artificial fingers may then be presented to the sensor, which may or may not result in an imposter gaining access to the system. “CRYPT ATTACK” refers to an imposter gaining biometric data by breaking the encryption (*or implementation of*) during data transmission. The data can then be used to perform

another type of attack, say “REPLAY”. A “CASUAL” attack occurs when an attacker repeatedly attempts to authenticate by presenting his or her biometric to the system. This type of attack relies on the false accept rate of the system. Success will occur if the system accepts the imposter’s biometric as matching that of the claimed identity. “ARTIFACT” refers to the fabrication of a fake biometric which is then used to authenticate as a legitimate user. Some biometrics like voice or gait are relatively weak. “MIMIC” involves an attacker imitating the biometric of a legitimate user in order to fool the system. “EVIL TWIN” attacks can occur if an individual’s biometric is close enough to that of a legitimate user to fool the system. “REPLAY” is the same as the second attack point in the Ratha’s framework. “FAKE TEMPLATE” involves the fabrication of a template which may be inserted into a database or stored on a smartcard. This may result in the attacker gaining access to the system. If “NOISE” is applied to the system it may erroneously allow access to the attacker. “POOR IMAGE” may exploit the quality control module of the system. If poor quality images are accepted at enrollment, the attacker may be able to fool the system into accepting his or her biometric as that of a noisy image. Similar to “POOR IMAGE”, “WEAK ID” relies on a weak template to fool the system. Finally, with “FAR/FRR” the attacker relies on the false accept or reject rate to fool the system into accepting his or her biometric. [4], [24]

It becomes apparent that a vast number of attack points and possible exploits exist. Understanding them in the general case may aid the analyst in identifying vulnerabilities for a specific system. Neither all of the vulnerabilities nor all of the attack points will be present in every system. This framework is meant to be general enough to be easily tailored to any system. The specifics of the system should be used to determine which areas of the framework are relevant and which need not be considered. Once this is completed the framework can be utilized in vulnerability identification.

2.3 Known Vulnerabilities

It is important for an analyst understand how attacks are performed. The analyst must be able to identify possible means of exploitation. To do this well one must be able to come up with attack scenarios that may or may not have been performed in the past. Like any subject, becoming proficient in creating something new must often begin with knowledge of what has been done in the past. For a biometric system security analyst this means studying typical vulnerabilities which are

present in the generic biometric system. In this section two potential vulnerabilities mentioned in the previous section are further elaborated.

2.3.1 Forged Biometric

Forging a biometric is one of the most common vulnerability exploits mentioned in published research [24], [1], [2], [8], [6]. As mentioned previously in the description of “ARTIFACT”, forging a biometric occurs when an attacker fabricates a fake representation of the biometric of a legitimate user. This may be in the form of a fingertip made out of silicon rubber, a photograph of an individual’s face, or a contact lens with a legitimate user’s iris pattern printed on it among others.

There are a number of ways an attacker may obtain the biometric data necessary to perform this type of attack. The data may be obtained directly from the original biometric [6], from a template [5], or it may be generated [2], [8]. Certain factors may be out of the system’s control. For instance, it is impossible to stop an attacker from obtaining biometric data directly from a legitimate user’s biometric. The data may be obtained from a latent fingerprint on a glass surface, from a photograph of a face or iris, or even directly from a mold of a finger or a hand. Furthermore, this may be done with or without the knowledge of the victim; or with or without their consent. This may happen in the vicinity of the system or wherever else the victim may wander.

Though the system cannot prevent the acquisition of biometric data from the user, measures may be taken to prevent the generation of the data. There have been techniques proposed that allow an attacker to generate a biometric template which is close enough to that of a legitimate user to fool the system into accepting it [2]. This approach, called a “hill climbing” technique, assumes that the attacker has access to the match score. It then can be determined if certain alterations of the generated template will raise the score. Repeating this process a large number of times will eventually result in the acceptance of the generated template. Using fingerprints as an example, templates carry information about minutiae points. A technique proposed by Ross et. al. describes a method of reconstructing fingerprints from minutiae points [5]. Combining these two techniques an attacker may use the minutiae information in the generated template to reconstruct an image of a fingerprint which can be used to make an artificial finger that may fool the system.

Prevention of data acquisition is not entirely possible, although it is possible to stop the acquisition based on information in the system such as match scores and templates. Preventing one means of acquisition will only force the attacker to find another means. The only surefire way to

thwart this type of attack is to prevent the system from accepting artificial biometrics, a task that is far from trivial.

2.3.2 Replay Attack

A Replay Attack is performed by replaying a previously stored signal in order to fool the system into accepting the signal as legitimate. We will assume the context of attack point 2 in Ratha's framework, which is the communication channel between the sensor and the feature extractor. One obvious way to thwart such an attack is to encrypt the transmission. Although this may be effective there are drawbacks, the relevance of which may vary from system to system.

Encryption may be computationally intensive depending on the algorithm used. Depending on the processing requirements, this could potentially raise the cost and size of the sensors which is undesirable. Furthermore, depending on the number of sensors, there may be a need to manage a large number of keys, which could potentially introduce another vulnerability. If the symmetric key used to encrypt the transmission of data from the sensor to the feature extractor does not change with each transmission, the encryption will have no effect in thwarting a replay attack of this nature. The attacker can simply replay the encrypted signal as if it were generated by the sensor.

Replay attacks are a real threat and countering them may not be as straightforward as it may appear. Recall the structure of a biometric system from Ratha's framework (*for simplicity*). The sensor and feature extractor modules may reside in the same vicinity or at separate locations altogether. The communication channel may be a small network or the Internet. If the modules are forced to communicate over the Internet, the number of devices that can snoop the transmitted packets is large, greatly increasing the threat. Encryption may solve some security problems but it is not foolproof. A better method is needed to ensure that a signal has originated at the sensor and is not a replay.

2.4 Countermeasures for Known Vulnerabilities

Because the vulnerabilities described in the previous two sections are well known in published works several countermeasures have been proposed to thwart these types of attacks [1], [14]. Liveness detection and challenge response are two that are well known. The next two sections elaborate on these with a general description and specific examples of each.

2.4.1 Liveness Detection

Liveness detection is a countermeasure that prevents the use of artificial biometrics by detecting whether or not the presented biometric is part of a living person. Many techniques have been proposed but not all are effective [18]. Liveness may be detected by adding extra hardware to the sensor or by using features of the captured data. The context of fingerprints will be assumed to simplify the discussion of the subject.

Although adding extra hardware to the system is often expensive, it is simpler to detect liveness in this way than it is using the captured data. Some suggested hardware methods use temperature, optical properties, pulse, blood pressure, and electric resistance among others. These methods will detect the presence of life at the sensor without a doubt. However, it may be easy to fool the system by using a wafer thin artificial print placed on the tip of a live finger. In this case the life that the system is detecting is underneath the artificial print. [18]

Using features of the captured data to detect liveness is a more promising approach despite the added complexity. Several methods have been proposed but few have been thoroughly researched. Among the less researched methods are skin deformation (*when pressed against the sensor*), pore size and position, and side fingerprints. A promising method observing the perspiration over a short time period in a fingerprint image is presented in [17]. This method uses the fact that perspiration from pores increases over time to detect liveness in fingerprints. When a live finger is placed on a capacitive sensor and left there for a period of time the regions close to the pores will get darker. This does not happen in cadaver or artificial fingers. The research presented in [17] indicates that testing of this method resulted in correct classification 100 percent of the time, though the sample tested consisted of only six live, cadaver, and artificial fingers (*18 total*). More research is needed to determine the method's effectiveness in varying conditions. [18]

Fabrication of artificial biometrics is one of the easiest and least expensive types of attack. The fabrication of artificial biometric fingerprints in particular has been proven to fool commercial biometric systems that claim to have liveness detection technology [6]. It is apparent that a more robust method of detecting liveness is needed to thwart these attacks. The method presented in [17] looks promising but further research is needed to determine usefulness under various operating conditions.

2.4.2 Challenge Response

Much like an artificial biometric is used to fool the sensor, an artificial signal can be used to fool the feature extractor (*replay attack*). Similarly, as liveness detection is used to check the liveness of a biometric, challenge response can be used to check the liveness of a signal. Challenge response in the traditional sense has been used in applications such as credit card inquiries over the phone to increase security. Typically the customer is asked for their mother’s maiden name to ensure that there is not an imposter on the other end of the line. The same principle can be applied to the communication protocol between the sensor and the feature extraction module.

Ratha et. al. presents a challenge response scheme in [1] based on a system with a remote sensor client. In this scheme a transaction is initiated at the remote end at the time of signal acquisition. The server pseudo-randomly generates a response string and sends it to the client where it is then sent to the sensor. It is important to note that the sensor is assumed to have processing capability, perhaps a dedicated response coprocessor. Once the sensor receives the challenge string it sends the acquired signal followed by its response, which is based on the challenge string and the acquired signal. For instance, the response string may consist of a series of pixel numbers; say “10, 33, 102”. Based on this string the sensor will send the values of the 10th, 33rd, and 102nd pixels in response. Since the response string is different for each signal acquisition it cannot be guessed. Likewise, since the response coprocessor is tightly coupled with the sensor it is very difficult for an attacker to intercept the communications between them. Note that it is assumed the client/server communication is secure. Figure 2.6 illustrates this challenge response scheme. [1]

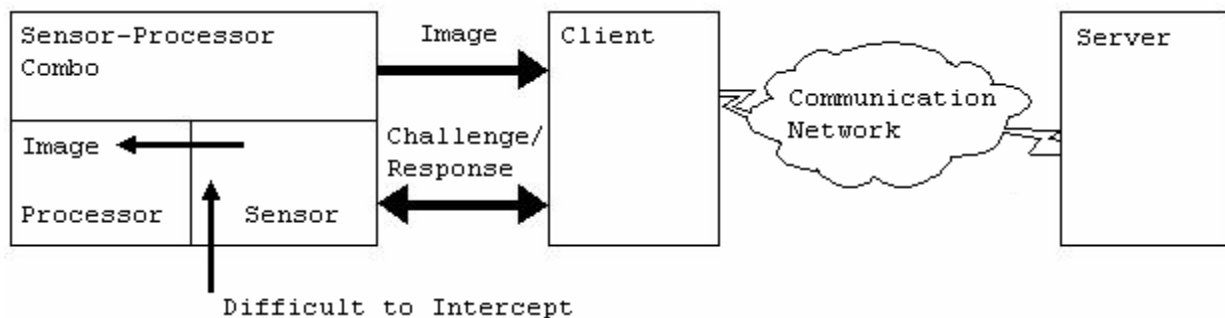


Figure 2.7: Ratha’s Challenge Response Scheme [1]

The response string in this example is just one possibility. The system may use multiple response functions and choose between them randomly. For instance, the sensor may possess some secret information such as a set of keys. Suppose these keys have the values “29, 31, 66” and the

sensor receives the same response string. Challenge response is an effective countermeasure to replay attacks, especially when the sensor resides at a remote location.

Chapter 3

Application of Attack Trees to Biometric Systems

As stated in Section 2.1, attack trees provide an effective means of vulnerability analysis because they are easy to understand, they scale well, and they facilitate modularity and collaboration. These benefits can be utilized in the design of biometric systems. In the following subsections two attacks on biometric systems will be analyzed using attack trees. These attacks are specific exploitations of the vulnerabilities described in Sections 2.3.1 and 2.3.2. For each attack a detailed description is given, the attack tree representation is shown, and the attack is analyzed.

3.1 "Gummy" Finger Attack

The "Gummy" Finger Attack is an exploit of the Forged Biometric vulnerability. In this attack the fingerprint of an authorized user of a particular biometric system is replicated in the form of a wafer thin piece of gelatin which bears the print of the user on one side of its surface. The attacker places the piece of gelatin over his real finger. He then presents the biometric to the sensor and gains access to the system; afterwards he eats the "gummy" finger. This attack is based on Matsumoto's experiments in [6].

In order to perform this attack the attacker must make the "gummy" based on an authorized user's fingerprint AND use the "gummy" finger to gain access to the system. To make the "gummy" finger, he must make the mold and fill the mold with gelatin, after which he must let the gelatin cool and carve out the back of the "gummy" finger until wafer thin. The mold can be made by either having an authorized user press her finger into impression material OR using a latent fingerprint to make the mold. An authorized user is unlikely to voluntarily participate in the attack without some convincing. The attacker may be able to bribe the user if the price is right and the user is corruptible. A more likely scenario would involve forcing the user to participate in the attack which would probably include a threat to the welfare of that individual. [6]

Alternatively it has been show that molds based on latent fingerprints can produce "gummy" fingers that will fool several commercial systems with roughly the same success rate as those made

from a live finger, which is about 80% of the time. In order to make a mold from a latent fingerprint the attacker must transfer an image of the print to a transparency sheet AND use the transparency image to etch the print onto photo-sensitive printed circuit board. [6]

To accurately transfer the image to a transparency sheet a few things must happen. Latent fingerprints are not always easy to see clearly; therefore it must be enhanced with a cyanoacrylate adhesive. Once this is done it can be photographed with a digital camera. The image can then be transferred to a computer and enhanced with a photo suite such as Adobe Photoshop. The image can then be printed on the transparency sheet. [6]

Once the “gummy” finger is made from either a live finger or latent fingerprint the attacker can use it to gain access to the biometric system. For a more holistic view of this attack refer to its attack tree representation in the next subsection. As mentioned earlier AND nodes are denoted with circular arch connecting the arrows drawn from the node to its children. All child events must take place to satisfy the sub-goal. OR nodes are denoted by the absence of the circular arch. Only one child event must take place to satisfy the sub-goal in this case. All nodes associated with the tree are rectangular and have attributes associated with abilities of the attacker. These include educational complexity level, denoted by $E\#$ (*where # is a number*) and defined in Table 2.1; financial cost, denoted by $F\#$ and defined in Table 2.2; and temporal cost, denoted by $T\#$ and defined in Table 2.3. The root node has the additional attribute risk. Risk is associated with the system defense. The method for assessing risk is described in section 2.1.3.2, Tables 2.4 – 2.5, and Figure 2.2. The oval shaped node describes a countermeasure to thwart the attack. It has a financial cost associated with it. The dotted line connecting this node to the tree denotes that the countermeasure thwarts the root’s right child sub-goal, thereby thwarting the entire attack.

3.1.1 Attack Tree Representation

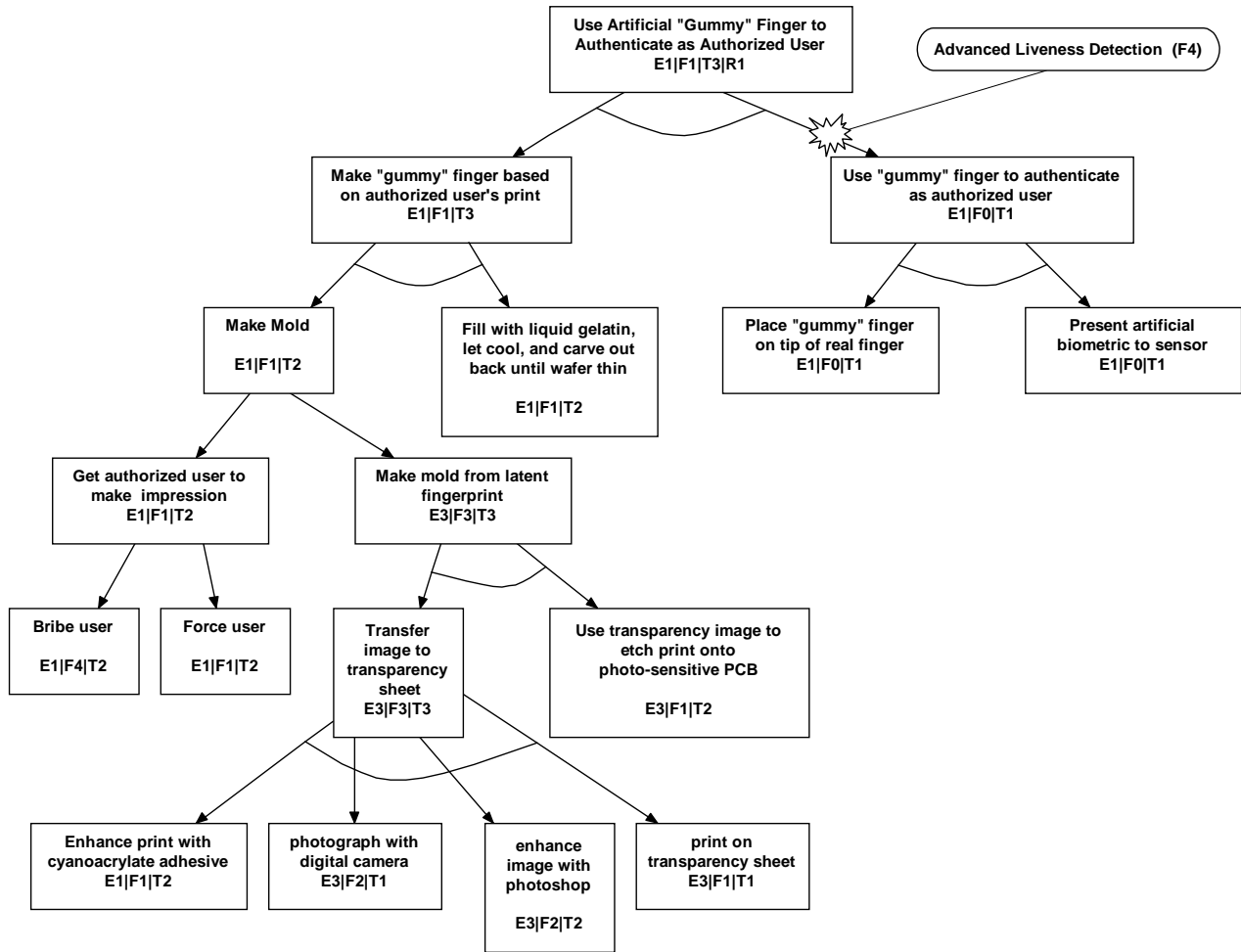


Figure 3.1: “Gummy” Finger Attack Tree Representation

3.1.2 Analysis

The attributes associated with the leaf nodes of the attack tree representation in the previous subsection must be quantified in terms of the attacker and in terms of system defense. The quantification relative to the attacker is calculated by propagating the attribute aggregation in the direction of the root node. The attributes that will be aggregated in this case are the educational complexity, financial cost, and temporal cost. Refer to section 2.1.4 for the attribute aggregation criteria. The method of aggregation depends on the information desired. For instance, if the analyst wants to know the least expensive attack, the financial cost attribute will be quantified as follows: the value of an AND node becomes the sum of its children’s values; the value of an OR node

becomes the lowest child value. Similar methods can be applied to find the attack with the least educational complexity requirement or the one that takes the least amount of time.

In this example we are assuming that we want the attack with the lowest financial cost, least educational complexity requirement, and least amount of time. Root node values should be considered separately in this example. The root values were quantified at E1, F1, and T3. This means that the attack may be carried out by someone with no computing education; it may cost as little as under a hundred dollars; and it could take as little as hours to complete. This does not mean that any form of attack will take on these values; it just means that they are the minimum values possible for each attribute. In practice, it is likely the analyst would explore all means of attack and quantify the attribute values for each.

The root node value R1 indicates a level 1 risk. Risk is only considered at the root node since it is the end result of every attack, and therefore the same for all. In other attacks there may be a risk associated with certain sub-goals or leaf nodes. Deciding which nodes will possess this attribute is ultimately up to the analyst. Assessing risk may be somewhat subjective and depend largely on the system being analyzed. In this case the root goal is assumed to have a category II consequence severity and a category B likelihood of occurrence, resulting in a level 1 risk, which should be countered with the highest priority.

The chosen countermeasure is advanced liveness detection (*perhaps the method described in [17]*). Liveness detection (*assuming it is effective*) will stop the user from using the “gummy” finger to gain access to the system, therefore cutting off the entire right side of the tree and thwarting the attack on the system. This countermeasure comes with a level 4 financial cost (*tens of thousands of dollars*) which may make the analyst think twice before implementing it. However, since the risk is so high it becomes apparent that the cost of not implementing the countermeasure outweighs the cost of implementing it.

3.2 Replay Attack

The Replay attack described earlier will be represented in the form of an attack tree in the following sub-section. This attack assumes a system structure similar to that of Figure 2.7. The point of attack would be 2 under Ratha’s framework and 7, 8, or 9 under Bartlow and Cukic’s framework.

In order to replay the signal to the authentication server the attacker must first obtain the signal AND then use it to fool the authentication server. To obtain the signal he can either decrypt it after it leaves the client OR obtain it before it leaves the client. To decrypt signal a brute force OR side channel approach may be taken. Although neither one looks promising in this case. To obtain the signal before it leaves the client he must gain administrator access on the client's system AND install a malicious program to obtain the signal. To gain administrator access he could exploit a buffer overflow OR break the administrator password. Exploiting a buffer overflow requires the attacker to identify susceptible code, construct the input to inject, AND execute the code. To break the administrator password the attacker can use brute force, social engineering, OR bribe the administrator (*unlikely*). Once administrator access is gained the attacker may (*remotely*) install a malicious program that will obtain the biometric signal before it leaves the client and send it to the attacker.

Once the attacker has the signal he can then send it to the authentication server. To do this the attacker can either use the malicious program (*if installed*) or use a fake sensor to send the signal. The attacker may be required to be in the vicinity of the system to utilize the exploitation of the vulnerability so an accomplice is used in the case that the malicious program is used to send the signal. Therefore the attacker must call the accomplice to give the "go" order AND the accomplice must send an instruction to the malicious program. The program will then send the biometric signal to the authentication server. Alternately, the attacker can fabricate a fake sensor to send the pre-recorded signal to the client. To do this the attacker must build the fake sensor AND connect the fake sensor to the client system. Building the fake sensor requires that the attacker purchase a real sensor and a programmable microcontroller; program the microcontroller; and replace sensor insides with the programmed microcontroller. Once the fake sensor is built, it must be connected to the client system. Connecting the sensor requires that sensor interface specifications be researched, the old sensor be disconnected, AND the fake sensor must be interfaced with the client system. The fake sensor can then be used to send the pre-recorded signal. Refer to the attack tree representation in the next sub-section for a holistic view.

3.2.1 Attack Tree Representation

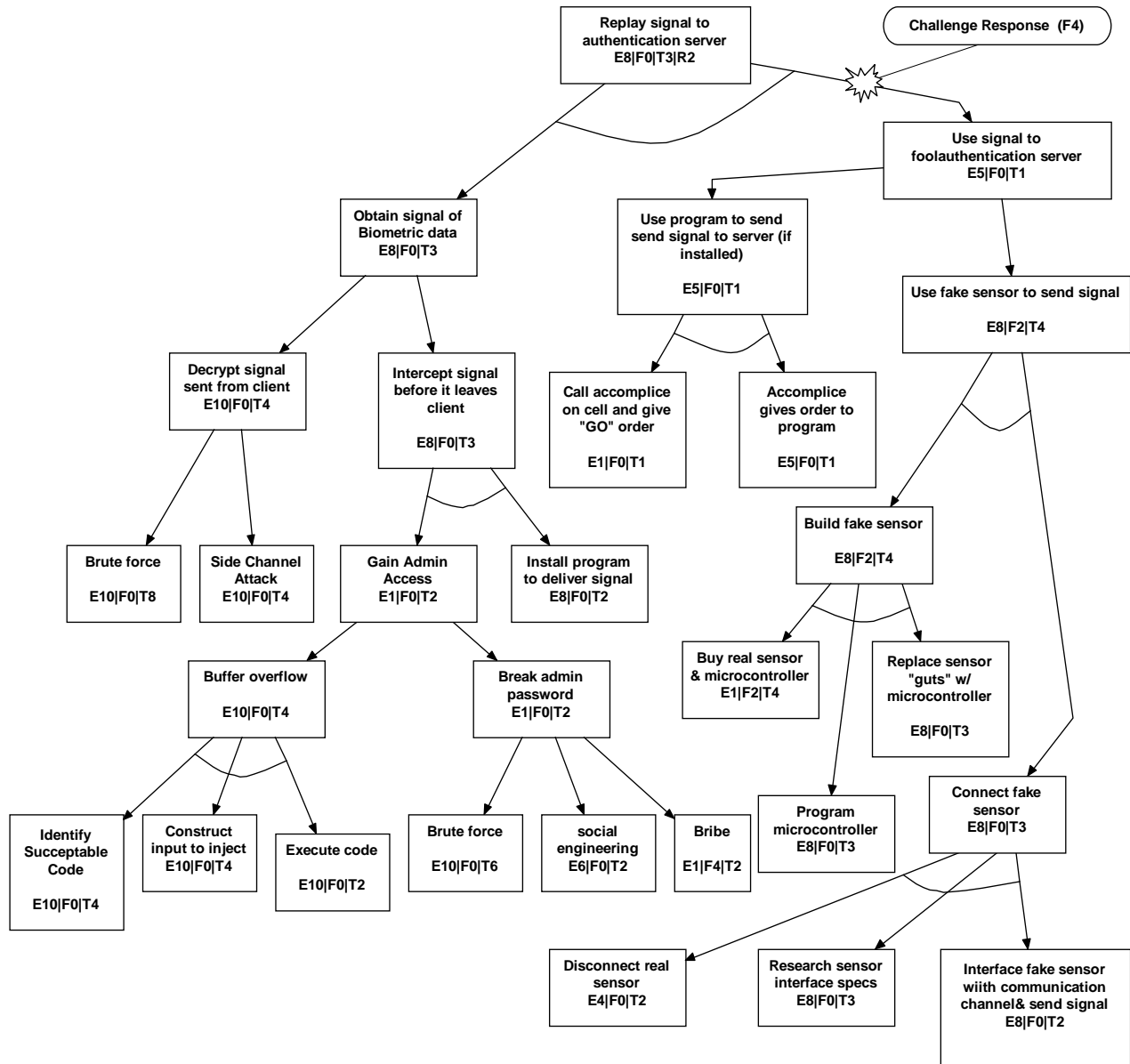


Figure 3.2: Signal Replay Attack Tree Representation

3.2.2 Analysis

The analysis of this attack tree was performed in the same manner as that of the “gummy” finger attack. Therefore it is not necessary to go into the same amount of detail because the methodology has already been explained.

Looking at the quantified attributes of the root node it becomes clear that the attacker must be well educated. In addition, he may not need much money or time to complete the attack. Although the likelihood of this attack is remote the consequences of exploitation are high, thus earning a risk level of 2.

The chosen countermeasure is a challenge response scheme (*perhaps the one presented in [1]*). Since challenge response checks the liveness of the signal it can tell whether it came from a real sensor within a reasonable timeframe. Any replayed signal will be considered invalid. The sensor is assumed to have a response protocol (*and possibly secret information*) which cannot be determined outside the sensor or authentication server. Responses generated by the sensor are based on data received from the server, secret data within the sensor, and the biometric signal, so reproduction is deemed improbable. Therefore the right side of the tree is “pruned” and the attack is thwarted. This countermeasure has a financial cost level of 4 which may seem like a lot, but considering the risk level it is well worth the price.

Chapter 4

Summary and Future Work

4.1 Summary

Vulnerability analysis is an important step in the security process of any system. The security process starts with system design; thus this is where the analysis should begin. Biometric systems aim to provide secure and reliable authentication; therefore it is crucial that the security of these systems be analyzed at the design stage so that vulnerabilities are identified and countered before they can be exploited. Although biometric systems solve many problems associated with traditional authentication systems, drawbacks in the form of vulnerabilities and threats exist and must be addressed.

The task of identifying vulnerabilities and potential means of exploitation is complex and time consuming. The large task of identifying vulnerabilities may be partitioned using a framework of biometric system structure to identify potential attack points. Ratha's framework views the generic biometric system as a pattern recognition system. Component modules include the sensor, feature extraction, matcher, and template storage. Although this framework is useful in identifying potential attack points it is too abstract to deal with the intricacy of today's biometric systems. Wayman's framework provides a better means of identifying potential attack points but only addresses the traditional stand-alone biometric system. Bartlow and Cukic's framework is an improvement over both. It is more comprehensive than either one while maintaining the holistic view of Wayman's framework. Bartlow and Cukic's framework addresses three areas which were not considered previously: administrative supervision, an underlying IT environment, and token presentation. Using this framework the task of vulnerability identification is simplified by allowing for a greater degree of partitioning.

After vulnerabilities are identified they must be analyzed to determine potential means of exploitation and develop countermeasures to thwart attacks. Much like a framework simplifies the task of vulnerability identification, the attack tree methodology simplifies the task of vulnerability

analysis. This methodology allows attacks on a system to be represented in a tree structure, with the root node as the goal of the attack and its children as ways of accomplishing that goal [10]. This allows the analyst to represent complex attack scenarios while maintaining a holistic view. Attack trees help the analyst understand ways in which a system may be attacked, which helps determine which countermeasures may be necessary to thwart the attack. The attacker's abilities are also represented in the tree, which helps assess the likeliness of the attack. Risk, assessed using the method in [22], along with cost of implementation may be used to determine whether or not to counter a threat.

Despite vendors' reluctance to admit it, biometric system vulnerabilities do exist. Matsumoto's "gummy" finger attack is proof of this [6]. It's difficult to determine why so many commercial systems possess the same vulnerability. However, it's not difficult to see that there may be flaws in the vulnerability analysis techniques used by these vendors. Whether it's their methodology or the competency of their designers, the security process can be improved upon. Attack trees aim to do just that: improve the security process of system design and maintenance.

4.2 Suggestions for Future Work

A general approach to vulnerability analysis of biometric systems using attack trees has been suggested. Implementations of biometric systems may vary in system structure and functionality, which affects the presence of particular attack points and vulnerabilities. The framework used to identify vulnerabilities should be tailored to each individual system so that only relevant attack points are considered. A formal methodology for framework tailoring would simplify this task. Also, the underlying IT environment will vary greatly from system to system. A framework for modeling the interaction between the biometric system and the underlying IT environment may simplify the task of identifying the associated potential attack points.

Many security vulnerabilities could be avoided or discovered early if knowledge sharing between groups of system engineers and security experts were improved upon. Steffan and Schumacher suggest a wiki¹ web based system for collaborative attack modeling [21]. Biometric system engineers could use this type of system to share identified vulnerabilities along with the

¹ A wiki is a type of website that allows anyone visiting the site to add, remove, or otherwise edit all content, quickly and easily, often without the need for registration. (www.wikipedia.com)

attack tree representations. This could speed up the vulnerability identification process as well as improve upon the completeness of attack trees and evaluation of countermeasure effectiveness.

References

- [1] N. K. Ratha, J. H. Connell, R. M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," IBM Systems Journal, VOL 40. NO 3 2001.
- [2] U. Uludag, A. Jain, "Attacks on Biometric Systems: A Case Study in Fingerprints," Proc. SPIE-EI 2004, Security, Steganography, and Watermarking of Multimedia Contents VI, San Jose, CA, January 18-22, 2004.
- [3] J.L. Wayman, "Technical Testing and Evaluation of Biometric Devices," in A. Jain, et. al, Biometrics – Personal Identification in a Networked Society, Kluwer Academic Publisher, 1999.
- [4] A. Kong, A. Griffith, et al, "Department of Defense & Federal Biometric System Protection Profile for Medium Robustness Environments," Version 0.02, March 2002.
- [5] Ross, A., Shah, J., Jain, A. "Towards Reconstructing Fingerprints From Minutiae Points," SPIE Conference on Biometric Technology for Human Identification II, Vol. 5779, pp. 68-80, March 2005.
- [6] T. Matsumoto, H. Matsumoto, K. Yamada, S. Hoshino, "Impact of Artificial Gummy Fingers on Fingerprint Systems," Proceedings of SPIE Vol. #4677, Optical Security and Counterfeit Deterrence Techniques IV, 2002.
- [7] B. Schneier, "Inside Risks: The Risks of Relying on Cryptography," Communications of the ACM, Vol. 42 No. 10 Oct. 1999.
- [8] N.K. Ratha, J.H. Connell, R.M. Bolle, "An Analysis of Minutiae Matching Strength," Proc. 3rd AVBPA, Halmstad, Sweden, June 2001, pp. 223-228.
- [9] E. Chen, P. Szor, "Blended Attack Exploits, Vulnerabilities and Buffer-Overflow Techniques in Computer Viruses," Symantec Security Response, Symantec Corporation, June 2003, <http://enterprisesecurity.symantec.com/PDF/Blended_Attacks.pdf?EID=0>.
- [10] B. Schneier, "Attack Trees," Dr. Dobb's Journal, vol. 24, no. 12, December, 1999, pp. 21-29.
- [11] Vidalis, S., Jones, A., "Using Vulnerability Trees for Decision Making in Threat Assessment," Technical Report, University of Glamorgan, Pontypridd, June 2003.
- [12] Moore, A.P., R.J. Ellison, and R.C. Linger, "Attack Modeling for Information Security and Survivability," Software Engineering Institute Technical Report CMU/SEI-2000-TN-001, March 2001. <<http://cert.org/archive/pdf/01tn001.pdf>>
- [13] L. O’Gorman, "Fingerprint Verification," in A. Jain, et. al, Biometrics – Personal Identification in a Networked Society, Kluwer Academic Publisher, 1999.

- [14] N. K. Ratha, R. M. Bolle “Smartcard Based Authentication,” in A. Jain, et. al, *Biometrics – Personal Identification in a Networked Society*, Kluwer Academic Publisher, 1999.
- [15] Moore, A.P. and R.J. Ellison, “Survivability Through Intrusion-Aware Design,” Software Engineering Institute Technical Report 2002.
<<http://www.cert.org/archive/pdf/intrusion-aware.pdf>>
- [16] Arbaugh, W.A., W.L. Fithen, and J. McHugh, “Windows of Vulnerability: A Case Study Analysis,” *IEEE Computer*, Vol. 33, No. 12, Dec. 2000.
- [17] R. Derakhshani, S. A. C. Schuckers, L. Hornak, and L. O’Gorman. “Determination of vitality from a non-invasive biomedical measurement for use in fingerprint scanners,” *Pattern Recognition*, 36(2):383-396, 2003.
- [18] Sandström, Marie. “Liveness Detection in Fingerprint Recognition Systems” [Detektering av Artificiella Fingeravtryck vid Användarautentisering].
<<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-2397>> (2006-03-27)
- [19] S. Pankanti, R. Bolle, and A. K. Jain, *Biometrics: The Future of Identification*, Guest Editor’s Introduction, *IEEE Computer*, pp. 46-49, February 2000.
- [20] J. Ortega-Garcia, J. Bigun, D. Reynolds, J. Gonzales-Rodriguez, Authentication gets personal with biometrics, *IEEE Signal Processing Magazine* 21 (2) (2004) 50-62.
- [21] J. Steffan, M. Schumacher, “Collaborative Attack Modeling,” *Proc. of the 2002 Symposium on Applied Computing*, March 2002.
- [22] General Accounting Office, *Information Security Risk Assessment Practices of Leading Organizations*, GAO/AIMD-99-139, August 1999.
- [23] B. Cukic and N. Bartlow, “Biometric System Threats and Countermeasures: A Risk Based Approach,” PowerPoint Presentation, *Proc. of the 2005 Biometric Consortium Conference*, Sept. 2005.
- [24] N. Bartlow, B.Cukic, “The Vulnerabilities of Biometric Systems – An Integrated Look at Old and New Ideas,” *Technical Report*, West Virginia University, 2005.
- [25] A. Jain, S. Pankanti, and R. Bolle “Introduction to Biometrics,” in A. Jain, et. al, *Biometrics – Personal Identification in a Networked Society*, Kluwer Academic Publisher, 1999.