INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®

Security of Information Systems:

What Technical Solutions Exist and What is Needed

Ryszard Gwozd

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Computer Science at

Concordia University

August 2000

Abstract

Security of Information Systems:

What Technical Solutions Exist and What is Needed

Ryszard Gwozd

The objective of this thesis is to review the need for information system security, evaluate the technical solutions commonly used today, make recommendations on where extra research is needed, and to overview the technologies, algorithms, standards, and security policies used in providing computer and communications security.

The information-security issue is equally important to the users of the Internet (e.g., businesses engaged in e-commerce) and to users of stand-alone ("air-gapped") systems. The problems that these two groups of users face are both different and have a different scale. However, there is no question that the urgent security problems are driven by the increasing demand for all-to-all connectivity; security is a top issue primarily because of the Net. Internet security is the most urgent and the most challenging field in computer security.

Moreover, cyber security involves issues that go beyond technological solutions, algorithms and hardware. Often ignored or not stressed enough are user education and prudent engineering.

Today, we are more alert about security, but still lack a complete understanding of what it means, how far it should go and how to accomplish it. Nevertheless, people are starting to feel that they should accept it, or at least try to understand it. Many users have an impression that they ought to know more about computer security, but few make the effort to learn about it. This thesis explains why the user has to be educated about security, what are different security levels, trusted systems, encryption, mandatory access control, and legislation. It helps the readers to understand the basics of computer security that will persuade them to practice safe computing.

Acknowledgements and Dedications

I am grateful to the many thoughtful reviews of interim versions of this thesis done by my thesis supervisor Dr. David K. Probst. His extraordinary perspective on computer technologies, especially computer security, his constructive criticism and his guidance were invaluable in helping to shape this thesis. The exchange of ideas we have had, helped me validate the foundations of this study. These discussions also helped me stay focused on most essential topics of information security.

I would like to dedicate this work to my wonderful parents, Maria and Włodzimierz Gwozd. Both of them were very supportive in all my educational endeavors. Even though neither of my parents were experts in high-tech industry, they understood completely the importance of education and research in advancements on personal and societal levels. My parents were great advisors, guides and teachers; my mom was actually my first professor in the primary school. Thank you mom, I appreciate your support and plan to present you with a copy of this thesis on my next visit home. Before I finish, I wish to thank all my family for constant encouragement and for being there for me, when mostly needed.

Ryszard Gwozd

List of Figures viii
Chapter 1 1
Introduction 1
1.1. OVERVIEW 2
1.2. CONTRIBUTIONS 4
1.3. IMPORTANT TERMINOLOGY 5
1.4. REMARKS 9
Chapter 2 11
Computer Security Fundamentals 11
2.1. EVOLUTION OF SECURITY NEEDS 11
2.2. INFORMATION IS A STRATEGIC RESOURCE 12
2.3. INFORMATION SECURITY VULNERABILITIES 13
2.3.1. Physical Security Holes 14
2.3.2. Trust Model and Security Policy 14
2.3.3. Faulty Configuration 15
2.3.4. Software and Hardware Errors 15
2.3.5. Cryptographic Protocols 15
2.3.6. Education and Personnel Assurance 16
2.4. CATEGORY OF ATTACKER 16
2.4.1. Professionals and Amateurs 17
2.4.2. Insiders and Outsiders 17
2.5. BUILDING A SECURE SYSTEM 18
2.5.1. Security through Obscurity 19
2.5.2. Suitable Security Policy 19
2.5.3. Security Perimeters 20
2.5.4. Physical, Personnel and Legal Controls 21
2.6. REMARKS 22
Chapter 3 25
Access Control 25
3.1. SECURITY MODELS 27
3.2. DOD GOAL SECURITY ARCHITECTURE 29
3.3. OPERATING SYSTEM ACCESS CONTROL 30
3.3.1. Decision Subsystem 30
3.3.2. Enforcement Subsystem 31
3.4. APPLICATION ACCESS CONTROL 32
3.5. ALTERNATIVES 32
3.6. REMARKS 34
Chapter 4 36
Identification and Authentication 36
4.1. USER IDENTIFICATION 36
4.2. SIGNATURE SCHEMES (DIGITAL SIGNATURES) 38
4.2.1. RSA 39
4.2.2. ElGamal 41
4.2.3. Diffie-Hellman 42
4.2.4. Binding Signatures to Messages 43
4.2.5. Verifying Signatures 44

4.2.6. Idempotence and Non-repudiation 44
4.3. IDENTIFICATION SCHEMES 45
4.3.1. Secure Remote Login 49
4.3.2. Address-based Authentication 50
4.3.3. Cryptography-based Authentication 51
4.3.4. New User Services 51
4.4. AUTHENTICATION CODES (MACS) 52
4.4.1. Message Integrity 53
4.4.2. Message Author Authentication 53
4.5. REMARKS 54
Chapter 5 57
Cryptography and PKI 57
5.1. BASIC CRYPTOGRAPHIC SERVICES 58
5.2. PRIVATE KEY CRYPTOGRAPHY 60
5.2.1. Data Encryption Standard 61
5.2.2. International Data Encryption Algorithm 61
5.3. PUBLIC KEY CRYPTOGRAPHY 62
5.3.1. Rivest, Shamir, Adleman Algorithm 64
5.4. VULNERABILITIES OF CRYPTOGRAPHIC PRODUCTS 64
5.5. KEY MANAGEMENT PROBLEM 65
5.6. KEY DISTRIBUTION CENTERS 65
5.7. PUBLIC KEY INFRASTRUCTURE 66
5.7.1. Large-scale Deployments 69
5.7.2. PKI Promoters 69
5.8. REMARKS 70
Chapter 6 73
Network Access Control 73
6.1. CLOSED USER GROUPS 74
6.2. FIREWALLS 74
6.2.1. Limitations 77
6.3. VIRTUAL PRIVATE NETWORKS 78
6.3.1. Quality of Service 81
6.4. GUARDS 81
6.5. REMARKS 82
Chapter 7 85
Availability and Denial of Service 85
7.1. Attacks 86
7.1.1. SYN Floods 86
7.1.2. UDP Diagnostics 88
7.2. SERVICE THEFT 88
7.3. DEFENSES 89
7.3.1. Resource Requirements 89
7.3.2. Router Configuration 90
7.3.3. Protocol Modifications 91
7.3.4. Firewall Approach 91
7.4. REMARKS 95

Chapter 8 97
Foreign/Mobile Code 97
8.1. SOFTWARE DISTRIBUTION 98
8.2. WORLD WIDE WEB 98
8.3. COMMON GATEWAY INTERFACE 99
8.4. COOKIES 100
8.5. ACTIVEX 101
8.6. JAVA 103
8.7. JAVA VERSUS ACTIVEX 104
8.7.1. Signed Java Applets 105
8.7.2. Plug-ins 105
8.7.3. Final Comments 106
8.8. REMARKS 106
Chapter 9 109
Conclusions 109
9.1. E-BUSINESS 110
9.1. COMMERCIAL OFF-THE-SHELF COMPONENTS 110
9.2. OBSTACLES TO SECURITY 112
9.2.1. Software Life Cycle 113
9.2.2. TCP/IP 113
9.2.3. Liability Disclaimers 114
9.3. OPEN PROBLEMS 114
9.4. REMARKS 116
Chapter 10 119
Recommendations 119
10.1. FINE-GRAINED ACCESS CONTROL 120
10.2. SOFTWARE FAULT ISOLATION 121
10.3. PROOF-CARRYING CODE 121
10.4. RECOMMENDATIONS FOR USERS 123
10.5. RECOMMENDATIONS FOR SYSTEMS BUILDERS 124
References 126
Acronyms 131
Appendix A – Hacker 135
Appendix B – Electromagnetic Radiation 137
Appendix C – Trusted Computer Systems 139
Appendix D – Kerberos 141
Appendix E – Pretty Good Privacy 142

List of Figures

Figure 2.1 Security perimeters 20
Figure 3.1 Basic access control model 25
Figure 4.1 Simple authentication protocol 46
Figure 4.2 Protocol with nonce 47
Figure 4.3 Flawed mutual authentication protocol 47
Figure 5.1 Encryption and decryption 57
Figure 7.1 Three-way handshaking sequence 87
Figure 7.2 Host protected by firewall as a relay under attack 92
Figure 7.3 Host protected by firewall as a relay during legitimate connection 93
Figure 7.4 Host protected by firewall as a semi-transparent gateway 93
Figure 7.5 Host protected by firewall as relay during legitimate connection 94

Chapter 1

Introduction

"...security is keeping anyone from doing things you do not want them to do to, with, on, or from your computers or any peripheral devices." [ChBe94:3]

Information-system security can be viewed as a chain of users, computers, processed information, communication facilities and computer networks. Its continued good operation depends on engineering standards, user awareness, trustworthiness of software and hardware components, and is regulated by security policies.

The security of information systems is only as strong as its weakest link and various incidents indicate, that the weakest link is people. Without outsiders[1], the system will be secure (albeit not absolutely secure) if the people who access it are secure. Assuming that all those people were responsible, honest, system literate and security literate, and that backups were made in case of hardware problems, there would be very little need for security [Cobb90:481]. The problem arises when a need for secrecy (confidentiality), integrity, authenticity, non-repudiation, timely service, record keeping and sometimes anonymity has to be fulfilled. Unfortunately, even with high quality personnel assurance for insiders, there will always be malicious outsiders for simple financial, competitive and intelligence reasons. As a consequence, most security solutions will combine technical solutions, physical security and personnel controls.

The purpose of computer networks is to share data and services, to share computer resources in distributed computing, to facilitate communication, research and to exchange the information. Networking used to be an arcane subject of interest only to universities, government research and development institutions, and military installations [GaSp96:537]. Things have changed and today, most computers are connected via the Internet. As network protocols were developed, little attention was paid to security issues because sites allowed onto the network were assumed to be trustworthy. (Indeed, security architecture always starts from an articulated trust model.) However, as commercial organizations joined the interconnected public networks to take advantage of its resources and services, they did not necessarily want to allow external access to all of their own resources and information. Because the networks assumed a central role in the information systems, the interest is growing in network access control mechanisms.

[1] Outsiders and insiders are group of users with different privileges. See 2.4.2.

Hardware and software used in computer systems are prone to design errors of every kind. Prudent engineering and good design standards are beneficial, if not essential, in preventing trouble [AbNe94]. Still, there is more to application security than observation of software life cycle standards. The lack of protected execution environments (provided by operating systems) for processes limits what applications can do to protect themselves against users and against other applications.

The evolution of computing and communication facilities has been accompanied by increased demand for security mechanisms. However, the development of computing and communication technologies has outpaced the ability to secure them. Information-systems security has to meet the evolving needs of securing private or sensitive data, while maintaining the benefits of connection to a worldwide information system [HBHo95:23.4].

Security of information systems is an international issue because information frequently crosses international boundaries. These systems play an increasingly significant role in a multitude of activities, including national economies, international trade, government and business operations, health care, energy, transport, communications and education. While growing use of the interconnected computer systems provides many benefits, it has also shown up a widening gap between the need to protect systems and the degree of protection currently in place. International policies address the issues in both public and private sectors and apply to all information systems.

1.1. Overview

The focus of this study is on what technical solutions exist, what works well and what is needed. (This should not be misunderstood as a solution to most of the current vulnerabilities.) Without doubt, the physical and personnel controls are essential to any security system. However, these mechanisms kept pace quite well with the evolution of information systems and are covered in detail in [HBHo95:12.1-13.29], [GaSp96] and [Cobb90].

The technical controls have not done as well and with continuously more sophisticated attackers[1], these mechanisms become more and more vulnerable.

[1] Attackers term in this thesis is synonymous with adversaries, interceptors, interlopers, intruders, opponents or enemy.

The following topics in information-systems security are identified as requiring the most attention:

- Access Control Policies
- Identification and Authentication
- Cryptography and Public-key Infrastructure
- Network Access Control
- Denial of Service
- Foreign Code

Chapter 1 points out major information security problems, describes the contributions of this dissertation and introduces terminology used throughout this thesis.
Chapter 2 provides a background on information security: what needs to be protected, what are the major threats and the role of technical and, especially, non-technical controls in secure information system. This is the last time we discuss non-technical controls at any length.
Chapters 3-8 expand on the most important topics (as identified above) in information security; chapter 3 is about access control, chapter 4 is on identification and authentication, chapter 5 describes cryptography and public key infrastructure, chapter 6 depicts network access control, chapter 7 is on denial of service and chapter 8 portrays mobile code. These chapters evaluate processes, methodologies, protocols and applications that are available today to protect information systems. You will find out what works well, what you have to watch for and what can be done to protect the information from unauthorized disclosure.
Chapter 9 concludes this thesis by highlighting unresolved information security issues and discussing of the implications of this research.
Chapter 10 overviews emerging security technologies, provides recommendations to users and organizations on how to secure their information and offers suggestions for future work.

The conclusions at the end of each chapter, in the Remarks section, provide critical analysis of the existing solutions to information security problems, listings of the most urgent areas that need remedies and suggestions on the future work and research.

References in this thesis are placed within brackets at the end of the referenced passage. The reference starts with four letters that identify the author(s), followed by a two-digit number indicating the year, a colon, and specific page number(s). When authors share the same last name,

then the last letter indicates the author's first name. If the publication year is not available (some Web documents), the number following the author (or organization) is just a counter.

At the end of this thesis, you will find the complete list of references, which inspired this dissertation. The list includes books, journals, government standards, newspapers and the Web. For the Web documents, URLs are provided.

Following the references are appendices on hackers, electromagnetic radiation, trusted computer systems, Kerberos and Pretty Good Privacy.

1.2. Contributions

Computer security technologies, terminologies and definitions have been recklessly used in academic and industrial environment to describe a broad set of problems and solutions when objectives have not been properly articulated. Providing a nomenclature for these terminologies is one of the goals of this dissertation.

There is no computer system that is absolutely secure and probably never will be. A wholly encrypted computer system, surrounded by armed guards, coupled with effective personal assurance and technical controls as important peripherals, is perhaps the best approximation. Since absolute security is unfeasible (at least in the foreseeable future), an alternative is to identify insecurities and move them to less exposed and less vulnerable parts of a system. Building such a secure information system is costly, complex and not a well-understood process. The reader will find in this study an advice on how to construct a cost-effective, secure (in the specific context) information system.

This thesis is built around six areas identified at the beginning of this chapter. The assessment of the (existing) technical solutions in these areas is fundamental in promoting and developing secure information systems. This study examines the market of security products, evaluates available technical controls and makes recommendations on where and what kinds of research are needed. We will attempt to spot technology trends and evaluate the ones that have or are very likely to have a significant impact on information security. However, we will not give recommendations on a particular product or application: because of the rapid changes in information technologies, these products will likely be obsolete, in a matter of years.

In short, this dissertation will strive to achieve the following:

1. Provide a relatively simple and coherent taxonomy without trying to be utterly comprehensive.
2. Provide an overview of computer security to help readers understand their computer security needs and develop a sound approach to the selection of appropriate security controls.
3. Systematically indicate where current security solutions are sufficient and implemented and where they are insufficient. Examine technical control mechanisms commonly employed in the six identified areas (see the beginning of this chapter) and make recommendations on where extra research is needed.

1.3. Important Terminology

It is important for the reader to become familiar with key definitions and terminologies used throughout this thesis.

Attack is a means of exploiting some vulnerability in a system.

Availability is an assurance that authorized users can access and use information in the authorized manner at the authorized times. Availability attacks aim to prevent the temporary or permanent access to the resource or information from authorized users. Information availability is one of the major components of information security, the others being confidentiality, authenticity, integrity and validity. See chapter 7 for more details.

Authentication is a key part of any scheme for preventing unauthorized activity. In a network containing programmable elements, authentication is an essential ingredient for protecting those elements from performing illegitimate actions requested by attackers. User authentication is a process designed to verify the truth of user's claimed identity. Methods include passwords, tokens and biometrics. Chapter 4 describes authentication in more detail.

Authenticity is a prevention of incomplete or inaccurate but authorized modifications to information. Authenticity is concerned solely with use of the system in the authorized manner. It includes an element of timeliness, in that adequate completeness requires the system to be

updated before decisions are made on the basis of stored information. It also addresses some aspects of illegitimate usage, in that fraudulent transactions could be input by an authorized user in the authorized manner without breaching integrity. Authenticity, like integrity is a property of the information itself.

Ciphertext in cryptography refers to the unintelligible text that results from encrypting original text. Sometimes called *codetext*, *cryptotext*, or *cipher*. Refer to chapter 5 for more details on cryptology.

Computer system refers to the entire spectrum of information technology, including application and support systems.

Computer security is a prevention of unauthorized access or unauthorized use of computers and networks by the attackers. Computer security is not a goal but a means toward a goal, namely: information security [ChBe94:4].

Confidentiality is a prevention of the accidental or intentional disclosure of information to unauthorized persons (without the permission of the owner). Confidentiality is one of the principal components of information security. Note that after a loss of confidentiality, the information is unchanged and it would be impossible to determine by any means of inspection that confidentiality had indeed been lost.

Cryptoanalysis is the study of methods for decrypting the *ciphertext* without knowledge of the involved key. It is a science of breaking ciphers.

Cryptoanalyst is a practitioner of cryptoanalysis, whose goal is to break *ciphertext*.

Cryptography is principles and methods for rendering information unintelligible and for restoring encrypted information to intelligible form. When applied to text, encryption is the transformation of plaintext into ciphertext and decryption the transformation of ciphertext into plaintext.

Cryptology is a discipline comprising *cryptography* and *cryptoanalysis*.

Data is a set of recorded symbols. These symbols are normally a formalized representation of facts, numbers and meanings that are stored or recorded on some medium. Such recording may be of very limited duration, e.g., being stored in random access memory, buffered in a communications system or being displayed on a computer screen. Data represents information [Goll99:10].

Data authentication assures that data comes from the verified source.

Discretionary Access Control is a method of restricting access to object based on the identity of subjects (or groups). The controls are discretionary in the sense that a subject with given access permission is capable of passing that permission on to any other subject, unless restrained by mandatory access control. See chapter 3 for more details.

Information is data with its assigned meaning [Goll99:11]. This meaning is assigned to or derived from the data by applying formal conventions to it. Because the definition of data implies storage, the definition of information implies an element of storage too.

Information Security is a combination of confidentiality, authenticity, integrity, availability and validity. Although the scope of this definition is primarily concerned with technology-related aspects, the definition of information security is equally valid for all media on which information may be held (e.g., magnetic or paper). It comprises both computer and communications security.

Integrity is a prevention of the unauthorized modifications to information. Integrity ensures that information is only added to, altered or deleted by authorized users.

Least privilege is a principle that each subject be granted the least (most restrictive) set of privileges needed for the accomplishment of authorized tasks. The application of this principle limits the damage that can result from accident, error, or unauthorized use of information system.

Mandatory Access Control is a method of restricting access to objects based on the sensitivity of the information contained in the objects and the formal authorization (clearance) of subjects to access information of such sensitivity. See chapter 3 for more details.
Owner is a user with granted privileges with respect to security attributes of specific subjects and objects.

Plaintext (also referred to as *cleartext*) is intelligible data with available semantic content.

Privacy is the ability of an individual or organization to control the collection, storage, sharing and dissemination of personal and organizational information. It is also a right to insist on adequate security of information and to define authorized users of such a system. Since the concept of privacy depends on legislation, its definition cannot be very precise.

Public key cryptography (*asymmetric* cryptography) is a cryptography using two matched keys in which a single private key is not shared by a pair of users. Instead, users have key pairs consisting of a matched private and public key. See chapter 5 for more details on cryptography.

Risk is a probability that a particular threat will exploit a particular vulnerability of the system, usually involving an assessed balance between threat and vulnerability.

Secret key cryptography (*symmetric* or *conventional* cryptography) is a cryptography based on single key. It uses the same secret key for encryption and decryption.

Security perimeter is a boundary between a hostile external network and a trusted internal network.

Spoofing (impersonating, masquerading, mimicking) is an attempt to gain access to a system by posing as an authorized user.

Stand-alone system is a system that is physically and electrically isolated from all other systems.

Subject is an entity, generally in the form of a process or device, that causes information flow among objects or changes the system state.

Threat is a circumstance, an event or an adversary with the potential to cause harm to a system by exploiting a vulnerability. This harm could be destruction, disclosure, modification of data and/or denial of service.

Trusted computing base (TCB) is a complete set of protection mechanisms within a computer system, including hardware, firmware and software; the combination of which is responsible for

enforcing a security policy. The ability of a TCB to correctly enforce a security policy depends solely on the mechanisms within the TCB and on the accurate implementation of the security policy by the administrators in charge.

User is any person who interacts with information system either directly or indirectly (through the network). This definition excludes operators, system programmer, system security officers and other support personnel.

Validity is a total accuracy and completeness of information. Integrity and authenticity are concerned with the point of view of the information owner. However, the users who have only read access to a particular information system have no direct influence on integrity or authenticity but are extremely concerned with validity. The difference between validity and the sum of integrity and authenticity arises when the priorities of the information owner and the user, diverge. For example, in the military and diplomatic spheres, information may well be as valid as its owner requires while being deliberately inaccurate or incomplete, i.e., disinformation.

Vulnerability is an error or a weakness in the design, implementation or operation of a system.

1.4. Remarks

Computer security depends on the complexity management of networked information systems, including their users (people). When complexity management fails, important vulnerabilities of our computer systems can go undetected for long periods of time. The most important sources of complexity are the elaborate interactions among subsystems, and between subsystems and users, designers, programmers, and managers. Many of the subtleties of security issues relate to unexpected interactions between subsystems, individuals and the system, and to formulating access and authorization policies that provide security yet allow employees to do their job. No information system can be secured without properly addressing these security issues.

It should not come as a surprise that a major threat to information systems are users themselves and without their education, there is little technical controls can do. The next chapter expands on threats to information security and classifies attackers based on two criteria: level of skills and available resources, and level of privilege to access information systems.

Still, it is probably prudent to apportion blame for successful penetrations somewhat evenly between systems and humans. Humans sometimes do very stupid things, e.g., maintenance engineers temporarily turning off security controls during routine maintenance. Yet countless intrusions were made possible because of very subtle bugs or flaws in computer systems that were not at all evident. As the complexity of modern systems grows, there will be increasingly many system vulnerabilities that will remain extremely well hidden for long periods of time. Until such time as there is a breakthrough in complexity management, we will have to build security while acknowledging the continued existence of vulnerabilities in our most important applications.

Prudent (software and hardware) engineering, observation of software life cycle procedures and education of users are essential in solving information security problems.

Technology is inherently a double-edged sword; it works both for the benefits of legitimate users and provides the potential for new criminal activities. The simplest example is the growth of features in popular applications. As part of the rivalry between Netscape and Internet Explorer, the number of features offered by a Web browser has increased alarmingly. But these rich features cannot come except at the price of unmanageable complexity of the browser software. We are condemned to use applications that it is quite unreasonable to trust, even if we think highly of their authors. More research is needed to understand how to build information systems that are secure in a given context. One major promising security mechanism is confining untrusted applications to minimal protection domains.

Information security terminologies have been used recklessly. The end result is that vendors and consumers of the security technologies use the same vocabulary but mean different security features.

Chapter 2

Computer Security Fundamentals

"The impact of computer risks, exposures, and losses can include financial and personnel losses, loss of reputation and client base, inability to function in a timely and effective manner, inability to grow, and violation of government laws and regulations." [HBHo95:9.3]

As computers and networks process increasingly sensitive and mission-critical information, it becomes increasingly important to protect them from catastrophe, from incompetents, vandals and hackers[1] (see Appendix A). Many people, referred to as hackers, are adept at breaking into networks and hosts [Anon98]. The threat from malicious organized hackers (e.g., foreign intelligence services or organized crime) is, of course, a far more serious threat than "vanilla" hackers, who often have no deep motivation to do harm. Sometimes, users with good intent might accidentally expose corporate data or services from within a network. Regardless of the motivation or the reason for the information security breach, the end result to the information owner is the same: it is expensive.

Many commercial banks contain trade secrets and other information worth many millions (sometimes billions) of dollars to competitors. There are several cases of such data being stolen [Brun00] & [Bent00]. A variety of nuclear-weapon systems are under computer control. Since computers assume such vital functions in our society and industry, the data must be protected against unauthorized access. The data must be protected from unintentional negligence, program errors, transmission errors, catastrophes, as well as from vandalism, sabotage and curiosity. Thus, cyber security borders on general fault tolerance.

2.1. Evolution of Security Needs

In early computer systems, physical controls were an effective means to control access to data and computer resources because the systems were isolated and single-user. The advent of multitasking and time-sharing allowed users to share programs and data. It also created a need for

[1] The hackers are either viewed as computer enthusiasts who have an ardent interest in learning about computer systems and using them in innovative ways, or malicious hackers who deliberately crash systems and delete files. Refer to [Anon98] on distinction between (good) hackers and (malicious) crackers.

mechanisms to control this sharing. As computers were connected to the network, the access controls to shared resources became even more important and complex. The stakes were raised once again with the move to distributed computing and the Internet connectivity. While all-to-all connectivity provides flexible sharing of resources, it also provides ready, remote access for attackers from around the world.

In industry, there is often more concern with the confidentiality of commercial data than with the privacy of individuals. Many administrators ignore individual privacy, since they are mostly concerned with economical aspects, particularly operating costs [SpLa89:4].
Privacy refers to the rights of individuals and organizations to determine for themselves when, how and to what extent information about them is to be disclosed to others. This sounds straightforward. Actually, balancing individual privacy rights against the rights of society is a nontrivial issue of political philosophy. Rachel [EWSh97:69-76] explains the importance of privacy in situations where a person has something to hide and in situations where a person has done nothing wrong. Kenny [Kenn85] describes legal and social constraints that have emerged to direct what may and may not be done with personal data. Another very interesting publication on privacy is due to Garfinkel [Garf00]. The author shows the threats to freedom, which are becoming apparent in our Internet enabled world, through the volume of databases, which are being created beyond the control of the citizen.

Privacy cannot be assured without security and many of the technical methods for achieving privacy also apply to security. However, privacy and security are not the same subjects. [Kenn85:179]

2.2. Information is a Strategic Resource

"Information, for its own sake or for the price it brings, is an eagerly sought after commodity." [HBHo95:11.17]

Information is one of the most significant assets [Dend99:23-25] of an organization. Survival of an organization may well depend on the quality and integrity of its information. In some cases immensely valuable information, such as machine specifications or geological prospecting data, can reside in a single tape or disk. This information is an attractive target for the attackers and

computers are handy tools in cyber crime. Computers enable massive theft of industrial information because they concentrate information and because networks allow such information to be moved rapidly.

International espionage used to concentrate mainly on government and military secrets. This still continues in full force. However, today large sums of money are being spent to obtain industrial secrets [Simo96:11-14]. The cost of initiating and developing new technological process is very expensive and it is obviously cheaper to steal knowledge from those who have made the initial investment in physical and intellectual capital. As international espionage takes place between entirely friendly nations, nations are on the continuum between "friend" and "enemy".

Access to information is an empowerment [Kenn85:133] and many organizations thrive on collecting, classifying and managing access to information. While people readily furnish information for collection and distribution, they seek to protect their privacy and confidentiality. This seems self-contradictory, but the person providing information may agree to do it for the specific usage or wishes to remain anonymous. With the rapid growth of computer technologies, the collection of information can be also done without the person's consent or knowledge.

2.3. Information Security Vulnerabilities

The media tend to exaggerate threats to information security. While (most of) their reports are true, they have a tendency to sensationalize. Only by thorough analysis of information system security vulnerabilities can you find practical solutions to the real threats.

Here is a list of major vulnerabilities or weaknesses or problems. If any one of them is neglected, it becomes a serious security loophole. The list is important because often security managers pay attention to most but not all of these vulnerabilities. For real security, one must have solutions to each and every one of these problems. The vulnerabilities are:

- Physical security holes
- Flawed trust model and lack of a suitable security policy
- Faulty configuration
- Software bugs and hardware errors
- Badly implemented cryptographic protocols
- Security illiteracy and weak personnel assurance

2.3.1. Physical Security Holes

"The potential problem is caused by giving unauthorized persons physical access to the machine, where this might allow them to perform things that they should not be able to do." [GaSp96:357]

The security system must be planned for an entire office building or plant, rather than just a computer facility. It is desirable to plan the security system in an integrated fashion and to avoid isolated gadgets. You should understand that electronic devices themselves do not buy security and they are only part of such protection. The first step is to completely define the protection that is needed from fire, flood and intruders. These and other threats such as dust, explosion, earthquakes, insects, vibration, food and drinks are described in [GaSp96:362-368]. The total security requirements including all aspects of security need to be assessed and an overall system designed that will maximize the protection with a given budget.

2.3.2. Trust Model and Security Policy

"Trust is the most important quality in computer security." [GaSp96:799]

Trust is a belief that an entity behaves in a certain way. The trust requirements of a system form the system's trust model. A trust model describes the trust relationship between all the entities in an information system. Examples include whether an operating system trusts an application, whether an organization trusts an employee, and countless other things. The trust model is the set of assumptions used in implementing a security policy. Not articulating it is a recipe for disaster, in general and particularly when the trust model evolves over time. The trust model identifies a set of components that are trusted not to violate certain security rules.

Closely related to the concept of trust is the concept of policy. A security policy is a manifestation of rules, procedures and practices that regulate how sensitive information and other resources are managed. Security policy should capture the security requirements of an organization. Implementation of the policy, guidelines and procedures should be planned carefully to ensure maximum effectiveness with minimum disruption of normal activities [HBHo95:2.14]. Security should not merely be superimposed but instead should be carefully fitted into the organizational structure. Unfortunately, information security is often an afterthought to the organization formation and as a result, security policy is defined in a hazardous way.

2.3.3. Faulty Configuration

Often, software applications that are shipped from manufacturers come with settings that do not provide the required security. The vast majority of applications are pre-configured by default to insecure settings, to fulfill their primary functional objectives with minimal hassle for the user. Unfortunately, rarely is there ever a one-size-fits-all security policy. Rather security features are configured for the lowest common denominator in security requirements. For example Unix systems are often shipped with the guest/demo default accounts installed out of the box and insecurely configured Web servers. To configure safely an operating system (OS) or other Web application is never an easy task. Every administrator of the information system should run a battery of tests, using special tools [FaSp94] to identify configuration errors.

2.3.4. Software and Hardware Errors

Computer hardware and software components are occasionally going to fail. How often will depend upon the degree of reliability that has been engineered into the system. To reduce the incidence of errors during software development and production, comprehensive quality control is needed. Furthermore, standards for development, documentation, modification, can help reduce software errors and facilitate corrections. Information security has to be an integral part of an overall application design [SpLa89:2]. It has to be considered in the context of the information system, which includes physical location, users, (sensitivity of) data and application objectives.

The designer of the security procedures must consider all the possible categories of failure and decide what should happen when each occurs. The devised steps must ensure that no important data are lost or accidentally entered twice. There should exist procedures for reconstructing files in case the file records become permanently damaged. To ensure that the correct procedures are followed, it is desirable to have failure drills periodically [HBHo95:7.1-7.35].

2.3.5. Cryptographic Protocols

Nobody can guarantee to deliver a bug-free software application. Cryptographic protocols are no exception and all concerns from the previous section apply here as well. However, implementation flaws in these protocols could have grave consequences, because people employing these protocols might have entrusted confidential information. Abadi and Needham [AbNe94] provide guidelines for designing cryptographic protocols. Another problem is

proprietary encryption algorithms. These secret algorithms are regularly published and broken. It is curious that the marketplace never learns that proprietary secret algorithms are a bad idea. But companies and industries continue to choose proprietary algorithms over public, free alternatives.

2.3.6. Education and Personnel Assurance

Many would like to buy a security product, install it and rest assured that their information is secure. Unfortunately, security is not a product; it is a process that has to be followed on the ongoing basis. A big part of this process will involve education of users to keep them involved in maintaining security. The user education is probably one of the most important [Simo96:213-214] and often neglected, factors in the system security.

Top management should support security awareness and training programs [Cobb90:484-485]. First, the management should learn the security importance, participate in the design of security policies, then set the proper example and tone in following the security procedures. All users should be taught the rationale behind the particular security policy for their organization by showing the examples of natural or man-made disasters and contrast between the resumption of essential business operations in those organizations that had a viable contingency plan and those that did not. They should be taught which procedures to follow in case of failure and security breach, to minimize the extent of the damage.

Personnel security is everything involving people: hiring (screening), training, monitoring and handling their departure (especially firing) [GaSp96:389-395] and [HBHo95:4.1-4.20]. To achieve high security, you need personnel with an elaborate security clearance and known to be loyal.

2.4. Category of Attacker

"The easiest way into a computer is usually the front door, which is to say the login command." [ChBe94:159]

The costs of measures that could protect a system from people who might harm it or misuse it ranges from very small to astronomical. To select appropriate measures it is necessary to determine who is your enemy. Categories of unauthorized penetration vary from accidental

disclosure of the secure information, to casual entry by skilled technicians, to attacks by professional cyber thieves and to attacks by organizations with massive funds. To protect against all categories of intruders, except for accidental disclosure of information, cryptography should be employed. If the concern is the enemy from the last two categories, the strong cryptography coupled with administrative and law controls, must be used. The attacks by organizations with massive funds can involve huge expenses and the defense against them requires an equivalent level of expenditure and capability. Your highly loyal staff should be able to protect the data from anyone other than the National Security Agency (NSA) or an organization with similar funds and skills. Even then, an ingenious and persistent criminal may slip through the defenses [Stol89].

2.4.1. Professionals and Amateurs

There are two groups of attackers on information systems: professionals and amateurs. Usually the amateurs are less skillful, less experienced and considered less dangerous since they are not organized. The skill level in the amateurs group can vary from script kids to semi-pros. Another thing that distinguishes those two groups is the amount of money they have. The professionals are usually well funded. The third distinction is their respective objectives e.g., exposure of vulnerabilities to force system patches versus, say, espionage or extortion.

One of the attacks used by well-equipped opponents involves the remote detection of the electromagnetic signals from computer (see Appendix B). This attack can be thwarted by properly shielding all the computer equipment and network cabling so that it does not emit these signals[1].

2.4.2. Insiders and Outsiders

For some time there has been a discussion on who is more dangerous to the information system: insiders or outsiders. According to Lewis, [Lewi98] between 70% and 80% of security incidents, are caused by insiders. These incidents ranged from sabotage, fraud and theft of proprietary information to unauthorized snooping in an email or storing pornography on a company computer. Insiders are generally regarded as the greatest threat to organization's information system [Dend99:131]. Should we decide to concentrate on insiders to increase information

[1] U.S. government standards for shielding technology are known as TEMPEST. There are hardware vendors who supply Tempest shielding commercially, although it may be subject to some kind of government licensing.

system security, first we have to find out who belongs to insiders, how an outsider becomes an insider and whether there is a third category.

We will use the following definition for an insider:
An *insider* is an individual who is a member of an organization (bank, automaker, security consultants), on permanent or temporary basis (full-time, contractor) and has a certain level of privileges to access information system resources that are not otherwise publicly available (information on the Web).

There are subtle similarities and differences between industry and military insiders.

There are problems with reports on security incidents. When insurance claims are involved, the damage is often overestimated. On the other hand damage is often underreported in the financial community to avoid loss of customer confidence. We need comprehensive reporting mechanisms for security incidents in place before a more accurate estimate on damages can be made. Another problem with reporting process is that it seems to inflate the relative number of insider incidents. It is because sophisticated outsiders leave minimal or no traces and force the suspecting party to go to great length to convince authorities that such an attack is under way. The insider attacks tend to be prosecuted more energetically and gain more publicity.

The most dangerous insiders are disgruntled system administrators [Simo96:16]. Since they have both the knowledge and the authority to alter data, cover their tracks by modifying audit logs and modify audit trails to direct suspicion at other individuals.

2.5. Building a Secure System

The critical issue is to ensure that information is entered or extracted in strictly controlled and monitored fashion. To build such a system, you must view all the security mechanisms employed in the security perimeters[1] in the entirety. If you put too much emphasis on any of the mechanisms, you may actually weaken the system, because users will always find ways to circumvent the security to do their job properly [Goll99:22].

[1] Please refer to 2.5.3. for more details on security perimeters.

However good the protective measures are, there will always be some means of damaging computer or its data. All you can accomplish is to build a secure computer system that balances the value of the stored information against the cost of providing security for such a system. See [HBHo95:6.8] for typical profile of computer thief.

When data is expensive to replace or sensitive then an audit becomes a necessity. Audit trails provide a way to assign responsibility and accountability for events that take place in the information system. The audit trail allows to review of the events that occur in the system and can detect and discourage security violations. Besides recording of the events, an audit trail must provide a support for (internal and external) auditing. The authorized agents must be able to access audit trails by a secure means. Because the complexity of the information systems increases, the expertise of the auditors needs to be augmented too [HBHo95:9.13].

2.5.1. Security through Obscurity

There is a concept of need to know in traditional security, derived mostly from military intelligence. It is believed that to reduce the likelihood of successful penetration, you should restrict knowledge of various aspects of the system to persons with a need to know. The technical manuals and documents describing the system should be locked away. The contents of the data files should be revealed only to those with a need to know. However, this concept might only be appropriate for the military, where inferential security is of concern [GaSp96:40]. Your system should rely upon mechanisms that are inherently strong, even if the attacker (and anyone else) knows about them. Keeping the system manuals away from the users will not stop a determined attacker, since such documentation might be widely available elsewhere.

2.5.2. Suitable Security Policy

The main purpose of a computer security policy is to inform users, staff and managers of their mandatory requirements for protecting information assets. The policy should specify the mechanisms through which these requirements can be met. Another purpose is to provide a baseline from which to acquire, configure and audit computer systems and networks for compliance with the policy. Therefore, an attempt to use a set of security tools in the absence of at least an implied security policy is meaningless.

The main areas that the security policy must cover are the following:

- Minimize the probability of the security incident
- Minimize the damage if it does happen
- Design a recovery method from the damage

2.5.3. Security Perimeters

Security is often visualized as a ring structure, where every ring has appropriate protection mechanisms, imposing successive barriers to penetration of secure areas (see Figure 2.1). These protection mechanisms define a security perimeter. The parts of the system that lie within the perimeter can be used to subvert its protection mechanisms. The attacks on the parts that lie outside of the perimeter do not affect its protection mechanisms. Of course, the attacks to the layers below affect the protection mechanisms in the upper layers [Goll99:14-15]. Organizations tend to array their defenses around the perimeter of their computing network and rely on deterrence mechanisms, such as audits, to discourage insider attacks.
Unfortunately, fine-grained access control is absent inside these perimeters and technical controls on administrators are minimal.

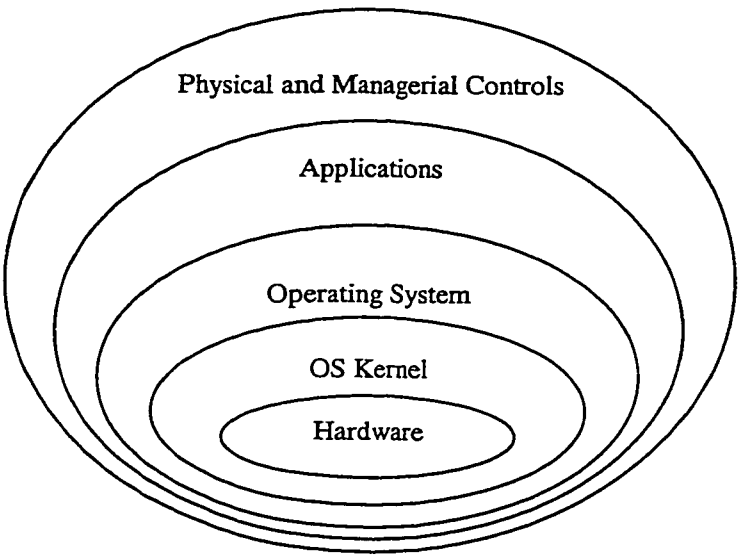

Figure 2.1 Security perimeters

Intrusion detection systems are promoted for combating both insider threats, as well as for detecting outsider attacks that have breached perimeter defenses. These systems collect data on

computer and network usage, apply pattern matching or heuristics and trigger alarms if they detect an improper activity. When these systems are used toward insiders, they proved to be inefficient. The amount of data that must be collected imposes performance penalty and raises concerns about improper workplace surveillance. Besides, the assumption that users display fairly constant behavior patterns is generally invalidated during emergencies, exactly when these alarms are least desirable. Proficient intruders can alter their usage patterns, so activities that would normally trigger alarm are now treated as normal.

2.5.4. Physical, Personnel and Legal Controls

"You may have the best encryption and security tools in place... However, if you cheerfully hire an industrial spy as your system administrator, and she walks off with your disk drives, those other fancy defenses aren't much help." [GaSp96:357]

Personnel security is inherent in any information system, because some people have to be trusted with respect to their authorized interactions with the system. These people configure the system, manage its operations and ultimately initiate authentication of other users to the system. Similarly, physical security is needed to thwart theft or destruction of data and computer devices. The physical, personnel and technical security are functions of the environment in which system operates. Individuals handling classified information will have thorough background checking. In contrast, most of information technology employers will apply less stringent screening for their staff. Likewise an amount of physical security needed by information systems of a stock exchange is greater than that of a typical commercial system. Legal issues play an important role in computer security, since the company is liable for violation of its legal commitments in the security area.

Unmistakably, physical, personnel and legal controls are essential components of any secure information system, however, this thesis is focused on technical solutions. For more information on physical and personnel controls the reader is referred to [HBHo95:12.1-13.29], [GaSp96] and [Cobb90].

2.6. Remarks

The computer security problem has changed enormously as we have moved from centralized to networked computers. Many solutions, which were robust in a centralized context, have yet to fully generalize in the networked environment. While all-to-all connectivity provides flexible sharing of resources, it also provides ready, remote access for attackers from around the world. Even simple things like hiding hashed passwords became complicated in local area networks. The Internet means that countless hosts now offer complicated services to each other and depend on complicated protocols.

Networks eliminate all hope of centralized control, unless you can magically ensure that your network is secure, and only trusted machines are connected to it. In practice, just about the only reasonable trust model is to assume that the network is totally under the control of the adversary.

Why is this? There are several reasons: It is hard to ensure physical security of network links. Infrastructure components (routers, name servers, etc.) introduce new single points of failure. Scaling at the level of the Internet makes complexity unmanageable. Centralized security mechanisms generalize imperfectly to heterogeneous distributed systems.

Much of the shift in network usage is being fueled by new ways to use computers and by their increasing ubiquity, power and diverse applications. At the leading edge of the business world, a new computing paradigm is emerging, one that de-emphasizes simple linkage of computers in favor of distributed computing. This anytime and anywhere connection to information is a powerful way of resource sharing but it also puts a lot of pressure on providing information security. The connectivity of computers complicates security because the growth of types of network traffic and the potential sources of network traffic severely overloads our attempt to filter potentially adversarial inputs.

Security is a function of an environment that needs its protection. Comprehensive security should balance value of information and costs to protect it. Information system security is a complex activity that starts with the design of the security models and is an ongoing process requiring maintenance and education. The objective of security education must be to obtain security by consent. It should never be imposed without reason. Security must be seen to be reasonable and necessary, and the procedures must be designed so that they do not cause unnecessary inconvenience.

In practice the degree of protection taken and hence its cost will depend upon the assessment of the potential intruders. This in turn will depend upon the sensitivity of the data and its possible value to an attacker, vulnerability assessment, threat assessment, cost model of information loss or degradation and cost model of system loss or degradation. The risk analysis approach to information security while appropriate for the industry is unsuitable for the military. It is impossible to calculate the cost of human life, security of the country or the planet. When confronted with these kinds of threats, no expenditure is excessive.

Need to know or security through obscurity concept, derived mostly from military intelligence, is not applicable in modern information systems. It is just a bad idea. Why is this? Essentially, we need help in discovering the vulnerabilities in our systems.

There are a variety of aspects to make a system secure and neglecting any one of them represents a security loophole. We would like to emphasize the necessity to build the cryptographic protocols right. Since people employing them might have entrusted confidential information, the implementation flaws in these protocols could have grave consequences. Another problem are proprietary crypto algorithms. You should use only encryption algorithms that have been openly published and withstood years of attack. Do not create your own encryption algorithms unless you are an expert in cryptology. Will we ever learn?

To achieve high security, you need personnel with an elaborate security clearance and known to be loyal. Engendering loyal, well-informed, competent and contented workers is probably the best security measure you can take.

To select appropriate protective measures it is necessary to know your enemy. The intrusions can range from accidental disclosure of information, to attacks by organization with massive funds. The attackers can be grouped into professionals and amateurs. Professionals and amateurs have different skill levels, funds and objectives. We can also divide the attackers into insiders and outsiders. Insiders are generally regarded as the greatest threat to organization's information system. Technical defenses against insiders tend to be expensive and largely ineffective. The most practical solution is to know the people who have significant authority on the system and to maintain their loyalty to the organization.

The secure information system should encompass physical, personnel, technical and legal controls; it should also use cost-effective security mechanisms. Personnel security is inherent in any information system, because some people have to be trusted with respect to their interactions with the system. Legal issues play an important role in computer security, since the company is liable for violation of its legal commitments in the security area. These controls are interdependent. Physical security is not irrelevant when designing system techniques. The question of what data volumes should be stored off the premises probably affects most of the security controls.

Chapter 3

Access Control

"Authorization is the granting of rights, by the owner or controller of a resource, for others to access that resource. Access control is a means of enforcing authorization." [Ford94:149]

Access control ensures that users access only those resources[1] and services that they are entitled to access on network, operating system and application levels. They are system-based mechanisms used to specify who or what (process) has access to a specific system resource and the type of access that is permitted. Access controls can enforce the principle of least privilege, which refers to the granting to users of only those accesses minimally required to perform their duties.

Often, when describing access controls mechanisms, we refer to (active) subjects and (passive) objects. Users and processes are typical subjects, while files and servers are examples of objects. However, this distinction is only meant to differentiate between active and passive characteristics of an entity. In certain circumstances an entity might be a subject, while in others, it might be an object. The subject issues requests to reference monitor (see Figure 3.1), which either grants or denies the access to the object [Goll99:31].

Figure 3.1 Basic access control model

Access controls can be grouped into physical and logical controls. Physical access controls restrict the entry and exit of personnel (and often equipment and media) from an area, such as an office building, suite, data center, or room containing a LAN server. The controls over physical access to the elements of a system can include controlled areas, barriers that isolate each area,

[1] Resource is interpreted broadly to indicate information resources, processing resources, communication resources and physical resources.

entry points in the barriers, and screening measures at each of the entry points. In addition, staff members who work in a restricted area serve an important role in providing physical security, as they can be trained to challenge people they do not recognize. As previously mentioned, physical security is not the focus of this dissertation. Hence, further discussion in this chapter refers to logical access controls. Logical access controls provide a technical means of controlling what information users can utilize, the programs they can run and the modifications they can make.

Very often, access controls are described in terms of policies they support and their effectiveness is judged by how well they support those policies. Access control policies model notions of authorization that exists in a physical world. The effectiveness of access controls is based on two premises: proper user identification and integrity of user access rights. The first premise is met through user authentication at login. The second is achieved by protecting user rights and permissions from unauthorized modifications.

There are two basic types of access control policies: discretionary access control and mandatory access control. Discretionary or identity-based[1] access controls regulate how users may delegate access permissions or make information accessible to other users. Most of access control mechanisms that are implemented and deployed enforce discretionary access control policies. Individual users, group of users are identified as subjects, whereas computers, networks, files and processes are identified as objects. Discretionary access controls appear to mimic physical world policies of authorization, but there are certain differences. For instance, transitive sharing of data involving intermediary subjects can subvert the intent of discretionary access policies by allowing a subject to learn (indirectly) about the object, even though the policy forbids (direct) access to that object by the subject.

Mandatory or rule-based access controls also regulate how subjects can access the objects, but now only security administrator, instead of individual users, specify what accesses are permitted. Mandatory access policies are typically formulated for labeled objects and normally the policies are intended to regulate information flow from one object to another. In a military style of classifications, subjects are assigned clearances and simple rules dictate the clearance needed by the subject to access the object that has been attributed a certain label.

[1] The OSI Security Architecture (ISO/IEC 7498-2) uses identity-based and rule-based terminology instead of discretionary and mandatory policies.

While mandatory access controls can prevent Trojan horse attacks, discretionary access controls cannot. Discretionary access controls are vulnerable to Trojan horse attacks because application executing on behalf of a user inherits that user's privilege. Mandatory access controls block such attacks by limiting the access of all programs in a manner that cannot be circumvented by users.

There are three types of mechanisms used to enforce the security policies:

- Access Control Lists (ACL)
- Capabilities
- Security Labels

The difference between ACL and capabilities is the storage for access right to the objects. In case of ACL, the rights are stored with the objects themselves, while capabilities mechanisms stores access rights with the subjects. Security labels are typical mechanisms used by government agencies to implement multi-level security policies. Suppose that you have four linearly ordered security levels (labels): unclassified, confidential, secret to top secret. By assigning objects and subjects appropriate security labels, you can implement mandatory security policies [Goll99:36-37,41-44]. These mechanisms used to be considered as distinct approaches to implement access control. However, modern network access control mechanisms often cannot be simply categorized as one of the three types. More commonly, they mix characteristics of these types. Access control mechanisms have attempted to keep pace with the new ways of resource sharing supported by new generations of computer systems and to fend off a growing number of attacks, to which the systems are subjected. Some of these mechanisms, such as password-based[1], are not suitable for securing modern networks and that is why they were omitted in this dissertation. Others, evolved and found their employment in the operating systems, Message Handling Systems, Directory[2], file transfer and other applications.

3.1. Security Models

Many security policies of practical interest cannot be formulated as discretionary and mandatory access control policies. These controls focus on protecting information from unauthorized access, but they cannot model effects of malicious or erroneous software, nor do they completely address

[1] Password-based access control mechanisms have been used extensively in mainframe computer operating systems.
[2] Commonly known by their International Telecommunication Union (ITU) designator X.500, provides the basis for constructing a multi-purpose distributed directory service.

availability of system resources and services. They are described in an access control model, defined by Trusted Computer System Evaluation Criteria (TCSEC) [DoD85] that has limited expressive power, leaving the model unsuitable for certain applications dependent on access controls. (See Appendix C for more details on TCSEC.) TCSEC was inspired by the best-known form of mandatory access control Bell and LaPadula. [BePa73]

Bell and LaPadula showed that it was possible to give a fairly simple state-based formalization of security in terms of the passive objects (files, directories) held in the computer and the subjects (programs, processes), which act upon them. Having produced an abstract model of the state of a computer system, Bell and LaPadula identified classes of operation, which modified the system state, then specified the security policy in terms of changes to that state. This gave rise to the well-known *simple security* property that a subject cannot read an object above its clearance and the *star (*)* property that a subject may not write to an object below its clearance. These two properties are often referred to as "no write down and no read up". Bell and LaPadula gave both an abstract definition of the model and showed how to interpret the model for real computer system – Multics.

TCSEC model assumes that an organization has static policies with concise characterizations. Since organization's security policies usually change with perceived needs, this hypothesis is problematic. Even the Department of Defense (DoD) policy has numerous exceptions to handle special circumstances. For example, senior political or military officials can downgrade classified information. But the common form of mandatory access controls does not allow non-sensitive objects to be derived from sensitive sources, because it does not associate content with objects, nor does it formalize when declassifying information is safe. Policies involving application specific information cannot be handled either, since such information is not part of TCSEC access control model.

Other models have been proposed to take into account the application involved. The Clark/Wilson model sets rules for maintaining the integrity of data in a commercial environment. The Chinese Wall model by Brewer and Nash, consider dynamic changes of access rights to express rules for separating different organizational activities to satisfy legal structures in the financial world. Other interesting proposals for security models, are described in [SpLa89:41-96]. Today, it is generally agreed that there is no universally applicable model equally suited to all kinds of enterprise.

From the start, there has been a gap between organizational policy and the 1970s view of computing embodied by TCSEC access control model. This computing model had users remotely accessing a shared, central facility through dumb terminal equipment. As computing technology advanced this gap has widened. Processes, for example, are complex entities without clear boundaries. Modern programs likely have their own access controls independent of what is provided by the underlying OS and TCSEC access control models. The policy that does not capture this aspect of computing system is flawed.

Another problem with TCSEC model is that it implies that objects have uniform security levels, which is seldom a case for real objects. This all-or-nothing nature of TCSEC access control model renders it impractical. Designers who implement the model either make the system restrictive, rendering it unusable, or invent escapes, in which case, knowing that the system adheres to the model has limited practical significance. Additionally, TCSEC access control model does not account for defensive measures such as virus scans, approaches to executable content control, or firewalls. The system architect who is bound by the model has no incentive of deploying these technologies.

DoD has recognized some of the problems in building systems that enforce TCSEC access control model. The result of this is DoD Goal Security Architecture (DGSA). Instead of forcing usage of TCSEC model, DGSA supports a broad set of security policies that go beyond the traditional information flow policies. While DGSA encourages the users to employ the latest in object-based, distributed systems and networks, it does not offer any insights about how to achieve an appropriate level of assurance that these policies are correctly implemented.

3.2. DoD Goal Security Architecture

DGSA is oriented toward a range of access controls and integrity policies in an object-oriented, distributed-system environment. The range of security policies to be supported goes far beyond Bell – LaPadula [BePa73] information flow security policy. DGSA offers multiparty authorization, multilevel objects, role-based authorization and variable levels of availability. It embraces commercial off-the-shelf (COTS) products and commercial network resources. Commercial networks can be employed through the use of network security devices, but there is

the matter of achieving availability. If these networks were vulnerable to disruption, then DoD communications traversing these networks would be vulnerable to denial-of-service attacks.

Use of COTS operating systems and applications raises question about how to create multilevel information objects and how to enforce appropriate information flow security, as labeling is not generally supported in such commercial products. Perimeter security devices, such as firewalls and guards, are limited in the granularity at which they can enforce data separation, especially in the absence of labels.

Actually, DGSA can be viewed more as a list of goals than as an architectural specification. Available COTS technology and even research and development prototypes lag far behind what DGSA calls for. Most of the goals require substantial research and some may be unattainable relatively to credible national-level threats.

3.3. Operating System Access Control

Access control mechanisms in the operating system (OS) restrict access to files and other resources (memory, printers, databases, servers) to users holding appropriate privileges. These access controls are used primarily for integrity reasons and not so much for confidentiality. Conceptually, access control mechanisms are divided into two subsystems: a decision subsystem and an enforcement subsystem. The decision subsystem examines the security attributes of objects and processes according to a security policy and decides whether a particular access should be allowed. The enforcement subsystem ensures that the decision cannot be circumvented neither by user nor software. The presence of protection mechanisms in the information system does not guarantee security. If there are flaws in design of OS, process may be able to bypass security mechanisms [Dend82:231-235].

3.3.1. Decision Subsystem

The decision subsystem for discretionary access control usually employs access control lists (ACLs). An ACL is associated with each data object and consists of a list of users, enumerating what accesses to the object each user has. ACLs are popular operating system (OS) access control mechanisms. Two prominent operating systems that are used today are Unix and Windows NT

(Windows 2000). While promoters of Unix (and Linux) claim better security because the longer history, Microsoft argues that NT[1] makes the best of Unix and emerging security technologies. NT was developed in object-oriented manner and security modules are no exception. Each securable object has a security descriptor that designates its security attributes. These attributes recognize the object's owner as well as two ACLs: discretionary access control list and system access control list [Mill98:125]. Designing OS using object-oriented methodology helps us to validate its claims to the security.

ACL can be difficult to administer, because expressing authorization for a large number of users becomes awkward for large number of objects. UNIX systems employ a modified scheme where for each object the owner only specifies object access permissions for the user, for a small number of specified groups of users and for all other users. Windows NT also addresses this administration problem by supporting access permissions for groups. A number of users with similar access rights are placed in groups to facilitate the management of access control policies [Goll99:38].

The decision subsystem for an ACL-based discretionary policy obtains the name of the user on whose behalf a particular process is executing, checks the ACL for an entry containing that user name and grants accesses according to the ACL entry that is found. This has been called a list-oriented approach.

An alternative to ACLs is to associate a list of capabilities with each process. Each capability list names an object along with the kinds of access to that object that the capability holder is permitted [KaLa86]. The decision subsystem, for a capability-based access control mechanism, checks the list of capabilities associated with the process making the access to see if a capability is present for the desired data object and access mode. This has been called a ticket-oriented approach.

3.3.2. Enforcement Subsystem

Enforcement subsystems commonly operate in one of two ways. The first, often called file mapping, employs a processor's memory-management hardware. The decision subsystem

[1] NT stands for New Technology. Well, this technology is not that new anymore, maybe that is why Microsoft decided to use Windows 2000 name instead of NT 5 for its latest release.

initializes this hardware upon the transfer of a file to active memory and no further software action occurs. The memory management hardware then enforces accesses. The second method distributes enforcement throughout the elements of the OS that are responsible for transferring data from passive storage to active memory and those that are responsible for performing other security sensitive operations. Many OSs use both kinds of enforcement subsystems.

3.4. Application Access Control

The issue of whether the operating system (OS) or the application should handle the access control and if they both share this responsibility, to what extent, is very controversial especially with Database Management Systems (DBMS). The DBMS performs a variety of data access such as edit, query and reporting operations and today, it manages most of the access control. There is some sharing of access control between OS and DBMS, but operating systems do not provide granularity protection needed by DBMS [SpLa89:11]. The access control mechanisms provided by OS are usually too coarse and concern only certain resources. Operating systems deal with object of a single level of granularity, while DBMS must be able to deal with objects of a variety of granularities, for example relation, field and tuple. The ability to support a fine granularity is therefore desirable. (See Chapter 10.1 for more information on fine-grained access control.)

There exist several architectures for secure DBMS (SDBMS). Graubart [SpLa89:167-190] identifies three most prominent SDBMS architectures: Hinke-Schaefer, Integrity-lock and Dual kernel approach. Then he compares them in terms of storage overhead, level of security, ease of retrofit and performance.
There is no clear winner in that comparison. Some of these architectures offer better security, some provide greater flexibility, while others provide better performance. Hinke-Schaefer architecture relies on the underlying OS to provide protection, while Integrity-lock and Dual kernel architecture require minimal OS support for security.

3.5. Alternatives

An information system is viewed as being secure if it can withstand an attack. The attacks exploit flaws in designed security mechanisms and more often interactions between different mechanisms. Testing can show some defects in system behavior, but it is impossible to uncover

all defects through testing. Testing is time consuming and expensive, and it cannot prove that a system is secure [Summ97:274].

An alternative to testing is to model the security of the information system. These models formalize the security policy supported by the system. Now we have to show that a particular model corresponds to a searched security aspect of the information system, next to prove that the model is secure. The problem is that formal models stipulating information system do not exist today and it would be difficult to build such a model. Furthermore, the methods for establishing correspondence between a system and a formal model were found to be impractical, even when such a system was built with such a correspondence in mind. Therefore, finding this correspondence for COTS products is not realistic, since COTS are not built with such verifications in mind and generally do not offer a required access to its architecture.

Experience shows that information systems can be secure up to a point. There will always be some degree of insecurity, some residual vulnerability. Instead of absolute security that preoccupied authors of formal security models, we should view system security with respect of the perceived threat. We agree that insecurity exists and it is impossible (at least in a foreseeable future) to destroy it. However, it is possible to shift insecurity from one place to another in such a way that a system is no longer vulnerable to a specific threat. As an example, cryptography used for secret message exchanges, shifts insecurity from protecting the message content to protecting cryptographic keys. In this manner, vulnerability of the system decreases with respect of the perceived threat, being in this case interception of messages. In a different context where it would be difficult to intercept messages but easy to access cryptographic keys, the perceived threat will be different and encryption would no longer work.

The assessment of system vulnerabilities let us identify system insecurities. An attack can be simulated by using a team of people with technical resources that are comparable with an actual threat. This team will search the system for vulnerabilities, examining the system in its usage context. The advantage of this approach is that a system is stressed in its working context. The disadvantages to this method include cost, because the assessment must be carried on a per system basis. Then, there is no method for analyzing the propagation of vulnerabilities and attacks through the system. If such a method existed, the designers would be able to move system insecurities away from the threats.

3.6. Remarks

The major challenges in access controls are extending the familiar centralized access controls to network access controls. Networks require new tools: techniques for dealing with scale, and techniques for secure communications and for managing trust. The latter techniques require strong cryptography. Scale is a problem because, as we move from network management to enterprise-level management to management of large-scale distributed systems, we progressively lose any notion of security state. There are various types of access control policies. An identity-based policy specifies access rights for individual users or group of users and specific target resources. A rule-based policy is expressed in terms of rules applying to all users and all target resources in a security domain. All controls require access control information to be generated, distributed and stored in a manner that assures its integrity. Provision needs to be made for revocation of access control information.

The two fundamental access-control mechanisms are access control lists and capabilities. While it is easy to revoke privilege with ACLs, it is probably prudent to build expiration dates into capabilities. Capability-based mechanisms have attracted rekindled interest with the advent of distributed systems. Capabilities are not a new concept, but up to now, they are not widely deployed. With these mechanisms, it is difficult to get an overview of subjects that have permission to access a given object. Also, it is difficult to revoke the capabilities.

Security models formalize security policies. Today, the formal policy models have limited usage in designing information systems. The dream of establishing conformance of a set of security mechanisms with a security model must await the development of newer more realistic and more comprehensive security models. The binary approach of software trustworthiness is not flexible enough for "real world" applications. Policies requiring a trusted computing base (TCB) to be based upon TCSEC might be acceptable for generalized software. But expanding TCB to include additional functionality requires reevaluation. Hence, further work is needed to remove limits of both system models and type of security policy that can be captured by the model. Other models have been proposed that target the involved application. These include Clark/Wilson, Chinese Wall and DoD Goal Security Architecture (DGSA). The DGSA is the most flexible; it is oriented toward a range of access controls and integrity policies in an object-oriented, distributed system environment. But, DGSA is more a list of goals than a security model.

The primary reason that formal models of the type of the Orange Book have failed is because their focus is on information flow and access control. Modern security policies have many components that cannot be properly modeled with these frameworks.

Establishing the correspondence between the information system and a formal model is not a practical approach to gain assurance that the system will be resistant to an attack. An alternative of shifting insecurity from one subsystem to another with respect to a threat has to be further studied in order to determine the feasibility of this approach. Also, we need a practical method to measure the security of COTS products. This evaluation of COTS products should give some incentives to its producers and customers. It should be financially viable and give an advantage over unevaluated equivalent products by providing an assurance of security.

Access controls are implemented at two levels: OS and application. The problem with OS is that the access control is too coarse. Moreover, modern applications tend to involve the security policies in terms of application-level abstractions rather than OS ones. Thus, it is obvious that enforcing security will be a responsibility shared between the OS and the applications. Research is needed to understand how the responsibilities might be best partitioned.
Proposed approaches and solutions in the database security arena in many cases have evolved from those used for operating systems. Database security, however, poses different challenges based upon the finer granularity at which multilevel security must be invoked and the different level of abstraction. Additional research is needed to better understand access control management, especially for updates in database (application) environment.

You can verify the system security through: testing, conformance to security policy and simulated attacks. Testing can show some defects in designed security mechanisms, but it is impossible to uncover all defects through testing. Testing is time consuming and expensive, and it cannot prove that a system is secure. Formal models stipulating information system do not exist today and it would be difficult to build such a model. An attack can be simulated by using a team of people with technical resources that are comparable with actual threat. The advantage is that the system is tested in working context. The disadvantage is that it is expensive, since it has to be done on a per system basis.

Chapter 4

Identification and Authentication

"Encryption is the principal technology that can be used to authenticate a person or a document authored by a person across a long-distance Internet link." [DdDp98:386]

Identification is a declaration of an entity's identity. In the simplest form, this declaration could be a claim that the entity makes. Authentication refers to the verification process by which a system establishes that an identification claim is valid. It is a process of proving claimed identity [ChBe94:119]. A number of authentication mechanisms are commonly used in practice. Historically, the mechanisms have been characterized as something you know, something you have or something you are. The latter refers to innate biological properties of the user and therefore is not applicable for computer-to-computer authentication.
Login is an example of user identification and authentication procedures. You announce who you are by typing your user name (ID) and prove your identity by providing the password.

4.1. User Identification

The first essential in making the information system secure is to identify the user. There are two reasons to authenticate the user: for access control and for records in the audit trail [Goll99:19]. The system can identify you using one or a combination of the following methods:

1. By something you know. You could memorize a password or personal identification number (PIN). Techniques of this type are the least expensive and can be made reasonably secure. The security of this scheme depends upon careful password selection. Unfortunately, many users choose common words for passwords [Cobb90:269-273].
2. By something you possess. You may have a key, a badge, an Automatic Teller Machine (ATM) card or a smart card. Like before, anyone who is in possession of the token can impersonate the legitimate user [Goll99:27].
3. By something you are. These methods use biometrics to authenticate a person such as palm prints, fingerprints, iris patterns, retina patterns or voiceprints. Unlike the previous forms of authentication, these cannot be stolen, transferred or altered. Biometrics are recorded and then compared with the actual measurements. Because these patterns will rarely match precisely,

we face a new problem, false rejection and false acceptance. These are the least used and usually the most expensive ways of recognizing a person [Cobb90:193-201].

4. By your location. When you login, the system may take into account your location. These kinds of decisions will become important in mobile and distributed computing.

Even though cryptographic protocols were invented to supplement passwords for authentication, passwords are still the most common method of authentication [Ford94:110-117]. To reduce the odds of successful password-guessing attack, many system administrators set policies managing passwords. The entire authentication function can be moved to a separate server on the network, where the passwords and keys can be further protected. Examples of such system are Kerberos [NeTs94] (see Appendix D) and Secure Remote Procedure Call available with operating systems from Sun Microsystems Inc. Spafford [Spaf92] proposed a scheme to eliminate weak passwords. The fundamental problem underlying password schemes is the reusability of passwords. A technology that allows one-time passwords would not be subject to attack of password guessing or sniffing. Leslie Lamport of Digital Equipment Corporation proposed a mechanism for one-time passwords that could be implemented in software. This mechanism was later released by Bellcore and is known under the trade name S/Key [Hall94].

The technology of smart cards is now mature and cheap enough to provide a practical mechanism for using one-time passwords. People are used to carry keys and cards, so addition of a smart card to access their computer resources would greatly improve the protection of their information. The future computing devices may be equipped with smart cards readers and smart cards will be activated by personal features of the holder as well as by the personal identification number (PIN). Companies eager to promote e-commerce such as banks, the Web bookstores and auctioneers are likely to push for adoption of such authentication technology.

The attraction of using a voiceprint to identify the user exists because only a telephone is needed. Since the user's voice differs from time to time, we must extract certain sound characteristics and compare these, using permissible limits of variation. For practical speaker verifications, the comparisons of the pitch, intensity and time warp[1] is used. The other physical characteristic that has been used to identify a person is the shape of the hand. The device measures the lengths of the fingers. Apparently, this method provides a better means of recognizing individuals than voiceprints.

[1] Time warp deals with speed variations of different parts of speech.

4.2. Signature Schemes (Digital Signatures)

Much like paper transactions, checks and purchase contracts that are authenticated by handwritten signatures, digital signatures play a similar role in electronic messages and transactions. Both kind of signatures identify the signer and prevent repudiation, but unlike handwritten signature, digital signature asserts the integrity of the message after it has been received [MOVa96:22]. There are other fundamental differences between digital and conventional signatures. The respective verification procedures are different. The verification of conventional signatures involves comparing the signature with stored template and this, of course, is not a very secure method. For digital signatures there exist publicly known verification algorithms, hence, anyone can verify the signature, but secure digital signature schemes will protect against forgery. Another difference is that "copy" of a digitally signed document is identical to the original, while signed paper document can usually be distinguished from the original.

Digital signatures are one of the services provided by cryptosystems. Most of the schemes use public key cryptography instead of conventional cryptography.
Some of the schemes are enumerated below:

- RSA.
- ElGamal.
- Modified Diffie-Hellman algorithm for key sharing.
- Lamport-Diffie digital signature based on DES.

Merkle extended the last scheme to allow any number of signatures and Michael Rabin developed a scheme having a security of underlying block cipher such as DES. All of these schemes are based on conventional cryptography and so far have not found any practical use.

Digital signature consists of two algorithms: signing algorithm and verification algorithm. The user will sign the message and the signature will contain some secret information. The receiver of the message will verify authenticity of the message by using publicly known algorithm and sender's public information.

Digital signatures using public key cryptosystems must be a function of an entire message. Since messages can be long and public key cryptosystems are relatively slow, one-way hash functions were introduced to create message digest, which is a unique and compact representation of a

message. The message digest is used to detect modifications in the message. However, the message digest alone is not enough, because the hash algorithms used to compute it are publicly known. To authenticate the message, the message digest has to be encrypted with a secret key, thus forming a digital signature.

Besides being secure, digital signatures have to meet certain performance criteria for signing and verifying. The availability of portable computing devices such as Hand-held Personal Computer (HPC) and smart cards, with reduced computing power, requires us to take into account the frequency of signing versus verifying done on these devices. Since we do not know whether HPCs are more often used for signing or verifying, both algorithms should be fast to be useful.

4.2.1. RSA

Rivest, Shamir and Adleman are authors of the best-known and most versatile public-key cryptosystem – RSA, invented in 1977. It supports both secrecy and authentication, and hence can provide complete and self-contained support for public key distribution and signatures. Because of versatility and ubiquity of this algorithm, we include its complete description.

The transformations for encryption E and decryption D in RSA algorithm are defined as follows:
$E(M) = M^e \bmod n = C$,
$D(C) = C^d \bmod n = M$,

where M is a message to be encrypted and C is the encrypted message,
(e, n) is a public key,
d is a secret key,
n is a large integer such that $n = pq$, where p, q are prime numbers, $ed \equiv 1 \bmod \phi(n)$ and
$\phi(n) = (p-1) * (q - 1)$
p, q and d are kept secret, while e and n are published.

The generation of a pair of secret and public keys used by RSA, is done in the following steps:

1. Generate two large primes, p and q. Finding prime numbers in practice, involves generation of large random numbers which are then tested for primality using a probabilistic Monte Carlo algorithm such as Solovay-Strassen or Miller-Rabin.
2. Compute $n = p*q$ and $\phi(n) = (p-1) * (q - 1)$.

3. Select a random e such that $0 < e < \phi(n)$ and $\gcd(e,\phi(n)) = 1$. The requirement for gcd is verified by using Euclidean algorithm.
4. Compute $d = e^{-1} \bmod \phi(n)$ using extended Euclidean algorithm.
5. Publish n and e in public directory and keep d as a secret key.

The security of RSA is based on the belief that $E(M) = M^e \bmod n$ is a one-way encryption function with the trapdoor that is a knowledge of the factorization $n = pq$, which is used to compute decryption exponent d. It is conjectured but not proved that breaking RSA is polynomially equivalent to factoring n. Hence, one obvious attack on RSA is an attempt to factor n. Therefore, it is necessary to find n that is large enough that factoring it will be computationally infeasible. Since current factoring algorithms and computational power of today's processors allow factoring of numbers up to 155 decimal digits, it is necessary to chose modulus n of at least 220 decimal digits.

As previously mentioned, RSA can be used for both secrecy and authenticity. After signing a message with the secret key d, the sender encrypts it using receiver's public key. Receiver decrypts the message with her secret key, then encrypts it with sender's public key to obtain plain-text message. In practical solutions, only the message digest is signed. The signature altogether with the original message is encrypted with receiver's public key. Receiver will decrypt the message, generate its digest and compare it against the received one to authenticate the sender and test for the message integrity. Of course, the received message digest has to be encrypted with the sender's public key before comparison. Hence, RSA protocol for digital signatures works as follows:

1. Generate message digest by using SHA or other publicly known hash algorithm.
2. Sign the digest with the secret key d, $S = H(M)^d \bmod n$ and send it with the original message, after optional encryption step. H indicates hash function.
3. Receiver obtains the message and its signature, generates its digest and compares it to the received one. Since $S = H(M)^d \bmod n$, by raising both sides to the power e, which is sender's public key, we obtain $S^e = H(M)^{de} \bmod n$. For authentic sender d is an inverse of e and hence, this equation has to hold $S^e = H(M) \bmod n$.

An encryption and decryption transformations involve exponentiation modulo n, which for large n is done most efficiently by using square-and-multiply algorithm. These functions can be

performed in polynomial time as showed in Stinson [Stin95:127]. Note that setting up RSA requires generation of n, e and d, which time complexity is $O((\log n)^3)$.

It is a known fact that current hardware implementation of RSA cryptosystem using 512 bit modulus n is about approximately 1500 slower than DES. For this reason, RSA is used mainly for secret key distribution and digital signatures.

The question about RSA security that still remains unanswered is whether it is possible to break the cryptosystem without factoring n. The obvious treats to RSA today are developments of new factoring algorithms and increasing processors computing power.

4.2.2. ElGamal

ElGamal public key cryptosystem was proposed for digital signatures and for confidentiality. It is based on discrete logarithm problem, for which there is no known polynomial-time logarithm and is regarded as a difficult problem in finite field GF(p), where p is a large prime number. To thwart the known attacks, p should have at least 150 digits and $p-1$ should have at least one large prime factor. An exponentiation modulo p for suitable prime p is the one-way function in ElGamal scheme, while a secret key x is the trapdoor. In a finite field, where p is a prime number and α is a primitive element, discrete logarithm problem can be stated as finding unique exponent x such that $0 \leq x \leq p-2$ and $\alpha^x \equiv \beta \bmod p$.
Suppose Alice wants to use ElGamal scheme. She needs to choose a secret key x, compute her public key y, $y = \alpha^x \bmod p$ and publish (α, p, y). Now if Bob wants to send Alice a message, he needs to choose a random number k such that $0 \leq k \leq p-1$, compute
$c_1 = \alpha^k \bmod p$, $c_2 = My^k \bmod p$ and send to Alice a pair of integers (c_1, c_2). Since Alice knows x, she can compute y^k by using α^{kx}, then obtain M by dividing the second element of the received c_2 by y^k, as follows $M = c_2 * (c_1^x)^{-1} \bmod p$.

ElGamal Signature scheme was described in 1985 and National Institute of Standards and Technology (NIST) adopted its modification as a Digital Signature Standard (DSS).

The signature function sig(M, k) = (r, s) takes two parameters: message M and random number k, such that $0 \leq x \leq p-1$, and k cannot be used for signatures of any other previous messages.

Stinson [Stin95:209] shows how the system can be broken if the same k was used for two different messages.

The signature consists of two integers r and s computed as follows:

$r = \alpha^k \bmod p$

$s = (M - xr)k^{-1} \bmod (p - 1)$

Note that triple (α, p, y) is public and x is secret as it was in ElGamal cryptosystem.
If the signature was constructed correctly, then verification will succeed, since:

$y^r r^s \equiv \alpha^{xr} \alpha^{ks} \bmod p$

$\equiv \alpha^M \bmod p$

The signature is computed by using both, secret value x and random value k (used to sign one message M), and verification is accomplished by using only public information (α, p, y, r, s).

To forge the signature for message M, the opponent has to find a tuple (r, s). If she chooses r and than tries to find s, she has to compute discrete logarithm $\log_r \alpha^M y^{-r}$. If she chooses s and tries to find r, then she needs to solve this equation $y^r r^s \equiv \alpha^M \bmod p$ for unknown r. The solution to this problem is unknown but it does not seem to be related to Discrete Logarithm problem or any other known problem.

There are methods for producing valid forged signatures for "random" messages produced by ElGamal scheme. One of them involves simultaneous calculation of (M, r, s) and the other uses previously signed message. Neither of the methods allows opponent to forge a signature for the message of her own choosing, hence, they do not seem to represent a threat to the security of ElGamal scheme.

4.2.3. Diffie-Hellman

The approaches of establishing secret keys fall into two main categories: key distribution and key agreement. In key distribution, one of the parties generates the key and transmits it to other party or parties. Key agreement is a protocol under which interested parties will establish a secret key simultaneously by communicating over public channel. Diffie-Hellman is the first known key

agreement protocol. The strength of this algorithm is based on a difficulty of finding the discrete logarithm in finite fields or Galois fields. Stinson [Stin95:267-268] proves that breaking the ElGamal cryptosystem is equivalent to solving Diffie-Hellman problem. We mention this algorithm not because it is useful for authentication purposes, but to dispel confusion about it.

Summers [Summ97:198] suggests that Diffie and Hellman invented public key cryptosystem that could be used for authentication by using their algorithm for public key system, but in reverse order. The problem is that Diffie-Hellman is not a public key cryptosystem, nevertheless, it is possible to Diffie-Hellman key-sharing algorithm for authentication purposes.

Diffie-Hellman algorithm cannot be used directly for encryption or signing. We propose modifications to original Diffie-Hellman for authentication purposes that use the possession of a secret shared key as a proof of identity. Say, Alice wants to prove her identity to Bob and they both established secret key through Diffie-Hellman. To do that, she might generate message digest by using Secure Hash Algorithm (SHA) and then "sign" it by XOR-ing the key with the digest. There are algorithms that can be used to convert the shared secret key to cryptographic keying material. The length of such key needs to be adjusted, so it matches the length of the message digest. Bob, after receiving the message from Alice, will generate its digest using the same hashing algorithm and apply to it key material based on their shared secret key. If the result matches Alice's signature then Bob can be certain that nobody but Alice sent the message. However, this is not a digital signature in a general sense because, it cannot be verified by a third party and there is no way to verify whether the author is Alice or Bob.

4.2.4. Binding Signatures to Messages

The sender sends the digital signature along with the message. First, a message digest is created, by applying cryptographically strong one-way hash function to the message of an arbitrary length. It is used to detect changes in the message, but unlike other methods used for this purpose (like CRC), it is computationally infeasible for an attacker to devise a substitute message that would produce an identical message digest. Since a message digest alone is not enough to authenticate the sender has to encrypt (sign) the message digest with the secret key. The message digest encrypted by the secret key forms a digital signature.

Signature should be bound to the plaintext message by encryption. Otherwise the opponent can intercept a plaintext with digital signature and substitute the signature with her own, potentially without modifying the original message. The receiver would infer that the opponent is the author of the message. Similar scenario arises when the digital signature is generated after encrypting the message. The opponent obtains ciphertext and its signature, replaces the signature with her own, and forwards it to the receiver. Note that opponent can sign the ciphertext without knowing the plaintext. Because of this difficulty, it is recommended to sign a message before encryption.

4.2.5. Verifying Signatures

Verification algorithm uses publicly known information, so anyone can verify the authenticity of the message, not only receiver. Since, the signed message is usually encrypted, the legitimate receiver first has to decrypt the message then detach the signature to verify it and possibly store it as a proof for the third party, for example, arbitrator. After decrypting the message the digital signature is verified. Because only digest of a message is signed, the receiver computes a new message digest from the plaintext and checks the one recovered from the digital signature. The match proves the message was not altered and that it came from the sender who owns the public key used to check the signature. A potential forger would have to either write an altered message that produces an identical message digest, which is infeasible, or to create a new digital signature from a different message digest, also infeasible, without knowing the true sender's secret key.

4.2.6. Idempotence and Non-repudiation

Because the copy of a digitally signed message is identical to the original, some precaution has to be taken to prevent reuse of the signed message. If the user authorizes a withdraw from her account of a certain amount, she wants to be sure that the amount is withdrawn once, that is pay $1M != pay $10M. Hence, the signature should contain a time stamp that prevents the message from being reused.

Non-repudiation services provide unforgeable evidence that a specific action occurred. It can occur on three levels: non-repudiation of origin, non-repudiation of submission and non-repudiation of delivery. Non-repudiation of origin protects against any attempt by a message originator to deny sending a message. Non-repudiation of submission protects against any attempt

by a Message Transfer Agent (MTA) to deny that a message was submitted for delivery. Non-repudiation of delivery protects against any attempt by a message recipient to deny receiving a message. Digital signatures prove who sent the message (non-repudiation) and that the message was not altered either by error or design.

The non-repudiation services are similar to their weaker proof counterparts (i.e., proof of submission, proof of delivery, and message origin authentication). However, non-repudiation provides stronger protection, because the proof can be demonstrated to a third party. Symmetric encryption cannot guarantee non-repudiation. Since both the originator and recipient share the symmetric encryption key, either party can generate the proof.

4.3. Identification Schemes

Authentication of entities such as users, processes and machines is an essential mechanism in information system security. A user is given an identifier, which while unique, is not secret. To login to the system the user have to present this identifier and prove the claim to it by providing evidence that no one else can provide. The system verifies this evidence against stored information and either allows or refuses the access. The parties in the authentication process include the user, some login component and authentication server (AS).
Here are our definitions for identification and authentication for people.

Identification is the process by which a user provides a claimed identity to the system.
The most common form of identification is the user ID.

Authentication is the process of establishing the validity of a user's claimed identity to the system.

The methods described in 4.1. to authenticate a person to the machine are adequate for isolated machines. With client-server computing, the user's programs require services from servers. The client must be assured that the right server is handling its requests and the server has to be assured that client properly authenticated the user. In the computer networks, the machines must mutually authenticate themselves. As networking technology evolves and more computing is done on the network, network security struggles to catch up. At present, there is no widely accepted architecture or model for network security, while the networks spanning all continents are governed by different laws and standards.

Network security services cope with harder problems than the ones in the isolated systems. Passwords transmitted in clear text are vulnerable to eavesdropping. Mutual authentication is needed since the users cannot know what server(s) they are communicating with. Besides authenticating the users, hosts have to authenticate themselves to other hosts and processes to processes. This type of authentication is known in OSI terminology as peer entity authentication or identity authentication. Entities participating in authentication are called principles. In distributed processing, the principles are authenticated by the authentication servers employing different protocols and policies. This makes the trust relationship between authentication servers very complex. The authentication done at the beginning of the session might become invalid - it can be hijacked. Attacks can modify the messages, including the ones used for authentication. Networks might include nodes and links that are not trusted by source – destination hosts. The intruders pose bigger threat to networks since they are more remote and harder to trace. Cryptographic technologies become more indispensable in the networks since nodes send information through territories they do not control. The question is not whether to use or not cryptography to protect information systems, but at what layer [Goll99:225].

There are several identity authentication schemes and in the simplest form, a claimant authenticates itself to a verifier by providing some proof of identity such as a password. This scheme is disclosing since the proof of identity is revealed to eavesdroppers. Figure 4.1 shows the steps for simple authentication protocol.

Claimant ——— User ID ——▶ Verifier
Claimant ◀— Enter password — Verifier
Claimant ——— Password ——▶ Verifier
Claimant ◀— Accept or reject — Verifier

Figure 4.1 Simple authentication protocol

This scheme is very weak since the password is sent in cleartext and eavesdropper can intercept it and use it later to impersonate the claimant. Besides, the passwords tend to be weak and the authentication is one-way. The client has no way of verifying the identity of the host. Modification of this method is to apply one-way transformation function to the password. This way the attacker cannot learn the password but still can intercept it and replay.

In another scheme, the client sends both clear and encrypted message. Both claimant and verifier share a secret key. Again, the entire message can be intercepted and replayed. To protect the authentication protocol from replay attack, the verifier has to be sure that a message is fresh. It can be achieved either by including a time stamp or a nonce[1]. Figure 4.2 shows an example of protocol employing nonce (n).

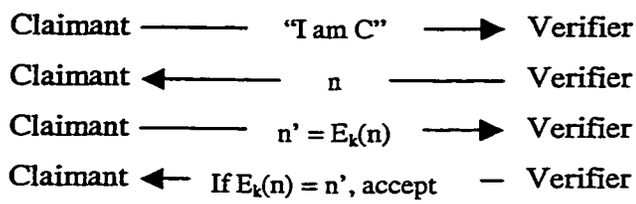

Figure 4.2 Protocol with nonce

The verifier sends a nonce (challenge) to the client. Because the attacker would never eavesdrop on the nonce before, it cannot replay the message. This protocol is an example of strong authentication in which claimant does not disclose any secret. However, one-way strong authentication is not good enough in distributed environment, which requires strong mutual authentication of the principles. But designing of mutual authentication protocols is very difficult, because they should be secure and efficient. Figure 4.3. shows and an example of a seemingly good yet flawed protocol.

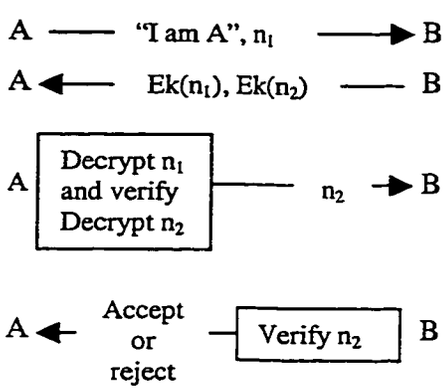

Figure 4.3 Flawed mutual authentication protocol

A proves to B its identity by showing that it can decrypt B's challenge – $E_k(n_2)$. B proves its identity to A by showing that it can encrypt A's challenge – n_1. The problem is that the

[1] Nonce is randomly generated identifier, used only once.

eavesdropper X can intercept first message from A to B and send the same challenge n_1 to A. Now, X can masquerade B once it receives response from A. Our solution to this is to reject by the principal all challenges that match exactly its own previous challenges. To achieve this, the principal would have to store its challenges. However, this method would quickly become impractical with the increase of the number of challenges. The length of the queue can be kept manageable by requiring challenge to have a certain life span. This would imply keeping time at nodes quite synchronized that is "exact" within certain time window.

The strong authentication protocols use either private or public key cryptography. For the key management and to avoid exposures to other security threats, third party, called authentication server (AS) has been used. AS is usually implemented on a separate host dedicated to authentication, administration of user IDs, resource names, passwords and management of cryptographic keys. This server would be either on-line or off-line. The decision would depend on the longevity of the credentials. For short-lived credentials, the server has to be on-line and this could create a bottleneck when multiple principles are involved. For long-lived credentials, the servers could be off-line. However, it is difficult or impossible to revoke credentials, which is a major disadvantage of off-line servers. Another principle in decision making on on-line versus off-line servers, is the type of cryptosystem used. For secret key cryptography, the on-line system must be used and for public key cryptography, either type of server configuration will do.

The authentication service design should consider which protocols and cryptosystems are used, how encryption keys are managed, how much trust must be placed in AS and how authentication works across multiple domains of authority. Practical implementations of authentication protocols use challenge-response, with challenge changing for every use to prevent replay attacks. The challenge itself is either a time stamp or a nonce. Since it takes some time for exchanging messages, the time stamp is deemed valid within a certain time frame. To assure integrity of current time, a secure time service might be used. In addition to cryptosystem attacks, the authentication service is vulnerable to authentication protocols attacks. The attack where an intruder uses a second execution of a protocol to trick the principles to reveal critical information is called an oracle session attack. The attacks using one message of the protocol to be used later in the protocol are known as interleaving attacks.

4.3.1. Secure Remote Login

Secure Shell (SSH) was developed in 1995 in Finland by Tato Ylonen, a computer science professor with a goal of secure remote access. Since 1998, there are commercially available SSH client programs, intended to replace the telnet, rlogin and rsh utilities, providing secure communications over an insecure network. These applications use strong cryptography for protecting all transmitted confidential data, including passwords, binary files and administrative commands. SSH protocol is the de-facto standard for encrypted terminal connections on the Internet. The benefits of SSH include:

1. No passwords are sent in cleartext to protect against password theft.
2. Multiple strong authentication methods to shield from spoofing, trojan horses, etc.
3. Strong authentication using agents to enable a single login to multiple systems.
4. Encryption and compression of data for security and speed.
5. Secure file transfer.
6. Tunneling and encryption of arbitrary connections.

SSH allows secure remote sessions that provide strong authentication with effective privacy without the need of one-time password systems. Even though one-time passwords can be quite effective in preventing reusability problems related to ordinary password schemes, they have a number of drawbacks. The administrative load of a one-time password scheme is higher than with ordinary passwords and you need to provide a secure storage for one-time passwords.

The disadvantages in SSH include an inherent risk in the fact that private RSA keys are sometimes stored in a shared user environment like UNIX, which makes them vulnerable to hacker attacks. SSH provides some protection against this threat by refusing to use RSA authentication if local files are not appropriately protected, and by the need of a user pass-phrase[1] to unlock the private key. The major disadvantage in SSH, however, is that it does not support X.509 public-key certificates to distribute and manage RSA keys. This hinders SSH from several management applications and integration to other Intranet applications, such as Light Directory Access Protocol (LDAP) directories and certificate servers. Despite these facts, SSH is an effective, simple, and easily deployed tool that can improve the security of a corporate architecture while providing the flexibility of remote access to the Intranet.

[1] Pass-phrases are usually longer than passwords and are easier to memorize.

4.3.2. Address-based Authentication

The dominant form of host authentication in today's Internet is address-based. That is, hosts often decide to trust other hosts based on their IP addresses[1]. Another form of host-to-host authentication on the Internet is name-based. It is even worse than IP-based, because the intruder can attack the mechanism that is used to map IP addresses to host names. This authentication scheme can be circumvented by misrepresenting (spoofing) IP address and number of attacks can be accomplished by sequence number guessing [Morr85], or route corruption, or address to name translation mechanism.

An address spoofing attack exploits several weaknesses:

- The trust relationship between two computers.
- The predictability of TCP's Initial Sequence Number (ISN), which plays an important part in the ordering of all subsequent exchanges in the conversation.
- The weak (IP address based) authentication used by some commands.

The possibility of an address spoofing attack was first suggested by Robert Tappan Morris in 1985 [Morr85] and expanded by Stephen Bellovin [Bell89:32-48] who noted that:
"Some of these flaws (sequence number spoofing, routing attacks, source address spoofing, and authentication attacks) occur because hosts rely upon IP source address for authentication... Others exist because network control mechanisms and in particular routing protocols, have minimal or non-existent authentication".

IP can be encapsulated within TCP, a technique known as tunneling. Such tunnels can then be used to bypass the firewall. The extent of damage done depends upon how routing information is propagated [ChBe94]. In conclusion, the address-based authentication is weak because network address can be impersonated quite easily. There are ways to prevent forgery of IP addresses to some degree but since they all depend on the network service provider, they violate the least privilege principal and hence are questionable solutions. The real solution is to employ cryptographic protocols for network authentication and security, and these protocols are now growing in prominence on the Internet.

[1] Actually, it is worse than that because much authentication is name-based, which opens new avenues of attack. But if an attacker can spoof an IP address, there is no need to attack the name service.

4.3.3. Cryptography-based Authentication

The emerging on-line business is vulnerable to competitors and hackers who might obtain, divert or alter e-business information. A robust network security system must provide confidentiality and authentication to prevent these attacks through cryptography. For usability and scalability reasons, cryptographic authentication schemes evolved into sophisticated protocols to exchange information between two or more parties. These protocols are used to identify the users and to distribute session keys [Schb96:56-65]. Cryptographic authentication can employ either symmetric or asymmetric cryptosystems. The large deployments of cryptographic authentication systems rely on a third party. For conventional cryptography based systems, they are called Key Distribution Centers (KDC) and public key cryptography uses Certification Authorities (CA).

Cryptographic authentication requires possession and hence the storage of a secret. If no auxiliary storage, such as smart card, is available then the secret or private key is derived from a password or pass-phrase. In case when keys are derived from password, Gong et al. [GLNS93] demonstrated how cryptographic communications protected by such keys can be attacked using password guessing. Such attacks have been reported against S/Key and Kerberos [NeTs94]. Although techniques to guard against the attacks are known [BeMe92] [GLNS93], they are rarely employed.

A major advantage of cryptographic approach is that it can provide continuous authentication, in which each packet is authenticated. The alternative to continuous authentication is to authenticate the entity at the beginning of the session (or whenever the authentication process is invoked). However, this method is vulnerable to session hijacking, where an attacker impersonates the authenticated entity [Jonc95]. The continuous authentication becomes critical as expertise of the imposters grows.

4.3.4. New User Services

The continuously increasing number of information system services in financial, educational, transport and postal industries require user authentication to ensure privacy and integrity of information. Here are some of the issues related to new services.

What are the security implications of adding new, more and more sophisticated user services? Will the existing security protections be spread thinner and thus provide less protection for today's services?
Is the Internet ready for business?

The users will adopt the new electronic services provided these services are secure. The idea of doing business electronically is not new. The products can be ordered through phone and fax, and payments can be done through Automatic Teller Machines (ATM). The challenge is to provide all these new services over unreliable and insecure medium like the Internet [Losh95:2].

The new services are secured by ad-hoc and proprietary solutions. For example, there are multiple consortiums of banks and financial institutions that push for e-business without waiting for some standardized framework to take place. There is a great deal of work to standardize secure communication protocols, to secure world-wide payments over the Internet, to establish Public Key Infrastructure (PKI), and to link territorial and national PKI.

Whether the Internet is ready for business depends on the requirements of the business. There are already numerous examples of businesses using the Internet for advertising, marketing, sales of products and services and coordination with business partners. On the other hand, the Internet is susceptible to eavesdropping and denial of service attacks. Consequently, most e-businesses users restrict what they entrust to the Internet.

4.4. Authentication Codes (MACs)

The Message Authentication Code (MAC) is a cryptographic checksum appended to a message to ensure message integrity. Unlike digital signatures, MACs are computed and verified with the same key. There are four types of MACs: unconditionally secure, hash function-based, stream cipher-based and block cipher-based.

Stinson [Stin95] proposed an unconditionally secure MAC based on encryption with a one-time pad. The ciphertext of the message authenticates itself, as nobody else has access to the one-time pad. However, there has to be some redundancy in the message. An unconditionally secure MAC can also be obtained by use of a one-time secret key.

Hash function-based MACs (often called HMACs) use a key or keys in conjunction with a hash function to produce a checksum that is appended to the message. An example is the keyed-MD5 method of message authentication.

Lai, Rueppel and Woolven [LRWo92] proposed a MAC based on stream ciphers. In their algorithm, a provably secure stream cipher is used to split a message into two substreams and each substream is fed into a Linear Feedback Shift Register (LFSR); the checksum is the final state of the two LFSRs.

MACs can also be derived from block ciphers. The DES in cipher block chaining mode (CBC) MAC is a widely used U.S. and international standard [NIST85]. The basic idea is to encrypt the message blocks using DES in CBC mode and output the final block in the ciphertext as the checksum. Bellare et al. give an analysis of the security of this MAC [BKRo94].

Historically, MACs have been used to provide authentication in financial systems. One of the difficulties of using MACs has been the complexity of conventional key management. However, a standard has evolved to assist in key management for well-defined communities, such as ANSI Standard X9.17.

4.4.1. Message Integrity

Message integrity refers to unauthorized modifications or destruction of a message. The Message Integrity Code (MIC) is a synonym for message digest that is used for digital signatures. A cryptographically strong hash algorithm such as MD5 generates the MIC.

4.4.2. Message Author Authentication

Identification and authentication of message senders or network users plays a significant role in security protocols devised for message exchange or changing parameters for programmable network components. Message author authentication, also known as message origin authentication, allows the verification of a message originator identity. Since origin authentication has limited utility without content integrity, the message origin authentication service also provides assurance that the message content has not been modified.

Message origin authentication can be provided by one of two methods: a message origin authentication check or a content integrity check.

The message origin authentication check allows the identity of a message originator to be verified by the message recipient(s) and by any Message Transfer Agent (MTA) transferring the message. It is provided on a per-message basis, using asymmetric encryption techniques. The message origin authentication check is a digital signature attached to the message. If the signature is computed using the plaintext content, the message origin authentication check also provides non-repudiation of origin. This provision is not maintained if the signature is computed using the encrypted message content [Stin95:205]. The message originator, although unable to deny sending the encrypted content, can deny that the content is the same as the original plaintext.

The second method to provide message origin authentication is the content integrity check. It is provided on a per-recipient basis, using either symmetric or asymmetric (i.e., a digital signature) encryption techniques. The content integrity check is a cryptographic checksum included as a per-recipient field in the message envelope, or in the message token. Unlike the message origin authentication check, the content integrity check must be computed as a function of the plaintext message content.

4.5. Remarks

Authorization is an increasingly vital link in security protocols. Traditional authentication mechanisms, especially over the Net, are increasingly obsolete. We need to find stronger, more robust authentication for both human and computer communications over the Internet.

Biometric identification technologies have limitations when employed in network contexts. While biometric features are unique and cannot be stolen, the authentication needs encryption to avoid replay and man-in-the-middle attacks. Strong, mutual authentication cannot be achieved with biometrics alone. Besides, most of the authentication interactions are between network nodes and processes. Still, for use in a closed environment, biometric techniques that employ microphones or low-cost cameras in personal computers are worth exploring.

The fundamental problem with passwords is their reusability. This allows capture by an adversary. Leslie Lamport proposed a mechanism for one-time passwords that could be implemented in software and used in smart cards. One-time passwords would not be subject to attack of password guessing or sniffing. The security of smart cards lies with strength of the algorithm that generates one-time passwords.

By combining what do I know (passwords), what do I have (smart card) and what I am (fingerprints), it is possible to reliably authenticate humans to computers or physical gateways. More challenging is mutual peer-to-peer authentication of computers over the Net. MACs are not really strong. It would be nice if we could move to cryptographic challenge/response protocols based on cryptography. Challenge/response and timestamp-based are two methods to provide replay protection. The disadvantage of challenge/response is the need for extra messages. While one-way authentication could be done with a single message and mutual authentication with one message in each direction, the challenge/response scheme always requires at least three messages. The disadvantage of the timestamp-based scheme is that it requires secure synchronized time. The factors limiting usage of cryptography in these protocols are firewalls, because the firewalls cannot deal with encrypted packets.

New user services constantly arise on the Internet. Hence, vulnerabilities of the service become vulnerabilities of the Internet. For example, the Voice over IP (VoIP) is likely to be more vulnerable than traditional Public Telephony Networks (PTN), most notably because Internet-based networks use in-band channels for routing and have end points that are prone to failure. Attention to these issues is needed.

Today, many organizations are interested in replacing paper-based systems with automated electronic systems. One of the inhibitors to the increasing use of electronic commercial transactions has been the concern for the risks of forgery over unsecured networks. This focus has brought the need for a reliable, cost-effective way to replace a handwritten signature with a digital signature. Even though, there are a number of digital/electronic signature software vendors in the marketplace today, this technology is still considered to be evolving and immature.

Digital signatures are one of the services provided by cryptosystems. Most of the schemes use public key cryptography instead of conventional cryptography. Note that vulnerabilities of underlying cryptosystem become vulnerabilities of digital signatures.

Because the copy of a digitally signed message is identical to the original, some precaution has to be taken to prevent reuse of the signed message. Digital signatures prove who sent the message and that the message was not altered during transmission. It also provided non-repudiation, which means that the sender cannot easily disavow the signature on the message.

We need to establish a focus group to identify business processes that would be enhanced or require the use of digital signature technology. The requirements from these business processes will stimulate this group to establish digital signatures as a part of the worldwide standard for securing e-commerce.

Secure Shell (SSH) should be used instead of traditional telnet, rlogin and rsh and other utilities for secure communications over insecure networks.

The dominant form of host authentication in today's Internet is network-based. That is hosts often decide to trust other hosts based on their IP addresses. The address-based authentication is weak because network address can be impersonated quite easily. Network-based (IP-based) authentication technology is inappropriate to high-assurance implementations. A much better option is to use cryptographic authentication in authentication protocols. But cryptographic protocols are difficult to get right and the technology for verifying these protocols is far from mature. Further research on crypto-based authentication techniques and supporting tools is strongly encouraged.

Chapter 5

Cryptography and PKI

"The basic purpose of encryption is to protect sensitive data from unauthorized disclosure." [HBHo95:16.2]

Encryption is a fundamental tool[1] for the protection of sensitive information, if it is correctly applied as part of an overall information security. It can be used to protect sensitive data in storage (files, tapes, hard disks) as well as to detect active attacks, such as message or file modification. Cryptography transforms the representation (appearance) of information without changing its content, to ensure its secrecy or authenticity or both. Plaintext (cleartext) is one representation of the information expressed in natural language, intelligible to all. Ciphertext is a different representation, designed to conceal the information from unauthorized persons. Encryption is the transformation from plaintext to ciphertext. Decryption is the reverse transformation[2]. The reason for encrypting is for transmissions over insecure channels. In the most basic form, cryptography provides secrecy by rendering information useless to anyone without the decryption key. Figure 5.1 demonstrates encryption and decryption mechanisms.

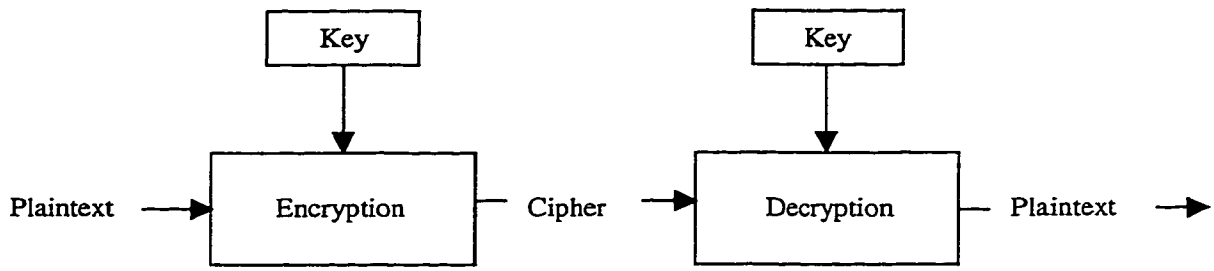

Figure 5.1 Encryption and decryption

[1] Encryption is a tool because it is a means for achieving an end, not and end in itself.
[2] Normally, transformations from plaintext into ciphertext and vice-versa involve one or more key.

Cryptography theory was first developed by Shannon[1] and was based on information theory [Shan48] & [Shan49]. Later, cryptographic algorithms depended on number theory and computational complexity. Even though many cryptography algorithms are built upon mathematical problems, it turns out to be very difficult to mathematically prove their security. Hence, cryptographers gain confidence in their cryptosystems as they survive the attacks by cryptoanalysts.

Keeping cryptographic algorithm secret does not make it more secure [Losh95:46], especially if it is employed in a commercially available application. Security through obscurity never works and soon enough, someone will reverse engineer the software to find out how the algorithm works. It takes a group of experts on cryptography to evaluate strength of a cryptographic algorithm, so keeping such algorithms secret is of questionable value. The end result may be a hole in the mechanism you thought was unbreakable and someone who found it out, may have access to your data without your knowledge [GaSp96:42].

5.1. Basic Cryptographic Services

"Those who claim to have an unbreakable cipher simply because they can't break it are either geniuses or fools." [Schb96:8]

Cryptography protects data transmitted over the communication lines or stored in the computer systems. Three principal objectives for such protection are confidentiality, authenticity and integrity. Confidentiality refers to denial of access to information by unauthorized individuals. It can be provided through access control or information-hiding approach [Ford94:178]. The latter approach is needed when information is transmitted over insecure channels. Authenticity refers to validating the source of the message, i.e., it was transmitted by a properly identified sender and is not a replay of a previously transmitted message. Integrity refers to assurance that a message was not modified accidentally or deliberately in transit, by replacement, insertion or deletion. A fourth service, which may be provided, is non-repudiation of origin, i.e., protection against a sender of a message later denying transmission.

Information transmitted over communication lines is vulnerable to passive (eavesdropping) and active wiretapping [Dend82:4]. Protection against eavesdropping is provided by rendering

[1] Shannon's papers were collected and published by IEEE Press in 1993 [Shan93]

information unintelligible for unauthorized parties. Defending against active wiretapping is more difficult; cryptography can be used to detect message modification but cannot prevent it.

Cryptography plays a central and the most critical role in the security applications. Some of these applications provide:

- Confidentiality of personal, military and commercial communications.
- Authentication of network users.
- Authentication of retail and banking transactions.
- Integrity of electronic funds transfers.
- Integrity of software and databases.

On some systems, cryptography is the only adequate way to safeguard the data. This is the case with banking cash dispensers. The information such as Personal Identification Number (PIN) and the amount of money to withdraw is transmitted over leased telecommunication lines to a central computer where records of the clients are kept. Cryptography is used to safeguard the message to computer and its response. It must clearly be done with a high measure of security to guard against wiretapping.

The design of the cryptosystem has to take into account security, speed and cost. The security is the first consideration and one of its measures is time required to break the system. The second criterion is crucial for bulk encryption. The cost includes initial investment and maintenance. With ever changing technology, very complex forms of cryptosystems are becoming practical. On the other hand, the same technology will help the attacker to search at high speed through very large numbers of possible transformations and in the cryptoanalysis. In balance, if cryptosystem designer and the user act prudently, the user is generally better off than the attacker as a result of computing technology, provided the attacker does not have enormous resources or endless time available [Schb96:9].

The attacks on the cryptosystem search to break it, recover plaintext from the ciphertext, or forge the message. Depending on how much information we assume the attacker has to work with, the attacks are classified as ciphertext-only, known-plaintext and chosen-plaintext. The chosen-plaintext attack is the hardest to withstand, since the attacker can submit any plaintext to cryptosystem and receive the corresponding ciphertext [Schb96:5-7]. Most cryptosystems are designed to withstand the chosen-plaintext attacks.

Claude E.Shannon [Shan49:669-670] lists the following five criteria as being the most important to judge cryptosystems:

1. High degree of secrecy.
2. Smallness of the key.
3. Simplicity of enciphering.
4. Low propagation errors.
5. Little expansion of message size.

Shannon showed that when encrypting natural language, these five criteria could not be met simultaneously. If any one of them is dropped, the other four can be met fairly well [Shan49:714].

There are two kinds of cryptography: private key cryptography and public key cryptography. In the private key cryptography, all trusted parties share the same key. In the public key cryptography, two mathematically related keys are needed, a public key and a private key.

5.2. Private Key Cryptography

Private key cryptography[1] is also known as conventional, one-key or symmetric cryptography. It deals mainly with the confidentiality and integrity aspects. There were some proposals on how to use private key cryptosystems for authenticity but none of them have seen practical implementation, so far. Conventional cryptography uses the same secret key to both encrypt and decrypt the message and the management of secret keys presents one of the major difficulties [ABHK96:545]. The key must be initially transmitted via secure channels so that both parties must know it before encrypted messages can be sent over insecure channels. This may involve couriers or other costly and not really secure methods. If keys are public, the task of key management may be simplified.

In comparison to public key cryptography, private key cryptography offers better performance and that is why it is used for bulk encryption. The disadvantage of private key cryptography is that, it is easier for untrusted parties to obtain copies of the key, because the secret key has multiple copies and has to be distributed through secure channels. Public key cryptography solves key distribution problem for conventional cryptosystems. In addition to distribution of secret keys, public key cryptography is used for authentication.

[1] When you refer to Figure 5.1, both keys for encryption and decryption are the same, for private key cryptography.

5.2.1. Data Encryption Standard

The most notable example of a conventional cryptosystem is Data Encryption Standard (DES). It was developed at IBM, as a modification of an earlier system known as LUCIFER, and published in 1975. In 1977, it was adopted as a standard for "unclassified" applications [Stin95:70].

It is a block cipher, operating on 64-bit blocks using a 56-bit key. Essentially the same algorithm is used to encipher and decipher. The key length is the primary weakness of the algorithm, which is half the size of the key used in the original Lucifer. The important characteristics of DES are its one-key feature and the nature of the operations performed during encryption/decryption. Both, permutations and table lookups are easily implemented, especially in hardware. Thus encryption rates of 1 Gbit/sec have been obtained. This makes DES an efficient bulk encryptor, especially when implemented in hardware. The security of DES is produced in a classical fashion: alternation of substitutions and permutations [Stin95:82-83].

The controversy arose as soon as DES was published, concerning whether it had weaknesses that intelligence organizations could exploit. A full strength DES product is still adequate for protecting commercial sensitive information but would not meet the strict requirements of a national security product review. In 1986, NSA announced that it would no longer certify DES for unclassified uses, less than ten years after DES was approved. This announcement somewhat substantiated the claims that NSA enforced halving the DES key length, to ensure that they themselves could break the ciphers.

Biham and Shamir have had some success on attacking the full 16-round DES [BiSh92]. However, these attacks require enormous time and data to succeed and this keeps breaking DES out of reach of almost anyone. Apparently, IBM designers knew about differential cryptoanalysis when they designed DES [Schb96:290].

5.2.2. International Data Encryption Algorithm

Another popular conventional cryptosystem is the International Data Encryption Algorithm (IDEA). This block cipher was developed at ETH in Zurich by James L. Massey and Xuejia Lai, and published in 1990. Schneier declared it as the most secure block algorithm available to the public [Schb96:319], at the time (1996). Early published papers on the algorithm called it Improved Proposed Encryption Standard (IPES) but they later changed the name to IDEA.

Similarly to DES, IDEA makes good use of confusion and diffusion. Confusion hides the way that the ciphertext statistics depend on the plaintext statistics. Defusion spreads the influence of a plaintext character over many ciphertext characters. Shannon suggested these two principles of cipher design to make cryptoanalytic attacks more difficult [Shan49:708-709].
While DES uses S-boxes for confusion, IDEA mixes three operations:

- XOR (exclusive OR)
- Addition modulo 2^{16}
- Multiplication modulo $2^{16} + 1$

While IDEA is designed to be more secure than DES, it has not been subjected to extensive cryptoanalysis. Biham and Shamir have been examining the IDEA cipher for weaknesses, without success. As this cipher continues to attract attack efforts from the cryptoanalytic world, confidence in IDEA is growing with the passage of time.

IDEA cipher has a 64-bit block size for the plaintext and the ciphertext. It uses a key size of 128 bits. It runs much faster in software than DES. Intel 66MHz 486 machines can encrypt 2.4 Mbps (megabit per second), while some VLSI chips encrypt data at the rate of 177 Mbps [Schb96:322-323]. Like DES, it can be used in cipher feedback (CFB) and cipher block chaining (CBC) modes. IDEA is designed to be more secure than DES against brute-force attacks and differential cryptoanalysis. It is patented in Europe, but can be used non-commercially without a fee.

5.3. Public Key Cryptography

Public key cryptography was invented[1] by Diffie and Hellman [DiHe76], in 1976, and independently by Ralph Merkle [Merk78]. The first realization of a public-key system came in 1977 by Rivest, Shamir and Adleman, who invented the well-known RSA cryptosystem. Public-key systems[2], also called two-key or asymmetric, differ from conventional systems in that there is no longer a single secret key shared by a pair of users. Rather, each user has a pair of keys: private and public. These keys are complementary; each key unlocks the code the other key makes. Knowing the public key does not help deduce the corresponding secret key. The keys are used together in a different combination, depending on whether confidentiality or authentication

[1] Ellis claims that public key cryptography techniques were known to UK cryptographers prior to these dates [Elli70].
[2] When you refer to Figure 5.1, key used for encryption is a public key of the receiver and key used for decryption is the secret key of the receiver.

is the goal. The public key can be distributed either through trusted parties or it can be widely published in a directory. The private key is kept secret and because no other copy is available, there is less chance than in conventional cryptography, for an adversary to gain a copy of the key [Stin95:114].

To provide message authentication, the sender's own secret key is used to encrypt a message, thereby "signing" it (see chapter 4.2. for more details). This creates a digital signature that proves that the sender was the true originator of the message and that the message has not been subsequently altered by anyone else, because the sender alone possesses the secret key that made that signature. Forgery of a signed message is infeasible and the sender cannot later disavow her signature [Schb96:37-38]. The confidentiality and authentication services are usually combined. First, a message is signed with sender's secret key then encrypted with the recipient's public key. The recipient reverses these steps by first decrypting the message with the secret key, then checking the enclosed signature with the sender' public key.

The applicability of public key systems is limited in practice by the relative low bandwidths associated with public key ciphers compared to their conventional counterparts. It has not been proven that time or space complexity is greater for public key systems than for conventional systems. However, the public-key systems, which have withstood cryptoanalytic attacks, are all characterized by relatively low efficiency. That is, using public-key systems for bulk data encryption is not feasible, at least for the present. On the other hand, there are two major application areas for public-key cryptosystems:

- Distribution of secret keys
- Digital signatures

The first involves using public-key systems for secure and authenticated exchange of data-encrypting keys between two parties. Data-encrypting keys are secret shared keys connected with a conventional system, used for bulk data encryption. This permits users to establish common keys for use with a system such as DES. Previously, users have had to rely on a mechanism such as a courier service or a central authority for assistance in the key exchange process. Use of a public-key system permits users to establish a common key, which does not need to be generated or revealed by a third party, providing enhanced security, greater convenience and robustness. Digital signatures are a second major application. They provide authentication, non-repudiation and integrity checks.

Public-key algorithms have the main advantage that the sender does not need to establish a secret key with the recipient, for communications to begin. Public-key algorithms, such as RSA, have become as popular and widely used as DES throughout the world, for integrity, confidentiality and key management.

5.3.1. Rivest, Shamir, Adleman Algorithm

Rivest, Shamir and Adleman are authors of the best-known and most versatile public-key cryptosystem - RSA. Over all proposed public key algorithms, it is the easiest to understand and implement. It supports both secrecy and authentication, and hence can provide complete and self-contained support for public key distribution and signatures[1].

It is believed that the security of RSA is based on the difficulty of factoring large integers. Hence, if the RSA is to be secure, it is necessary to carefully choose large numbers for the keys. Current factoring algorithms are able to factor numbers having up to 155 decimal digits. Unfortunately, nobody has an idea how to prove that factorization, or any realistic problem at all, for that matter, is inherently slow. It is conjectured that the security of RSA depends on the problem of factoring large number, but it has never been mathematically proven [Schb96:470].

Breaking RSA scheme cannot be more difficult than factoring, because fast factoring algorithm is an efficient cryptoanalytic procedure. However, this does not rule out finding an efficient algorithm for cryptoanalysis without finding corresponding factoring algorithm.

5.4. Vulnerabilities of Cryptographic Products

Major vulnerability exists if public keys are tampered with. In public key cryptosystem, the public keys do not have to be protected from exposure. In fact, it is better if they are widely disseminated. But it is important to protect public key from tampering, to make sure that a public key really belongs to whom it appears to belong to. This may be the most important vulnerability of a public-key cryptosystem. It might be so, in part because most novices do not immediately recognize it. When using someone's public key, you have to make sure that it has not been tampered with. A new public key from someone else should be trusted only if it is got directly from its owner, or if it has been signed by someone trusted.

[1] See 4.2.2. for the description of RSA algorithm.

To solve this problem, a widely trusted person could specialize in providing this service of distributing key certificates for others. This trusted person could be regarded as a "key server", or as a "Certifying Authority". Any public key certificate bearing the key server's signature could be trusted as belonging to the authentic owner. All users who would like to participate would need a known good copy of just the key server's public key, so that the key server's signatures could be verified.

5.5. Key Management Problem

The security of a cryptosystem depends, in large part, on the security of the methods and practices used to generate and distribute keys. For small systems, keys can be distributed by manually installing them. But this solution does not work for larger systems. There are two well-known approaches to the key-distribution problem in medium to large-scale systems: Key-Distribution Centers (for secret-key cryptography) and Certification Authorities (for public-key cryptography).

Key management involves a secure generation, distribution, storage and eventual disposal of keys. Key life cycle is a common problem for both conventional and public key cryptography, which can include the need for archiving, for example, seven years for income taxes (in Canada). Key maintenance is also required. For example, some keys may be lost or compromised. In addition, employee changes may make it necessary to cancel some keys and issue others. Key management has always been the hardest part of cryptography [Schb96:169].

5.6. Key Distribution Centers

In a conventional system, security is dependent on the secrecy of the shared key. For practical reasons, common keys for symmetric cryptosystems are exchanged via a centralized server – Key Distribution Center (KDC). A trusted entity in the network is designated to perform the KDC functions for a defined community of users. Each user individually has to establish a trusted relationship with the KDC, which has a duplicate of each of the user's master keys. The concentration of keys in one location (KDC) is a security disadvantage of conventional key management. The higher risk occurs because the KDC is in danger of an attack from an intruder and a single security breach by an intruder would compromise the entire system. Also, the cost or overhead of conventional key management is relatively high because of the need for the KDC to share all master keys. The higher overhead of a KDC occurs, in part, because of the bottleneck

effect. Since each pair of users needing a key must access a KDC at least once, the volume of activity would increase rapidly. In addition, each time a new secret key is needed, at least two communications are required for the user pair and the KDC. Furthermore, network availability could become a problem for the KDC based systems. Questions should also be asked concerning the capability for maintaining effective access control for the system containing secret keys. Examples of systems that provide this type of access control are security enforcing or trusted systems.

Other aspects of conventional key management are not unique to conventional cryptography. Manual key distribution must occur at least once for conventional cryptographic key management, after which automated distribution can take place. Master keys or *key-encryption keys* are manually distributed keys. These keys are used only to encrypt other conventional keys called working keys, or *data keys* [Schb96:176].

The increased application of communication technology in international e-commerce has accelerated the need for security in data network communications. These communications support global interconnectivity and distributed operations, thereby introducing security risks. New developments in communication protocols offer promise of providing solutions to reduce certain security risks. Joining large number of KDCs to support international e-commerce is a complex undertaking and will require careful planning and cooperation between KDCs.

5.7. Public Key Infrastructure

A Public Key Infrastructure (PKI) is a broad set of technologies that are utilized to manage public keys, private keys and certificates. The deployment of a PKI solution should not be taken lightly as there are major issues involved with scalability and interoperability.

In a secure environment, public keys can be stored in the Directory. A certificate is used to bind a public key to an individual or other entity and is issued by a certification authority (CA), which can be any trusted administration. A certificate contains at least a key and a name, but it should also contain the expiration date, the name of the certifying authority and the serial number of the certificate. Above all, it contains the digital signature of the certificate issuer. If a secret key is compromised or becomes otherwise invalid before its expiration date, it can be added to a

certificate revocation list (CRL), which should always be checked before a public key is used. CA's form a hierarchical structure, thus allowing cross-certification.

There are two distinct public key distribution models: web of trust and CAs. Pretty Good Privacy (PGP) is the most apparent example of using the first model, while Privacy Enhanced Mail (PEM) employs the second. In short, PGP (see Appendix E) is a distributed network of individuals [Zimm95:21-34]. In this web of trust, you can certify public keys of your friends and you do not have to trust but them. A user cannot verify the validity of every other PGP key, and one probably will not ever trust most people, so PGP is not very scalable. Even though this is the way PGP is used today, there are no technical reasons why the public keys could not be managed by the authorities. Another problem with this model is that conspiracies are possible with small webs: a group of users could certify dummy certificates and thus impersonate invalid users. At present, PGP public keys are distributed through e-mail, bulletin boards, or special key servers; there are also plans to store them in the X.500 Directory.

PEM public key management is hierarchical (bureaucratic): keys are verified at trusted CAs, and it is guaranteed that any two users have a common CA above them in the hierarchy. You have to trust your CA and in fact, also other CAs, since keys can be verified transitively, by a chain of CAs. PEM public keys can be distributed by many means, such as e-mail or X.500 Directory look-up. In theory, this seems nice, well organized, so it is not a big surprise that commercial and governmental applications are very interested in PEM. In practice though, the lack of infrastructure and public directories has been the main obstacle in proliferation of PEM.

Today many commercial organizations are developing PKI but it is not known whether these developments will address scale, performance, and diversity, distributing nature, sensitivity and a high consequence of some of the computing networks. The areas in which such initiatives are concentrated and will likely succeed include banking, e-commerce, long-term retention of signed documents and PKI interoperability.

PKI has to address archives for keys to enable long-term authentication of digitally signed documents and to access encrypted sensitive data. To support high consequence secure applications and emergency situations, PKI has to support multi-keyed Certificate Authorities. Research needs to be done on creating security gateways for legacy applications, so these applications could participate in secure distributed computing. Building such gateways will be

challenging because the legacy applications were not "security aware" and some of them exist in binary format only. Even if there is a source code available, the changes should be very superficial because of lack of documentation and bad software life cycle practices (spaghetti code). Any major change to such applications would run a risk of turning them unusable.

Mary Thompson et al. [THMH99] propose a model for sharing resources in widely distributed environment, based on attribute certificates instead of Access Control Lists (ACL). PKI undertakings must consider the implications of such environments on PKI architecture, find the implementation that is flexible, scalable and accurately reflects the existing policy: authority, delegation, and responsibility present in these environments.

In distributed environments, the resources such as: data, instrument, computational and storage capacity and communication channel have multiple stakeholders and each stakeholder will impose use-conditions on the resources. Traditionally, access to such resources, is controlled centrally by the administrator. This means that administrator must be trusted by all stakeholders and that the administrator is potentially a bottleneck to rapid updates to the policy. To change access control requirements for a stakeholder, the administrator must authenticate the request and only then make the appropriate change. Another problem in distributed environment is that there might be multiple principles managing the access control policy for a single resource. For example, executing a proprietary code on a supercomputer might require meeting two separately administered policy requirements: computer owner and author or owner of code.

It is common that principles, in large scientific experiments are geographically dispersed and multi-organizational. Accordingly, without reliable, widely deployed and easily administered access control mechanisms, it will not be feasible to administer a secure collaborative environment. The access control mechanism must allow secure, distributed management of policy-based access to resources and provide transparent access for authorized users and strong protection from intruders. These access controls must operate in an environment where stakeholders, users, and system/resource administrators may never meet face to face.

PKI is seen as the most effective and viable technology for secure transfer of information. It scales well with the number of users, organizations and sites. PKI is being adopted as a security basis for e-commerce. Today, there is no technological barrier in PKI deployment. Rather there

are political obstacles, agreements on common CA policies that put brakes on wide implementation of PKI.

5.7.1. Large-scale Deployments

In the early 1970s, the U.S. DoD developed Secure Telephone Unit, First Generation (STU-I) to secure telephone system. It was followed by STU-II that relied on KDC-based key management and served about 40,000 users. In the early 1980s, STU-II was superseded by STU-III that uses public certificates and serves more than 500,000 users. In the commercial sector, probably the largest KDC deployments are based on the Kerberos. OSF/DCE[1] is an example of such system.

Pretty Good Privacy (PGP) and Lotus Notes probably represent the largest deployed public-key systems. There are estimated 10 million certificates associated with Lotus Notes users that are distributed over many organizations. Like OSF/DCE, Lotus Notes is usually employed on an organizational basis. Because PGP does not offer a formal CA structure, it is difficult to establish exact size of PGP deployment.

Web browsers employ certificates, usually issued by public CAs, in using the secure socket layer (SSL) protocol to establish encrypted, one-way[2], authenticated communication paths. This deployment of public-key cryptography has been crucial for providing the secure paths necessary to send credit card numbers and other sensitive data in support of e-commerce [DdDp98:386]. But the biggest demand for certificates might come from secure e-mail, such as Secure/Multipurpose Internet Mail Extensions (S/MIME). Deployment of the Secure Electronic Transaction (SET) protocol for credit card transactions over the Internet has been slower than expected, but eventually it, too, could cause millions of certificates to be issued to the existing users of credit cards.

5.7.2. PKI Promoters

There are many dimensions of growing interest in building PKI infrastructure. With HPCs outgrowing the number of traditional desktop computers, booming e-trading, e-commerce and e-banking; there are many commercial organizations that see PKI infrastructure as a necessity to

[1] Open Software Foundation/Distributed Computing Environment - an industry standard for UNIX-based systems.
[2] SSL permits two-way authentication, through the use of client certificates, but this option is not often used.

fulfill their obligations toward their customers. Traditional stock exchanges have to move fast to keep up with technology changes and the competition from electronic computer networks (ECN) that enable users to trade in real-time. The banks and other financial institutions are embarking on programs with goals to establish PKI that will secure e-commerce transactions among the institutions and between individuals and institutions. In September 1999, Royal Bank of Canada has joined an international group of banks and banking associations in the establishment of a Global Trust Authority (GTA) to facilitate secure worldwide payments over the Internet. The goal of the GTA is to provide reliable online identification and authentication of both participants in a transaction, regardless of their location. The GTA also provides a mechanism for linking territorial/national Public Key Infrastructure (PKI) operations.

Financial institutions are not the only ones interested in building PKI; governments, government departments and intelligence agencies sponsor, promote and oversee such initiatives. Government of Canada (GoC) launched a PKI project in December 1995, involving six departments with the goal to provide more efficient services to Canadians, facilitate secure electronic commerce and to better protect confidentiality and privacy of information used within the federal government. The full implementation was planned for 1998.

There are still some applications which requirement might not be fulfilled by commercially or government sponsored PKI initiatives. These applications include distributed supercomputer programs, secure multimedia collaboration tools and sensitive distributed databases. They will require different secure type of authorization schemes and interprocess communication (IPC). Some of the standards and proposals for secure IPC include GSS-API, SSL, secure CORBA, and custom approaches such as Zipper. None of these provides interoperable security and more research is needed to accomplish scalable security for widely distributed supercomputer applications that require low latencies and high bandwidth in wide area network (WAN) environments. Since these issues apply to specific, high-end computers and applications, the organizations working in such environments have to fund research on PKI requirements specific to these organizations but need to cooperate with commercial security initiatives in order to create interoperable, global PKI.

5.8. Remarks

A significant fraction of security problems cannot be solved without the use of strong cryptography, although challenges from performance to standards to interoperability have limited

its widespread deployment. Encryption is a fundamental tool for the protection of sensitive information. The basic crypto services include: confidentiality, integrity, authentication and non-repudiation.

The single most important vulnerability of public key cryptography is tampering with the public keys. Hence, it is important to set up PKI and to establish standards enabling PKIs in different security domains to talk to each other. Another problem is that there exists no proof of strength of the existing public key cryptography algorithms. We know brute force methods of breaking them and computational requirements for such attacks, provide as an assurance that these attacks are unfeasible today and will remain this way, in the foreseeable future. The obvious danger is that better algorithms for cryptoanalysis already exist or will be developed in the near future and that technology will render brute force attacks feasible.

Cryptography is a particularly interesting field because a lot of work is done in secret. The irony is that today, secrecy is not the key to the strength of a cryptographic algorithm. Regardless of the mathematical theory behind an algorithm, the best algorithms are those that are well known and well documented. In fact, time is the only true test of good cryptography, since any cryptographic scheme that stays in use year after year, is most likely a good one. As a result, users should be wary of products that use a proprietary cryptography scheme, because the secrecy of the algorithm offers no advantage. This security through obscurity is doomed to fail.

Public-key cryptography is considerably more (computationally) expensive to use than secret-key cryptography. Therefore, most cryptographic systems that make use of public-key cryptography are in fact hybrids. For confidentiality, public-key cryptography is employed to encrypt a secret key that, in turn, is used with secret-key cryptography to encrypt data. Still, using this approach requires cryptographic algorithms that keep pace with communications transmission speeds.

Cryptographic services are very effective means for solving certain security problems in geographically distributed systems. Evolving Application Programming Interfaces (APIs) for cryptographic services will promote greater use of such services.

Regardless of whether a conventional or public key cryptosystem is used, it is necessary for users to obtain keys. Key life-cycle management is a common problem for both conventional and public key cryptography. For symmetric cryptosystems, keys are exchanged via Key Distribution

Centers (KDC). Because of concentration of all private keys in one node, the cost of successful penetration is higher. Certification Authority (CA) guarantees the link between user and cryptographic key by signing a document that contains user name, key, name of CA, expiry date, etc. In this scheme, any two users have a common CA above them in the hierarchy.

A Public Key Infrastructure (PKI) is a broad set of technologies that are utilized to manage public keys, private keys and certificates. Given the minimal experience to date with PKIs, many aspects of PKI technology deserve further research. This research should focus not only on the issuer CA aspects of PKI, but also on the client or consumer side. Most applications that make use of certificates have poor certificate-management interfaces for users and system administrators.
PKI systems have a root authority that stores a root key, which is the basis for all the encryption within a PKI. Generally, this root key is stored on a separate device that cannot be tampered with. Because of its importance, it is essential that the device be extremely secure.

Major challenges of PKI include scalability, performance and high consequence systems, as well as legacy applications, and sharing resources in widely distributed computing that have multiple stakeholders. Another important issue is the storage of certificates and certificate revocation lists.

Although PKI technology is intended to serve very large populations with diverse administrative structures, issues related to timely notification of revocation, recovery from compromise of CA private keys and name space management, all require further attention.
Today, we have a vast array of alternatives for deploying PKI technology. Vendors realize the enormous commercial potential and the need to move along quickly in PKI product development. Since development of common standard is going very slowly, different business groups have set up their proper PKIs. Some of the issues, such as certificate revocation, can become a real interoperability problem. At present, PKI is immature and burdensome, because of the lack of interoperability and the amount of administration and coordination required. Part of the renewed cooperation among vendors is spurred by increased user demand. If PKI is to succeed, PKI products from different vendors have to work seamlessly together.

Chapter 6

Network Access Control

"As a user of the Internet, you are fortunate to be tied into the world's greatest communication and information exchange - but not without a price. As a result of this connection, your computer, your organization's network, and everywhere that network reaches are all vulnerable to potentially disastrous infiltration by hackers." [ChBe94]

A major advantage of the network is that geographically dispersed collaborators can exchange information and share the computing resources. Certainly, the substantial benefits can be gained from connecting computers through the network. On the other hand, the connected computer represents much more promising target for the attacker, because they increase the potential gains from a single penetration. We need to allow users of a corporate network to access a public network, such as the Internet and at the same time we want to make available to the public some of the company's services and products. Network access control techniques must be used to limit the availability of the network resources such as sub networks, physical and logical channels and network services, to authorized users. Since the networks assumed a central role in the information systems, the interest is growing in network access control mechanisms.

Traditionally, organizations built and deployed mission-critical applications over private local and wide area networks, with the known infrastructure and tightly controlled access. The end result was a private data communications infrastructure that had somewhat predictable application availability, performance and security. Today, many mission-critical applications are being deployed across the Internet. Obviously, with the increased connectivity, you increase your exposure and therefore your potential security risks. A stand-alone personal computer (PC) with sensitive information is vulnerable only to people who can gain physical access to it[1]. Once you connect PC to the Internet, you drastically increase its exposure and vulnerability. To shield organization's resources from the attacks launched through the Internet, firewalls, VPN and guards are used.

[1] Good physical controls, including electromagnetic shields are sufficient for stand-alone systems.

6.1. Closed User Groups

Closed User Group (CUG) is a group of users that are permitted to communicate with each other but not with users outside the group. Often, subscribers of virtual circuit data networks such as X.25, frame relay and asynchronous transfer mode (ATM), are members of (one or more) CUG. These groups are managed by network administrator and provide control mechanisms to assure that only members of CUG can communicate with each other. This control is based on network authentication[1] and in some cases other information is added to access control list (ACL). For example, reverse charging or inbound versus outbound call initiation might be stored in ACL.

Because CUGs are frequently limited to one network, employing single network technologies and managed by a single administration, they are becoming irrelevant as networking migrates to the Internet. Virtual Private Networks (VPN) can resolve the limitations of CUG.

6.2. Firewalls

Network firewalls are combinations of computer hardware and software intended to keep unauthorized users from penetrating a company's internal network and gaining access to sensitive information. They are typically deployed at the boundary of trusted and untrusted computer networks. Firewalls are designed primarily to enforce a security policy by inspecting the data stream. They screen and filter all connections coming from the Internet to the protected (corporate) network, and vice versa, through a single, concentrated security checkpoint. Safe or presumed safe messages pass through the firewall, while others are blocked.

There are several models and configurations of firewalls, but the basic idea behind it is always the same. You need to allow users from a protected network, such as your corporate network, to access a public network, such as the Internet, and at the same time, to make available to this public network a number of the services and products of the company. Of course, connecting the company LAN to the Internet exposes it to the potential attacks from the Internet users. Hence, when you plan the protection of your network from the Internet threats, you start by thinking about firewalls. However, before you buy a firewall, you must define how and which services and information you will make available to the Internet users.

[1] Identities correspond to network layer address.

The firewall is only part of a broader area in protecting your network. In order to secure your corporate network, you must define your idea of a network perimeter. You need to determine what resources must be protected, develop a security policy and establish mechanisms to enforce this policy. It is critical to decide which services will be available remotely. You might need to deny rlogin, telnet, (t)FTP, SMTP, NFS and other RPC services. Another important decision involves the choice of the operating system (OS) platform, on which the firewall is based. The firewall can be implemented on UNIX, Windows NT (Windows 2000), DOS or a proprietary OS. You must carefully select the platform, as it will define all future projects, level of security and consequently the security policy being developed.

If you are going to develop a policy to deal with Internet access, Web administration, and electronic services in general, it must be flexible. The policy must be adaptable for the following reasons:

- As the Internet changes at rapid rate, services offered through the Internet also vary. With that, company's needs, will change too, so you should be prepared to edit and adapt your policy accordingly without compromising security and consistency.
- The risks your company faces on the Internet are not static either. They are continuously growing. Therefore, you should be able to anticipate these risks and adjust the security policy accordingly.

The security policy reflects the organization's reasons for connecting to a public network. You cannot design this policy without understanding the firewall capabilities and limitations, as well as the threats and vulnerabilities associated with TCP/IP. When deciding on how to provide the services to the network users, there are two basic approaches:

- Permit any service unless it is expressly denied
- Deny all services unless it is expressly permitted

These two approaches represent a fundamental choice between open and closed systems.

In theory, firewalls should not be necessary. If a single computer can be hardened against attacks, then, in principle, all computers can be. And if all computers on a network are hardened, then there is no need for an additional perimeter defense. In practice, firewalls have multiple advantages. A firewall can actually be less expensive for an organization, because all (or most) modified software and additional security software can be located on the firewall system, instead of being distributed on each machine. In particular, one-time-password systems and other add-on

authentication software can be located at the firewall rather than be on each node that needs connection to the Internet. It is easier to administer software on one or a small number of firewalls than to do so for the entire collection of workstations, PCs and servers. Physical access to a computer's console might be necessary for setting or checking its configuration, for example. Besides, a firewall can provide a network security administrator with a single point of policy control for an entire network.

When procuring or building your own firewall, you should evaluate how well it meets your needs [ChBe94:116]. Here is a checklist of features and attributes of the firewall:

- A firewall should support your security policy, not force one.
- A firewall should employ advanced authentication mechanisms or should be expandable to accommodate them in the future.
- A firewall should be flexible, so it can accommodate changes to your company's security policy, brought by the organizational changes.
- It must employ filtering techniques that allow or disallow services to specified servers as needed. The filtering language must be easy to program and capable of filtering as many attributes as possible, including source and destination IP addresses, inbound and outbound interfaces, protocol type, and source and destination TCP/UDP ports.
- It should contain the capability to concentrate and filter dial-in access.
- A firewall should use proxy services for services such as FTP and telnet so that advanced authentication measures can be employed. If services such as NNTP, X, HTTP, or Gopher are required, the firewall should contain the corresponding proxy services.
- It should centralize SMTP access in order to reduce direct SMTP connections between site and remote systems. This will result in centralized handling of site e-mail.
- A firewall should accommodate public access to the site in a secure way. That is, it should protect public information servers and isolate them from site systems that do not require the public access.
- It should contain mechanisms for logging traffic and suspicious activity.
- A firewall that is built on top of an operating system should employ a secured version of the OS, with other security tools as necessary to ensure firewall server integrity.
- A firewall and any corresponding OS should be updated and maintained with patches and other bug fixes in a timely manner.
- It should be simple in design so that it can be understood and maintained. Ideally, it should be developed in a manner that its strength and correctness are verifiable.

6.2.1. Limitations

A firewall is not infallible, its purpose is to enhance security, not guarantee it. If you have very valuable information in your LAN, your Web server should not be connected to it in the first place. You must be careful with applications that allow you access to your Web server from within the organization or vice versa. The firewalls can be a great aid when implementing security for a site and they protect against a large variety of attacks. But it is important to keep in mind that they are only one part of the solution. They cannot protect your site against all types of attacks.

Firewalls can enforce only policies defined in terms of restrictions on inbound and outbound traffic. There are limits to what can be accomplished using these restrictions. For example policy preventing an insider from communicating with the Web server can be accomplished by restricting port 100. An insider can set up a Web proxy server that monitors port 1000 on a machine outside the firewall. Now an insider can surf the Web using the outside proxy. More generally, there is nothing a firewall can do about threats coming from inside the organization. Also, using firewalls is pointless when paths exist to the outside that bypass those firewalls; an authorized link to some outside organization, an unprotected modem pool, or careless employee dialing out to an Internet Service Provider (ISP). If you need to use modems in your office, there are several measures you can take to beef up security of your modem(s) – password modems, callback setups, encrypting modems, caller-ID and Automatic Number Identification (ANI) schemes [GaSp96:419-420].

Packet filtering was always a simple and efficient way of filtering inbound unwanted packets of information by intercepting data packets, reading them, and rejecting those not matching the criteria programmed at the firewall. Unfortunately, packet filtering is no longer sufficient to guarantee the security of a site. Many are the threats, and many are the new protocol innovations, with the ability to bypass those filters with very little efforts. Hence, the decision regarding what protocols are allowed to pass through the firewall is critical for success. With the growing number of protocols allowed to bypass the firewall, the chances of exploiting a flaw in one of them are rising.

When deciding to implement a firewall, first you will need to decide on the type of firewall to be used. There are four basic types of firewalls: packet filters, circuit relays, application gateways and dynamic (stateful) packet filters. The first three correspond to layers of the protocol stack, the

fourth incorporates features of both network and application layer systems. Some of the firewall limitations come from the protocol layer at which the firewall operates. Attacks conveyed using protocol layers higher than the one at which the firewall works cannot be blocked by firewall.

Other limitations of firewall come from the way they are built. Since most of the firewalls are implemented as applications on top of standard operating system (OS), all are vulnerable to attacks directed at the underlying OS [ChBe94:82]. Some firewall developers strip out portions of OS that are considered vulnerable, but given the size and complexity of modern OSs, only limited hardening can be achieved this way. An alternative is to build a custom OS, but it may introduce other vulnerabilities that have not been detected through experience of a large community of users.

A lot of firewalls employ application proxies and often these proxies use existing application code as a base. This way, vulnerabilities in the base application code may be preserved in the proxy. Furthermore, proxies need to be modified (kept in synch) when the application changes or a new application is developed.

The efficacy of a firewall is also limited by the use of end-to-end cryptography. It is impossible for a firewall to inspect the contents of an encrypted packet, hence encrypted packets cannot be blocked. Other firewall services such as address translation cannot be done with encrypted packets. The usual solution is to terminate cryptographic associations at the firewall. In some cases, multiple levels of cryptographic protection are used, with an outer layer permitting passage through the firewall and the inner layer being end to end.

6.3. Virtual Private Networks

The term Virtual Private Networks (VPN) is one of the most overused buzzwords in the high-tech industry today. It has been used so carelessly and has so many definitions that many companies can claim that their products are actually VPN [Stei98]. Proponents claim that VPN can solve many issues, from providing remote users secure access to corporate Intranets, to securing corporate data for transport over the Internet. So, what exactly is a VPN?

The desire to use the Internet for business and the risk factors associated with it, have given rise to VPN. Informally speaking, a VPN is a private network constructed within public network

infrastructure, such as the Internet. VPNs use a public network for both, data and voice, and create an illusion[1] of a network that is devoted exclusively to subscribers of the VPN. The Centrex service offered by local telephone companies is an example of VPN for voice and is usually implemented through administrative controls in central office (CO). VPNs in data networks could be implemented in a similar way, but would be vulnerable to wiretapping on underlying physical networks. To prevent wiretapping, VPNs use encryption and tunneling.

Most of today's VPNs are IP-based networks (usually the Internet) and are built to achieve one or more of the following goals:

- Connect users securely to their own corporate network (remote access)
- Link branch offices to an enterprise network (intranet)
- Extend existing computing infrastructure to include partners, suppliers and customers (extranet)

The idea of VPN is to extend trust relationships across an economical public network without sacrificing security. Ideally, a VPN should behave similarly to a private network; it should be secure, highly available and have predictable performance.

The first packet network VPN technology was developed by Bolt, Beranek and Newman in mid-1970s and was called private line interface. This scheme was approved to protect classified data for transmission over the ARPANET, creating a VPN for a set of DoD subscribers. Since then, many other proposals have been developed. Examples of such technologies include: BCR and the Blacker (KDC-based VPN systems), the Xerox XEU and Wang TIU (manually keyed LAN VPN systems), the Motorola NES and Caneware (certificate-based, Internet VPN systems).

In the commercial arena, VPNs have been deployed for use with X.25 and ATM networks, as well as those for the Internet devices. Typically, the separate technologies used to provide confidentiality, integrity and authentication in a given implementation, are grouped into a broad VPN protocol. There are three widely used protocols: Internet Protocol Security (IPSec), tunneling and SOCKS version 5 (Socks5).

The IPSec protocol is becoming the de facto standard for VPNs. This protocol comprises a set of authentication and encryption protocols, developed by the Internet Engineering Task Force (IETF) and designed to address the inherent lack of security for IP-based networks. It is designed

[1] That is where "virtual" part of the name comes from.

to address data confidentiality, integrity, authentication and key management, in addition to tunneling. The IPSec protocol typically works on the edges of a security domain. Essentially, IPSec encapsulates a packet by wrapping another packet around it. It then encrypts the entire packet. This encrypted stream of traffic forms a secure tunnel across an otherwise unsecured network. The comprehensive nature of this protocol makes it ideal for site-to-site VPNs. Since IPSec operates at the Internet layer, it can protect traffic across any local area network (LAN) or wide area network (WAN) and can be terminated at end systems. Access control in IPSec is based on cryptographic authentication. The granularity of access control is determined by local policy and can range from subnet level protection to per user and per application controls. IPSec also includes an optional anti-replay facility, which prevents certain forms of denial-of-service attacks. However, IPSec cannot protect against denial-of-service attacks aimed at the switching and transmission media that implement VPN, which is also true for any other VPN implementation[1].

Point to Point Tunneling Protocol (PPTP) and Layer 2 Tunneling Protocol (L2TP) are the most popular tunneling protocols. PPTP provides remote users access to a corporate network over the Internet. Network layer protocols, such as IPX and NetBEUI, are encapsulated by the PPTP protocol for transport over the Internet. Unlike IPSec, PPTP was not originally designed to provide LAN-to-LAN tunneling. L2TP is a proposed successor to PPTP and is a hybrid of PPTP and L2F protocol.

Socks5 is a circuit-level proxy protocol that was originally designed to facilitate authenticated firewall passage. It provides a secure, proxy architecture with extremely granular access control, making it an excellent choice for extranet configurations. SOCKS version 5 supports a broad range of authentication, encryption, tunneling and key management schemes, as well as a number of features not possible with IPSec, PPTP or other VPN technologies. SOCKS v5 provides an extensible architecture that allows developers to build system plug-ins, such as content filtering (denying access to Java applets or ActiveX controls, for example) and extensive logging and auditing of users.

[1] A VPN cannot defend against attacks directed at the resources used to build the VPN.

6.3.1. Quality of Service

Most VPN technologies today do not address performance and availability issues. It is because the majority of VPN solutions exist on client machines and gateway servers at the extreme ends of the communication path. They just cannot consistently affect the performance of the network components in the middle. Unfortunately, this is exactly where the Internet fits into the architecture. Any cost savings that a VPN provides can be quickly negated if users are forced to sacrifice quality of service (QoS) beyond certain limits. Until a standard QoS mechanism becomes ubiquitous, end-to-end performance guarantees will be hard to implement. As a partial remedy, several Internet Service Providers (ISPs) are offering managed VPN services, which combine security capabilities with QoS guarantees. Some ISPs guarantee availability of 99.9% and round-trip latency of less than or equal to 125 milliseconds. This type of service can be an excellent choice for site-to-site connectivity, but can only apply to traffic within the network controlled by the ISP. Once it passes onto another ISP's portion of the Internet, all bets are off.

6.4. Guards

A Guard is usually used to enable connectivity that is normally prohibited because of information confidentiality. A combination of hardware and software components designed to allow secure Local Area Network (LAN) connectivity between segments operating at different security classification levels: *high* and *low*. The vast majority of guard implementations use a dual network (usually Ethernet) approach, which physically separates the high and low sides. In addition, most guard processors are high assurance platforms that host some form of trusted operating system and trusted networking software. If guards pass information *high* to *low* that is termed a *downgrade* operation. Information flowing the other way is called an *upgrade*. While first type of information transfer may require a review cycle, the latter requires a verification of the integrity of the *low* source system and network.

Automated filters within guards ensure that all traffic conforms to specified criteria, including field-by-field restriction on types or values. As traffic formats become more flexible and field values have greater range, it becomes less likely that an automated filter can correctly detect all prohibited traffic. Hence, some designs send all questionable traffic for a human review. Such review tends to be monotonous work and it is questionable whether people are more efficient in filtering than the machine processes. Despite limitations of guards, they are widespread access control mechanisms for electronic information systems in use today by the military.

6.5. Remarks

Networks pose two problems:

- Where should I place my security perimeter to filter adversarial inputs?
- How can I turn insecure networks into secure ones?

Placing the perimeter on the outside of an intranet means using a firewall. Making insecure networks secure probably means using a VPN. Problems exist because people place unreasonable demands on these two partial solutions.

Many organizations delegate their network access controls to a surrounding firewall. One of the advantages of the firewall is central administration of security policy and hence, cheap maintenance. There are different types of firewalls that include packet filters, circuit relays, application gateways and stateful packet filter. While the first three correspond to layers of protocol stack, the last one incorporates the features of both network and application layer systems. Every type of firewall has its advantages and disadvantages. For example, packet filters are transparent to users and offer good network performance. On the other hand they are vulnerable to IP spoofing attacks. Application gateways allow you to authenticate the user, but need a proxy for each type of supported service.

The best firewall solution for a company is one tailor-made for its needs. Unfortunately, tailor-making a firewall product is not a viable solution to most of the companies without internal Internet security experts or without relying on consulting services. When setting up the firewall, you need to decide which services you support. Some services with high potential risks, should not be supported via the firewall. These include the NFS, NIS, RPC services and SNMP.

In general a firewall could only protect the network from attacks that cross the firewall host. It cannot protect against malicious insiders. The firewall is useless if the attacks come from the internal network or an external host that bypasses the firewall.

The disadvantages of firewalls include cost, complex configuration and the limitations imposed by end-to-end cryptography. Firewalls are stressed when they need to cope with new protocols and are imperfect when they are asked to filter encrypted packets. Although they do offload some of the work and simplify management, organizations focus too much on a firewall's capabilities and feature set while ignoring their own requirements. You cannot choose the right firewall until you have performed a requirements analysis.

Certain firewall vulnerabilities come from the underlying operating system (OS). Some developers strip out portions of OS that are considered vulnerable, but given the size and complexity of modern OS, only limited hardening can be achieved this way. Firewalls need to be enhanced to support sophisticated security management protocols and negotiation of traffic security policies across administratively independent domains. With the development of more advanced network applications, we need to understand better how to define and enforce traffic policies at this level.

A Virtual Private Network (VPN) is a private network constructed within public network, such as the Internet. Some of the goals of VPN include remote access for employees, intranet and extranet. The main advantage of VPN is low cost. Ideally, a VPN should behave similarly to a private network; it should be secure, highly available and have predictable performance.

Many VPN technologies already exist, with more being developed, marketed and deployed each day. Some products are based on standards (usually emerging standards); others are proprietary[1]. Some address very specific requirements, such as secure remote access over the Internet for mobile users, while others focus more on secure LAN-to-LAN connectivity. The problem the security engineers face is to find compatible products, so that firewalls at different sites can talk to each other.

Some VPN products employ manual key management, which prevents large-scale deployments. The adoption of IPSec was expected to not only increase the number of products with VPN capabilities, but also ensure interoperability and promote the use of automated (certificate-based) key management protocols. Disappointingly, despite the fact that the majority of VPN vendors are implementing IPSec in their solutions, there are still interoperability issues that exist across different vendor's implementations.
There are three widely used protocols: Internet Protocol Security (IPSec), tunneling and SOCKS version 5 (Socks5). IPSec is becoming the de facto standard for VPNs. Point to Point Tunneling Protocol (PPTP) and Layer 2 Tunneling Protocol (L2TP) are the most popular tunneling protocols.

PPTP is built in to (Microsoft Windows) NT 4.0 and the client is a free add-on to Windows95. Microsoft's implementation of PPTP has been found to have several problems that make it

[1] Unfortunately, a typical VPN employs proprietary protocols.

vulnerable to attacks and it also lacks scalability in that it only supports 255 concurrent connections per server. The low cost and integration with NT and Windows 95, however, makes PPTP a viable remote access solution where multi-protocol access is needed, heavy-duty encryption and authentication is not needed, and a Microsoft-only solution is appropriate.

When SOCKS is used in conjunction with other VPN technologies, it is possible to have a more complete security solution than any individual technology could provide. A user may, for example, incorporate IPSec and SOCKS together. IPSec could be used to secure the underlying network transport, while SOCKS could be used to enforce user-level and application-level access control.

Much work remains to be done to facilitate VPN deployments. Support for dynamic location of security gateways, accommodation of complex network topologies, negotiation of traffic security policies across administratively independent domains and support for multicast communications are topics requiring additional work. Further, better interfaces for VPN management will be critical for avoiding vulnerabilities introduced by management errors.

Applications deployed across the Internet today are increasingly mission-critical, whereby poor performance or lack of security can jeopardize business success. VPNs can play a major role in ensuring that these risks are mitigated. By addressing security and performance issues, a VPN can be a viable alternative for dedicated, private network links. Understanding the myriad VPN solutions can help organizations build infrastructures that will support their tactical business needs today, as well as their strategic business needs for tomorrow.

Chapter 7

Availability and Denial of Service

"Denial of service occurs when an entity fails to perform its proper function or acts in a way that prevents other entities from performing their proper functions." [CSA92:22]

Availability of information system means that services are obtainable when they are requested. As organizations become increasingly dependent on computerized systems, it becomes more critical to ensure that these systems are available when needed and that the confidentiality and integrity of the data they process are maintained. Almost always, these security principles are not compatible. That is promoting data integrity, for example, will probably cause loss in service performance and flexibility.

Here is availability definition from [CSA92:2]:
Availability is the property of being accessible and useable upon demand by an authorized entity.

The way a system is designed, built and distributed determines how likely it is to fail or otherwise deny service. High-availability systems are needed for telephone networks, electrical networks, air traffic control, hospitals and military installations. These systems must provide high availability, if not just for consideration of potential liability or loss in public relations.

Denial of service in [CSA92:3] is defined as follows:
Denial of service is the prevention of authorized access to resources or the delaying of time-critical operations.

Denial of service is the least researched security property, even though there is adistinctive lack of security mechanisms to handle this problem. Denial of service can range from abusing or stealing computer resources, to loss of messages through deliberate overloading, to total loss of service. We will distinguish between malicious and inadvertent denial of service. The difference among the two is the motive, not the end result. Malicious denial of service is a deliberate destruction or degradation of availability. The inadvertent denial of service is caused by user error, by negligence or ignorance.

7.1. Attacks

Saturating servers and resources to degrade the system availability is one form of denial of service attacks. Disabling a critical subsystem in a widely distributed environment by crashing it or preventing it to respond is another one. The latter attacks are probably easier to detect, but not necessarily easier to submerge. More subtle and hence, more difficult to detect attacks involve exploration of system features such as automatic *roll back* in response to error conditions. The attacker can create an error condition that causes some legitimate updates to be rolled back, without the legitimate client knowledge of such lost update, effectively denying service.

Denial of service attacks, are difficult to avoid. Networks can be flooded with traffic, packets can be routed to the already overwhelmed routers by contaminating the Internet Domain Name Service (DNS) caches and dial-up ports can be monopolized to prevent users from even getting in [ABHK96:498]. The server operating systems are usually robust enough to prevent looping programs from monopolizing computing service, but vulnerabilities still exist. For example, an attacker could disable OS interrupts as to deny or badly degrade service. Other attacks include viruses, trojan horses and worms that can cripple the system or totally shut it down [Denp91].

7.1.1. SYN Floods

SYN flooding[1] is a network-based denial-of-service attack on Internet Protocol (IP) based networks. The attacker sends many Transmission Control Protocol (TCP) connection requests to a victim machine. Because the connections cannot be completed, they cause the connection queues to fill up, thereby denying service to legitimate TCP users [Cisc2].

The Internet Protocol (IP) is the standard network layer protocol of the Internet that provides an unreliable, connection-less, best effort packet delivery service [WrSt94:245]. TCP ensures reliable communications for applications and resides between IP and the application layer. It guarantees that datagrams[2] will be delivered in order, without errors and without duplication. To establish a connection between a source and destination hosts, TCP uses so called three-way handshake process. See Figure 7.1. First, source sends SYNchronize datagram. Then target host replies with SYN and ACKnowledgment (synchronize acknowledge). Finally the third message

[1] The name of the attack comes from SYN control datagram used by TCP to establish connection between hosts.
[2] Datagram is a basic unit of data transfer in IP.

from source to target has ACK (of the SYN ACK) datagram and indicates that both hosts agree on the connection being established. While waiting for the ACK to the SYN ACK, a connection queue of finite size on the destination host keeps track of connections waiting to be completed. The TCP SYN attack exploits this design by having source host generate SYN packets with phony IP addresses toward a victim host. The victim host sends SYN ACK to the phony source and allocates resources for data structures in its connection queue. Since the connection cannot be completed, the entry in the connection queue remains until a timer expires. If after a specified timeout (usually 75 seconds), the target host does not receive ACK, the half-connection is reset and the allocated resources for it are released. By generating fake TCP SYN datagrams at rapid rate, it is possible to exhaust limited pool of host resources dedicated for connection queue. Once the victim host resources are exhausted, no more incoming TCP connections can be established, thus denying TCP services (email, file transfer, or the Web) to legitimate users [Cisc96].

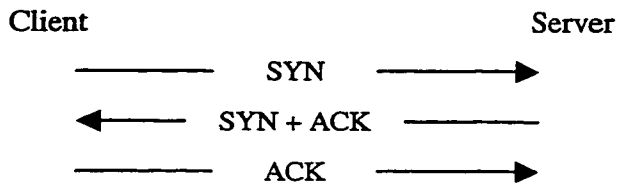

Figure 7.1 Three-way handshaking sequence

TCP implementations are designed with a small limit on how many half open connections are possible per port at any given time. The half-open connection state is called SYN RECVD state, i.e., SYN received. When the maximum number of half open connections per port is reached, TCP discards all new incoming connection requests until it has either cleared or completed some of the half open connections. Overall system resources are usually sufficient for several ports to be flooded. It is worth noting that neither outgoing connection attempts nor connections that are already established are affected by this attack.

Note that to be successful, the attacker has to provide source addresses of the hosts that are really slow, virtually or physically disconnected, down or non-existent. Otherwise the source host upon receiving SYN+ACK, without requesting connection, will send ReSeT to target host. This will consequently reset the connection and all resources allocated for the connection will be freed. Furthermore, if the attacker wants the denial of service condition to last longer than the timeout period, new connections have to be requested continuously from the victim machine.

This denial-of-service attack is very easy to mount and presently, it is difficult to trace an attack back to its originator. The attack exploits the weakness of TCP/IP protocol suite and is difficult to defend against, without correcting these protocols. Several solutions to this attack have been proposed. A group at Purdue University [SKKS96] has developed an active monitoring tool that classifies IP source addresses with high probability as being falsified or genuine. Their approach finds connection establishment protocol messages that are coming from forged IP addresses and ensures that the resulting illegitimate half open connections are reset immediately.

7.1.2. UDP Diagnostics

This attack is not as popular as SYN flooding, maybe because it is easier to deal with. Here, the attacker transmits a volume of requests for User Datagram Protocol (UDP) diagnostic services to the router, which causes all CPU resources to be consumed servicing the phony requests. By default, most of the routers have enabled series of diagnostic ports enabled for certain UDP and TCP services including *echo*, *chargen* and *discard* [Cisc1]. When a host attaches to those ports, a small amount of CPU is consumed to service these requests. However, if a single attacking device sends a large barrage of requests with different, random phony source IP addresses, it is possible that the router can become overwhelmed and slow down or fail [ChBe94:25]. The router under attack will have process table full and exec command *show process* will show a lot of processes with the same name, for example, "UDP Echo".

To defend against this attack, every router that has UDP and/or TCP services must be protected by firewall or have these services disabled.

7.2. Service Theft

There are others, sometimes less perceptible methods to decrease system availability. Using the system for games, personal e-mail and personal business can be fraud when computer time is charged to customer projects. They also divert system resources from being used for more appropriate and legitimate tasks. Hackers steal computing and communication services from system they penetrate and users can sell computer services to outsiders. Since theft of computation resources, diverts computing resources on the shared system, it presents some denial of service.

7.3. Defenses

Traditionally, two defenses were used against denial of service attacks: access control list (see chapter 3) and time-sharing. Time-sharing algorithms are used to protect monopolization of shared system resources by run-away maliciously or inadvertently coded program. This works quite nicely when all the requests come from the clients under the control of the operating system (OS), since OS can block any unreasonable requests or even terminate misbehaving programs. The story is different in widely distributed environment where there is no single trusted entity that can control the agents making requests. Individual servers might ignore specific client requests that seem unreasonable or that would degrade/deny service to others, but servers cannot slow or terminate the clients making those requests. Moreover, the testing whether the clients make reasonable requests or not consumes resources and even if servers successfully detect denial of service attacks, these attacks can still succeed.

Another problem with time-sharing solution for distributed systems comes from the statistical approach to sharing fixed-capacity resources. Accordingly, server capacity is chosen and scheduling algorithms are used to allocate service among contending clients. Since scheduling algorithms are based on workload assumptions, the attacker can subvert the scheduling algorithm by violating those assumptions and altering the character of the offered workload. Say, by increasing the number of clients making apparently reasonable requests, an attacker might succeed in degradation of system availability. This attack has more chance to succeed on the Internet, because the Web and foreign code provide a vehicle for the malicious code to be downloaded onto the hosts.

Unfortunately there are no technical solutions to inadvertent denial of service. The solution lies with establishing policies, procedures, standards and managerial and societal controls.

7.3.1. Resource Requirements

There are a number of solutions for SYN flood attacks (see 7.1.1. for the attack description). One of the most obvious proposals involves a significant increase of the number of resources for half-open connections, reducing the timeouts and disabling non-essential services. The increase of the backlog queue makes the system able to cope with more simultaneous half open connections than before. The smaller timeout helps in pruning the number of half-open connections from the queue. Disabling non-essential services limits the number of ports that can be attacked. The

disadvantage of smaller timeout is that now, legitimate but slower hosts will be denied the access. Then the size of the queue must be chosen in such a way that the port cannot be flooded within specified timeout, with the SYN requests. M. Graff in [Graf96] describes the formula that calculates required memory based on queue length and abort time. You need to remember that this configuration will be effective up to the point when the rate of arriving packets will be bigger than the length of the queue divided by timeout. At this point, all packets will be dropped. So, will your system benefit from additional physical memory? It depends on what amount of memory your system currently has, what version of operation system you are running and whether you can configure the TCP services to use shorter abort time. You should contact the vendor of your system to verify whether and how to upgrade your system.

7.3.2. Router Configuration

Another solution involves configuring routers to make it difficult for packets with spoofed source addresses to traverse. The external routers interfaces are configured to block packets that have source addresses from the internal network. The internal router interfaces block packets to the outside that have source addresses from outside the internal network. This limits the ability to launch a SYN flooding attack from your network, because the attacker would only be able to generate packets with internal addresses. Event though this will not prevent denial of service attack on your network, it rules out your location as the source of the attack. These measures can be effective, but only if taken on a large scale. As more Internet Service Providers (ISPs) configure their routers appropriately, the fertile ground for launching SYN flooding attacks may be reduced.

Some improvements in router configuration can be done in the environments where address spaces reachable from multiple router interfaces are well defined and disjoined. This seems to be the case for routers connecting an organization or a local ISP to a backbone network. The address prefixes separate the inside from the outside. However, there are practical obstacles to make this approach work. In general, routers in a complex backbone topology cannot make clear distinction between inbound and outbound traffic. Packets are routed based on current link availability and load, and can take numerous paths through the network. Legitimate packets from the same source can reach backbone router over different interfaces. As long as a substantial number of sites transmit packets to the backbone networks without checking of source IP address, hosts will be subject to untraceable attacks. Hence, additional backbone mechanisms need to be put in place to

cope with the network based attacks. One of such mechanisms would be encryption of the source addresses to allow tracing the physical transmission path of any IP packet to its source. This technically would not prevent SYN flooding or other network based attack, but the ability to trace the attack to its source would deter at least casual attackers. Tracing is extremely useful in fighting SYN flooding, but unfortunately it is not currently part of the Internet infrastructure.

7.3.3. Protocol Modifications

Another solution deals with asymmetric resource allocation for two hosts needed to establish three-way handshake protocol. The destination host needs to allocate large data structures in response to any SYN packet, without any guarantee of its authenticity. In particular, the destination host stores ISS of its second message. An alternative for storing this ISS would be regeneration of it upon receiving the third message. One such mechanism is to calculate destination's ISS as a cryptographic hash value of source and destination IP addresses, ports, the source's ISS and a destination specific secret key. The destination D, would calculate ISS in that manner and use it in its SYN+ACK message. At the time D received the third message of the three-way handshake, it can recalculate ISS by using its secret key, sequence number, the addresses and the ports found in that message. If the number matches the ISS of the third message, the connection is legitimate otherwise it is not. Note that this solution also provides some protection against sequence number prediction because of the statistical properties of good hash functions. Even though it hardens TCP based services against SYN flood attack, the serious drawback of this solution is that it would require modification to TCP protocol and all its implementations. Moreover, this method makes it impossible for the source to append any data to its third message, because its ISS is an input for the hash function. Since there are 2^{32} sequence numbers, there is a small probability that old or forged packet can open the connection.

7.3.4. Firewall Approach

Since many hosts connected to the Internet are already protected by firewall, it makes sense to use the firewall to defend it against SYN flooding attack as well. The idea behind this approach is that all packets destined to the hosts inside firewall have to be examined by firewall and hence, the decision can be made on the packet authenticity. Having this information, the firewall can take the action to protect the internal hosts. The drawbacks of this approach are delays on every

packet for additional processing. Not every firewall product is capable of adding functionality, such as a module to protect against SYN flooding.

There are two popular configurations in which firewall is employed to defend against SYN flooding: firewall as relay and firewall as semi-transparent gateway.
In the first approach, firewall establishes connection on behalf of the internal host. Only after successful completion of three-way handshake protocol, internal host is contacted by firewall. You need to note that this approach only works when the firewall itself is not vulnerable to SYN flooding. In case of the attack (see Figure 7.2), the firewall will terminate half-open connections after specified timeout. In case of legitimate SYN request (see Figure 7.3), the firewall opens a new connection to the internal host on behalf of the original client. Once this connection is established, the firewall has to play the role of a proxy; to translate sequence numbers for the packets traveling between source and destination hosts. The drawback of this method is that it introduces communication delay for legitimate users, first to open the connection and later to exchange data. The obvious advantage is that destination host never receives phony connection requests.

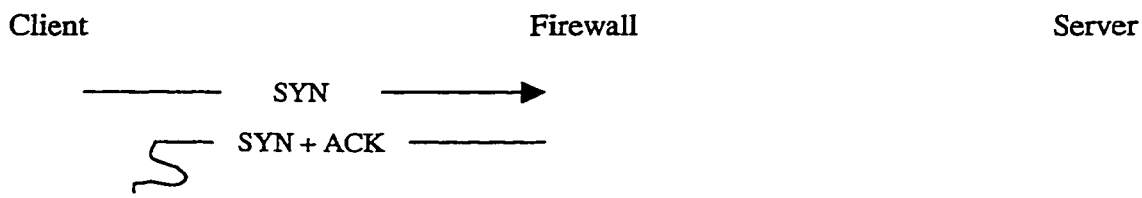

Figure 7.2 Host protected by firewall as a relay under attack

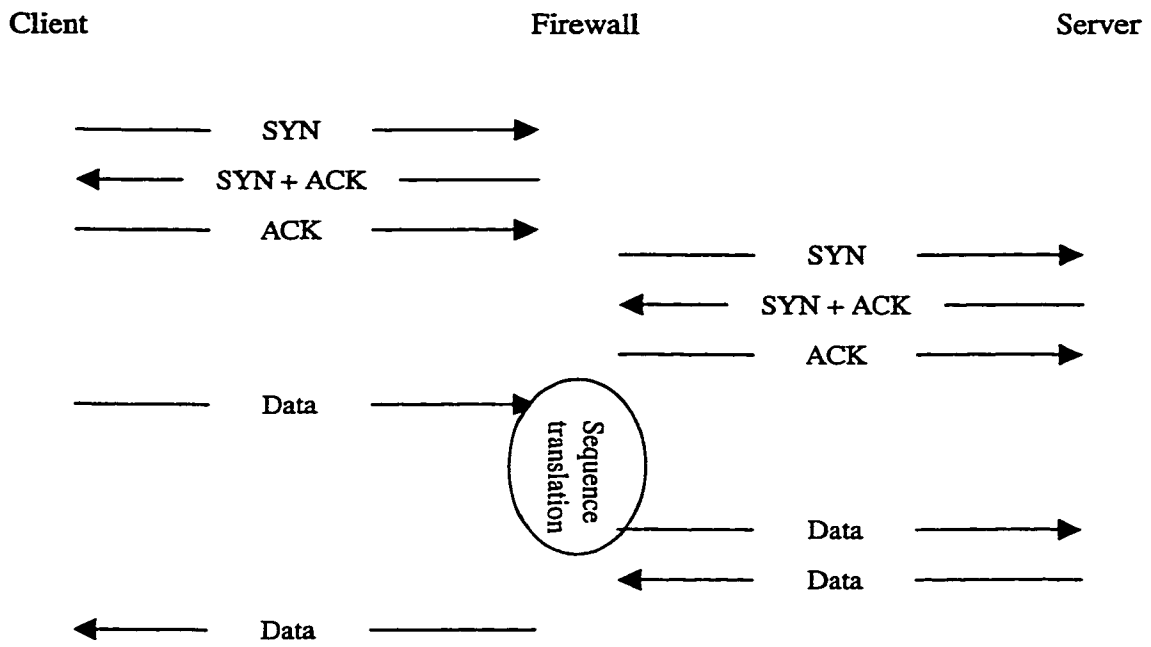

Figure 7.3 Host protected by firewall as a relay during legitimate connection

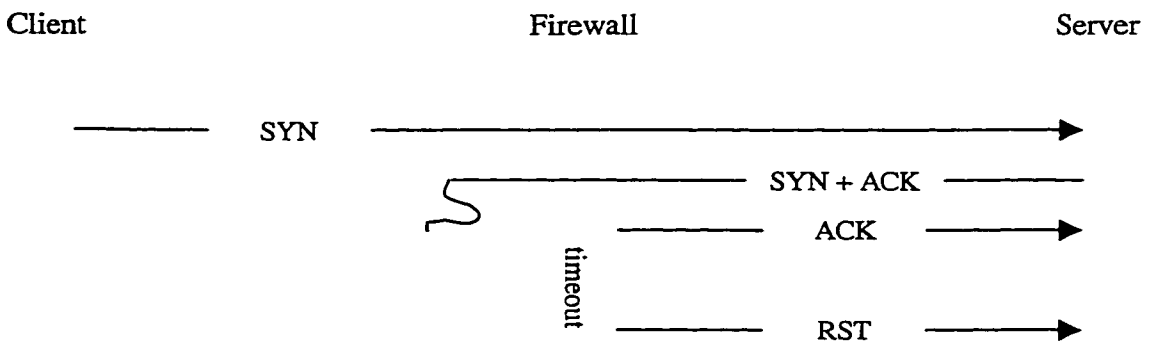

Figure 7.4 Host protected by firewall as a semi-transparent gateway

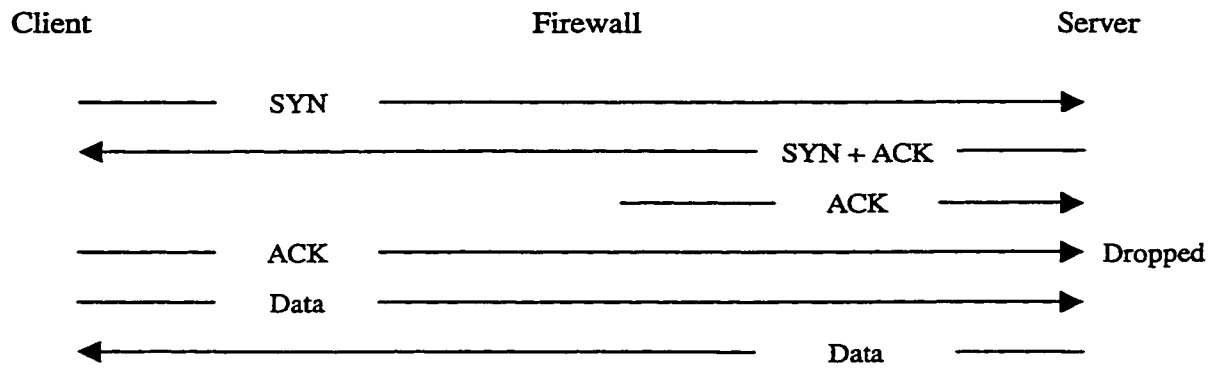

Figure 7.5 Host protected by firewall as relay during legitimate connection

When firewall is used as semi-transparent gateway, SYN and ACK datagrams are allowed to cross firewall but the communication between the source and destination is monitored. The firewall passes the original SYN datagram to the internal host. After receiving SYN+ACK from the internal host, the firewall allows it to travel to the source but generates ACK that seems to come from the source and sends it to the internal host. This allows the internal host to move the entry from the backlog queue to open connections queue. In case of an attack, the firewall will pass SYN+ACK from the internal host but if it does not receive ACK from the source after a timeout, it will send a RST packet to terminate the connection (see Figure 7.4). In case of legitimate connection (see Figure 7.5), the ACK datagram from the source will be allowed to cross the firewall and reach the destination. You notice that in this scenario, the destination will receive two ACK datagrams but this does not cause any trouble since TCP is designed to handle duplicate packets. The advantage of this approach is that there is no delay for the legitimate connections once they are established. However, when the host is under attack, there are many phony open connections. However, this will unlikely cause trouble because the number of entries in open connections queue is much higher than the limit of half-open connections, since it is limited by CPU and memory resources. As before, the timeout value has to be selected appropriately, so the large number of open connections will not overload the host and at the same time will not deny service to asynchronous connections, or slow hosts.

A very attractive solution involves active monitoring of the TCP network by software agents, because the agent can watch for certain condition to arise and react appropriately. This method is a general approach that can be used to detect and react to a large class of network-based attempts to breach security. This advantage of this approach is low cost and flexibility, since it does not

require modifying existing software (like TCP services) neither buying a new hardware (like firewalls or routers with filters). Such tools exist and are described in [Denp91] and [FaSp94]. According to user-definable parameters, the program monitors the local network for SYN packets that are not acknowledged after a certain period of time, and frees the allocated resources by sending matching RST packets.

7.4. Remarks

We need to rethink computer security to include denial of service. Denial of service is the least researched security property, even though it is the most popular and successful attack launched from the Internet. Denial of service can range from abusing or stealing computer resources, to loss of messages through deliberate overloading, to total loss of service. Defending against denial-of-service attacks is very difficult and there exists no systematic method of such defense, even though they are crucial in assuring availability.

Denial of service attacks are difficult to avoid. Networks can be flooded with traffic and packets can be routed to the already overwhelmed routers by contaminating the Internet Domain Name Service (DNS) caches. Popular denial of service attacks, include: TCP flooding and overwhelming UDP services. TCP SYN attacks consist of a large number of spoofed TCP connection set up messages aimed at a particular service on a host. Older TCP implementations cannot handle many faked SYN packets and will not allow access to the victim service. Several solutions to this attack have been proposed. One of them involves an active monitoring tool that classifies IP source addresses with high probability as being falsified or genuine.

Note that the SYN flood attack is not the result of any system bug, defect, flaw, or any error in implementing the TCP protocol. Rather, it represents the stark difference between the world in which TCP was designed and the Internet environment we have today. As mentioned before, when networks were first developed, the hosts were assumed to be trustworthy.

The most common form of UDP flooding directed at harming networks is an attack consisting of a large number of spoofed UDP packets aimed at diagnostic ports on network devices. This attack can cause network devices to use up a large amount of CPU time responding to these packets.

You should be mostly concerned about attacks on the services provided for large groups of users such as servers and switching equipment, since these have the widest impact. This is also the reason why these resources become an attractive target.

Defenses against SYN floods include: adding resources, protocol modifications, employing firewall and active monitoring.

One of the most obvious defenses against SYN floods, involves a significant increase of RAM for half-open connections, reducing the timeouts and disabling non-essential services. The increase of the queue size makes the system able to cope with more simultaneous half-open connections than before. Disabling non-essential services limits the number of ports that can be attacked.

Another solution deals with asymmetric resource allocation for two hosts needed to establish three-way handshake protocol. In particular, the destination host stores ISS of its second message. Without storing this number, any host could establish a connection by sending only the third message. An alternative for storing ISS would be regeneration of it upon receiving the third message. Of course, this implies modifications to TCP protocol and all its implementations.

Yet another defense against SYN floods uses firewalls. Two firewall configurations are popular: firewall as relay and firewall as semi-transparent gateway. The disadvantage of the first configuration is that it slows down the communication performance between two hosts and the firewall itself must not be vulnerable to SYN flooding. The disadvantage of the second configuration is that there are many phony open connections on the destination host during the attack.

A very attractive solution involves active monitoring of the TCP network by software agents. This method is a general approach that can be used to detect and react to a large class of network-based attempts to breach security.

The impromptu countermeasures have been successful in defending against denial-of-service attacks, but these countermeasures will become unsuitable for distributed systems. For example, some router vendors have added or are currently adding options to turn off the ability to spoof IP source addresses by checking the source address of a packet against the routing table to ensure the return path of the packet is through the interface it was received on.

Chapter 8

Foreign/Mobile Code

"There is a price that must be paid for the power of executable content. This price is very similar to the price that must be paid in order to connect to the Internet in the first place." [McFe97:24]

New developments in mobile code are fuelled by the growing expectations in the Web browser market. Often, Web pages contain an executable code that is downloaded and executed by a user's computer to enhance Web pages with animation and other locally generated special effects. This foreign/mobile code poses the biggest security challenges.

The competition, between Microsoft and Netscape in the Web browsers market, lead to improvised extensions to Hypertext Markup Language (HTML). These extensions are referred as Dynamic HTML (DHTML). There is no single specification for DHTML, since it is really a combination of several things. The HTML 4.0 of World Wide Web Consortium (W3C) includes much of dynamic HTML. Although JavaScript, Java applets and ActiveX controls were present previously in the Web pages, DHTML entails an increased amount of programming in Web pages, because more elements of the page can be programmed. Unfortunately, the user surfing (or crawling) the Web is often unaware about the execution of the mobile code. The Java applets and ActiveX code is executed automatically, since the default settings for Netscape and Internet Explorer allow for execution of such programs.

While the World Wide Web (WWW) began its life as a platform for sharing documents over the Internet, today the Web is being used for much more than simple document publishing. In fact, most commercial and corporate Internet sites can be more accurately described as Web applications, because they require complex processing to create a more compelling, informative experience for users. Web technology is also experiencing widespread growth as an effective platform for deploying corporate intranet applications. Unfortunately, the increased use of foreign code may enable enhanced functionality, but it will also erode system trustworthiness unless security mechanisms are developed and deployed for confining the effects of foreign code.

8.1. Software Distribution

Software products can be either purchased directly from commercial vendors such as Microsoft, IBM, America Online (AOL) and others, through software distributors, or can be downloaded from the Web as shareware/freeware. The first two distribution channels deliver shrink-wrapped software and the producer's reputation is at stake if there are problems with the software [Appe99]. These physical distribution channels unintentionally create obstructions to the distribution of malicious code. There is no such barrier for products distributed through the Web.

Note that no matter what channel you get your software from, you should avoid a bootleg software. If you buy your software from a reputable source and observe the terms of the software license agreement, you stand to gain a lot. Such software is very unlikely to be infected with virus (or any other form of malicious code) and it comes with warranty and support rights.

8.2. World Wide Web

The World Wide Web (WWW) is a system for exchanging information over the Internet. It is one of the most exciting uses of the Internet, but it also poses profound security challenges. In the early days of the Web, when you clicked on a hyperlink, your browser simply printed its contents to your screen. There is no danger there. But today, your browser may invisibly download and locally run a whole bunch of little programs. These "active" Web page elements, including ActiveX controls, Java applets and good old-fashioned *.EXE files, pervade the Web and are the wave of the future. Their presence as local programs is automatic and not apparent, which is the way it should be. How do you know these active elements do not attempt to annihilate your system? They run under your identity with your capabilities and, invisibly, can do everything you are able to do: delete files, or, perhaps worse, mail them off somewhere. They can even install a malicious program on your computer: a virus[1].

The Web browsers have configurable security settings such as, whether allow to execute Java applets or JavaScript, to store cookies and to download trusted ActiveX controls, but most of the users will never either know or want to change the default settings. Even when they do change these settings, often they find sites that would not work without installing cookies.

[1] One of the highest public profile attacks on information security. Today however, there is no reliable figure for likelihood of virus infection.

8.3. Common Gateway Interface

Common Gateway Interface (CGI) is a meta-language for translating Uniform Resource Locators (URLs) or HTML forms, into runnable programs. Any language that meets a few CGI requirements can be used to write these programs, but scripting languages like Perl, Tcl, or Safe-Tcl are particularly suited for this purpose [ABHK96:732]. Execution of CGI scripts does not change the security problem itself, but enlarges its scale. Now, more authors can run more programs on the server. CGI scripting allows creating more sophisticated Web sites. To create more interactive Web sites, you need the ability to write programs on the server that can dynamically generate Web pages, based on information in the HTTP request and from information stored on the server.

CGI works as follows: the client sends a URL or HTML form to the server, specifying a CGI script as an input argument. This request is transferred to a program that is executed with user identity of the Web server program. The Web server may invoke applications called Server-Side Includes (SSIs). When the client's document contains system commands, called SSI in-lines, these commands are evaluated and the result is inserted in the document returned to the client.

Generating Web pages dynamically through the use of server-side processing represents a fundamental shift in the architecture of the Web; it moves the Web from being essentially a document distribution technology, towards an application development technology.

There are numerous security tasks that need to be performed to manage security at the Web server, to prevent malicious CGI script from causing damage. Whether the Web server is Unix or Windows NT based, the server has to manage controlled invocation to its clients. First, the system manager should keep track of CGI scripts [GaSp96:547]. There are two options: place all scripts in one directory, or identify CGI scripts by their extension, e.g., .cgi. The first option is more "manageable" since it is easier to find all CGI scripts. For Unix hosts, user ID (UID) for the Web server program has to be decided. The best option is to create special UID for the Web server and carefully control its access rights. The downside of this solution is that all CGI scripts will run under the same UID. To make CGI scripts run under author's permissions, wrapper software such as CGIWrap is needed [ABHK96:741].

While server scripting on Windows NT hosts can be accomplished using CGI programs, the server may utilize Microsoft Internet Information Server (IIS) with Active Server Pages to provide server-scripting capabilities.

A code review of CGI scripts is also a good idea, if you have time and expertise to do it. Finally, the input to CGI scripts must be filtered. The format of SSI in-line is:
<!-#operator arg1="string1" arg2="string2" ...->

The maximum flexibility is provided by the operator exec with argument cmd. This way, the user can specify the program and its parameters to be executed. The danger can come either from the program or from the parameters, if these parameters contain shell escape. The *unescape* will remove shell escapes, some scripting languages like Perl, also provide this feature. This still leaves attackers with all the Unix commands and their inputs to play with. If the risks are too high, disable *exec* through configurations of the server options.

8.4. Cookies

Sometimes Web pages need to save information that has been obtained about the client. Saving this type of information for later use can help preserve user preferences or other information to provide a better experience for users who return to your site. You could store the client information (user profile) on the server, but storage requirements and search times would render it ineffectual for large number of users. Furthermore, HTTP requests do not automatically identify individual users, so it is easier to use the client's side to store information. When the client calls the next time, this information is provided to the server by the Web browser.

Another reason for storing this information is technical - HTTP is a stateless protocol, which means that all HTTP requests are treated independently, even when they come from the same client. Hence, if you need to specify a password to access the site, you would have to enter it repeatedly whenever you click on the same page. This problem was solved in HTTP 1.0 for the duration of a session. The browser would save the password entered at the first request and use it for subsequent requests from the server. To get around these problems, Netscape[1] developed the concept of a *cookie*. A cookie is a small data that is stored on the user's hard disk. In Web

[1] Netscape is part of America Online (AOL) since March 1999.

parlance, cookies are a means of saving information such as browser settings, recently visited Web sites and class paths, on the client machine.

Cookies let browsers create stateful HTTP sessions, reducing their administrative overheads as well as overheads of the user. They represent the first step in shifting workload and storage requirements from the server to the client. Do they represent a security hole?

There is some controversy about cookies. Some users feel that cookies are a significant security hole, providing nefarious programs access to information on the client machine. This perception is partly due to the opacity of cookies: users do not know when a site is storing a cookie on their machine, what information is contained in the cookie; or (most importantly) how the information in the cookie is going to be used. Cookies cannot violate the integrity of your system. They are data, not an executable code after all. They do not disclose the information to the server, because the server asks the browser to store the cookie in the first place. Hence, an individual cookie does not represent a confidentiality problem, either. It leaves a privacy issue. The privacy is not an issue when an individual cookie is used but might be invaded if the whole set of cookies is accessed, thus providing the server with the user profile. Hence, the access control of the browser is crucial. Responding to the concern of privacy invasion, most browsers enable users to disable cookies or to receive a prompt before accepting cookies, but this quickly becomes a nuisance. If you are concerned with invasion of privacy, you should delete cookies at the end of the session.

8.5. ActiveX

Microsoft Internet Explorer (IE) browser protects end users from malicious applications by ensuring that only safe applications run on user systems. ActiveX controls that are hosted on Web sites trusted by the user can be downloaded and run on the user's system using all the features of the operating system. Users accept such trusted ActiveX controls just as they trust shrink-wrapped applications today for the desktop. Because ActiveX will not run the trusted control if it is modified after leaving the trusted Web site, you are protected against malicious modification of ActiveX applications during download. Or are you? In this section we will point out that trust level of shrink-wrap applications and certified "active" components is not the same.

Authenticode is a code certification scheme, in which the reputable software vendors join a software maker organization and receive a certificate from the Certificate Authority (CA). People

who browse the Web install that certificate on their workstation. Vendors sign their active elements. Now, your browser can determine two important things: that an active element comes from a genuine member of that organization, and that it has not been tampered with from the time the vendor signed it. Microsoft Internet Explorer can check Authenticode on any active element (ActiveX, Java, and so forth). Is this sufficient assurance? From technological point of view, Authenticode assurance is stronger than assurance of shrink-wrapped software. It is infinitely harder to tamper with cryptographic signatures than shrink-wrap. However, ActiveX Authenticode approach presumes that users can decide whether to run a module based on knowing the identity or seeing some credentials of a vendor or distributor. This presumption has questionable validity, as the successful deployment in February 1997 of a malicious ActiveX module by the Chaos Computer Club of Hamburg confirmed [Leit97]. Users either do not bother to look at a signature, or cannot make an informed decision upon seeing a signature. The intended analogy between signatures and shrink-wrap packaging is likely flawed, not because of technological holes in Authenticode but because of the way the technology is (or might be) used.

Another difficulty with Authenticode is revocation of compromised keys. Since, there is no global infrastructure for managing signatures for code, it is conceivable that the compromised keys could be used for signing malicious code. Even when such keys were discovered, the revocation process across the Internet would be lengthened because most of the Internet users are not security aware. The solution would be to maintain the list of revoked Authenticode keys by CAs and configure the Web browsers, or other applications, to verify this list before executing ActiveX module. Actually, VeriSign has been keeping a list of compromised keys for ActiveX signatories and Microsoft has been verifying this list on their mobile code platforms. However, ActiveX users and administrators have not followed this path, so far. Furthermore, with growing number of ActiveX modules or any other signed executable modules for that matter, probability of inadvertent signing of malicious code increases. It is mostly so because of personnel, procedural and physical security holes in CAs and not because of code signatures weakness.

The difficulty of assessing author's identity is not solely ActiveX Authenticode problem; it exists today in all uses of signatures on the Internet. We believe that establishing PKI infrastructure, education of the Internet users and administrators, would go a long way in building trusted Web applications containing signed executable code. Today however, technological solutions only are not enough to guarantee the security of certified active content on the Web. Thus, another security scheme for delivering foreign code based on constraining the environment in which this

code is executed, seems to be a superior solution, since the security can be achieved without relying on PKI infrastructure and education of users.

8.6. Java

Java takes a different approach to security than ActiveX Authenticode. It enforces the security by confining code execution to the environment known as the Java virtual machine (JVM). The first versions of the system had very limited flexibility in expressing the security, with either very tight restrictions or almost none, depending on whether the code came from trusted source or not. Current versions allow for extensible and expressive access control to system resources, through access control lists (ACL).

Java byte code is a stack-based intermediate language, designed to be platform independent and to be interpreted[1] by JVM [ABHK96:696]. Java programs, in byte code format, contain the configuration of the run-time stack throughout execution, type information about their variables and the signatures of routines that are defined and invoked. Java is designed to be secure [GaSp96:562]. When Java code is loaded, JVM performs an initial check to verify that the program conforms to type-safety rules. These type-safety rules and other security checks are done by JVM during the execution of the Java program.

Evidently, Java applets do not necessarily gain full access to your system. Your browser (more properly, JVM that runs the Java applets) can limit these applets, for example, preventing them from writing onto your file systems. The scope of a Java applet is often called its "sandbox", or like the proponents of ActiveX Authenticode prefers to call it, prison cell. With the proper Java security model, this can be an effective security restraint. However, it inevitably trades off against the capabilities of the applet. Although, this trade-off still exists today, the security model is much richer than when first deployed and certainly, it is possible to build Java applets that match ActiveX functionality.

Since Java programs are interpreted by JVM, they are slower than programs compiled to the machine code native to the platform. Mostly for performance reasons, some Java compilers generate the platform's native code. This certainly makes such programs run faster but at the

[1] Actually, most JVMs do not interpret the Java byte code directly. Instead, they just-in-time-compile the byte code to native machine code and execute the native code.

same time makes them platform dependent and weakens system security, because Java security model was not designed to cope with non-Java (non-interpreted) programs.

The early Java all-or-nothing access control approach was not very useful since it did not allow building systems consistent with principle of least privilege. Current version – JDK 1.3, implements richer but also more complex security model [Melo99]. Now, the onus is on the programmers to correctly assess and configure suitable access rights for executing foreign code.

8.7. Java versus ActiveX

Note that the two technologies: Java and ActiveX are fundamentally different and we only compare ActiveX Authenticode versus Java approach to security on the Web. Both technologies are used to attach computer programs to Web pages in order to make these pages more dynamic and interactive. Java was developed by JavaSoft, which is a division of Sun Microsystems and is supported by both major browsers, Netscape Navigator and Microsoft Internet Explorer. ActiveX was developed by Microsoft and is supported by Internet Explorer. However, there exists an ActiveX plug-in for Navigator, which enables Navigator to run ActiveX code.

Both technologies introduce some security risk, because they can automatically download potentially hostile programs and run on your computer, just because you visited some Web page. The downloaded code can damage data on your hard disk, install virus or a trojan horse. Java and ActiveX take measures to protect you from this risk. In this section we will evaluate the approaches of Java and ActiveX, protecting the user from malicious mobile code.

ActiveX security relies on you to make correct decisions about which programs to accept. ActiveX programs come with digital signatures from the author of the program and anybody else who endorses it. Your browser can look at a digital signature and verify whether it is genuine, so you know who signed the program. Now, you can either accept or reject the program. The main danger in ActiveX is that you will make the wrong decision about accepting a program. Even though the risk of accepting one program is low, the risks add up when you repeatedly accept programs. And it takes only one bad program to cause extensive damage. The only way to avoid this scenario is to refuse all programs, no matter how interesting they sound, except programs that come from a few authors you know well.

Since ActiveX security relies entirely on user's judgment, it is suitable for self-disciplined and educated users who can make appropriate decision based on the authorship of the mobile code.

Java security relies entirely on software technology. The downloaded Java programs are run in security "sandbox"[McFe97:37]. The sandbox is like a fence that keeps the program away from your private data. As long as there are no holes in the fence, you are safe. The main danger in Java comes from the complexity of the software that implements the sandbox. Java is rather complicated and several breakdowns have happened in the past. Most of the users do not have the time or the desire to examine Java and look for implementation errors. However, prolonged analysis, have found no serious weakness in Java security model and the technology is generally trusted as it withstood the test of time.

8.7.1. Signed Java Applets

The original version of Java sandbox can be too restrictive [Melo99]. For example, Java programs are not allowed to access files, so there is no way to write, for example, a text editor. To work around this program, Java applets are now starting to use digital signatures. The idea is the same as ActiveX. The downside of this scheme is that it introduces some of the ActiveX problems. If you make a wrong decision about who to trust, you could be in deep trouble [McFe97:149]. There is no known way to avoid this dilemma. Some programs must be given power in order to be useful and there is no guarantee that those programs will well-behave.

Java signed applets still offer some advantages over ActiveX. You can put only partial trust in a program, while ActiveX requires either full trust or not trust at all. Java-enabled browser could also keep a record of which dangerous operations are carried out by trusted program, so it would be easier to reconstruct what happened if anything went wrong. Finally, Java offers better protection against accidental damage caused by buggy programs.

8.7.2. Plug-ins

Plug-ins make possible to add code to your browser. They have the same security model as ActiveX, that is, when you download a plug-in; you trust them completely. Hence all the warnings about ActiveX apply to plug-ins too.

Many plug-ins such as Sun's Safe-Tcl and Macromedia's Shockwave are complete general programming systems, just like Java. By accepting a plug-in like this, you are trusting that the plug-in program has no security-relevant bugs. As we have seen with Java, systems that are meant to be secure often have bugs that lead to security problems. If you are feeling paranoid, the only plug-ins you should have are those with less than general-purpose functionality. A plug-in, which handles a new image, video or audio format is less likely to be exploitable than a plug-in for a completely general animation system.

8.7.3. Final Comments

Most debate about security has been between JavaSoft and Microsoft. Each company has accused the other of being careless about security and some misleading charges have been made. For most people however, Java security model will be more appropriate than that of ActiveX. If you are informed about the risks, you can make a rational decision to accept some danger in exchange for the benefits of using Java and ActiveX.

To lower the risk, there are several things you could do:

1. Never surf the Web on a computer that contains highly sensitive information.
2. Think very carefully before accepting a digitally signed program.
3. Use up-to-date browser versions and install the security patches offered by the browser vendor.
4. Close your browser and remove cookies after terminating each Web session.

8.8. Remarks

The most extreme form of untrusted code is foreign or mobile code that arrives from imperfectly authenticated sources. Foreign code represents a new threat to the applications and their data running on the network nodes such as workstations, PCs, HPC, wireless phones and so on. The fundamental problem is running mobile code on our systems without exposing ourselves to losing the whole system. Code might be malicious. We need to enforce the principle of least privilege, giving the mobile code the minimum set of capabilities to do its legitimate work. Approaches proposed are: sandboxing (JVM), code signing (Authenticode), proof-carrying code (PCC) and software fault isolation (SFI). JVM works quite well, but it might be too restrictive for some programs. PCC and SFI were tested only on small samples, but results are quite promising.

Today, Java is by far the most popular implementation of the Web-based executable content concept. Lesser known competitors include JavaScript, SafeTcl, Telescript, Word macros, Excel macros, ActiveX and Postscripts. Any scripting language than can be embedded in the document, transferred around the Net, and run on different machines falls under the classification of executable content.

While the World Wide Web began its life as a platform for sharing documents over the Internet, today most commercial and corporate Internet sites resemble Web applications, because they require complex processing.

Software products can be either purchased directly from commercial vendors, through software distributors, or can be downloaded from the Web as shareware/freeware. While the physical distribution channels unintentionally create obstructions to the distribution of malicious code, there is no such "barrier" for products distributed through the Web.

ActiveX security relies on you to make correct decisions about which programs to accept. ActiveX programs come with digital signatures from the author of the program. Your browser can look at a digital signature and verify whether it is genuine, so you know who signed the program. Experience shows that authenticating the author or provider of foreign code has not been effective for enforcing security. Users are unwilling and /or unable to use a signature as a basis for denying or allowing execution. Revocation of certificates is necessary should a provider be compromised, but is not currently supported by the Internet, which limits the deployment scale of this approach.

Java security relies entirely on software technology. The downloaded Java programs are run in security "sandbox". The main danger in Java comes from the complexity of the software that implements the sandbox. However, confining foreign code according to an interpreter that provides a rich access control model has potential, assuming programmers and users have means to correctly assess and configure suitable sets of access rights.

The original version of Java sandbox can be too restrictive. Some programs must be given power in order to be useful and there is no guarantee that those programs will well-behave. Hence, the idea of signed Java applets has potentially more capabilities.

To lower risk from mobile code you should:

1. Never surf from computer containing sensitive information.
2. Be careful about accepting signed code.
3. Install browser patches and remove cookies.

Java is a futuristic tool for mobile computing. Having programs embedded in Web pages that can run on any platform is an excellent idea, but users take some risk and they have to trust JVM for providing the secure execution environment.

Chapter 9

Conclusions

Increase of computing power combined with the dramatic decrease of the cost, make computing devices more pervasive in every aspect of everyday life. These computing devices range from personal computers (PCs), auto PCs, so called wearable computers, computing appliances to network servers and mainframes. The computing cost is driven down by the competition and sales volume, and these in turn are fed by consumers' appetite for fast services, easy-to-use devices, mobility requirements, orderly access to information, entertainment and simply electronic gadgets. Very often, the computing devices are interconnected via heterogeneous networks. The networks components such as routers, bridges, fiber optic links, satellites, antennas and all the software components need to keep pace with computing capabilities of connected computers[1]. This growth of interconnectivity and the (cheap) computing power, mounts a serious challenge to information security.

Just as there has been an unstoppable increase in computing power, the world of distributed (wireline and wireless) computing has similarly advanced at an astonishing pace. This rate of distributed computing growth is causing revolutionary changes in network technology and is forcing changes in the information security technologies.

A computer device can be connected to the Internet via radio, or infrared LAN, wireless telephone, or direct connection through Ethernet or token ring network. As network services proliferate and become available universally, every network device will take advantage of mobile networking technology to offer maximum flexibility to users. Mobile IP provides many solutions to network mobility, but not all of them [Perk98].

[1] Such networks are referred to as either computer systems, systems or information systems.

9.1. E-business

The only way to truly secure the computer system is to isolate it from non-secured networks. But hardly any organization can afford this approach and the Internet connection is becoming a commercial requirement. As the number of and demand for the Internet services grew, proxy servers, application gateways and firewalls have become the standard means of protecting an organization from the Internet. Now, organizations are looking beyond passively protecting themselves from the Internet; they want to use the Internet for their competitive advantage. This has led firewall vendors to focus on authentication and encryption technology and the emergence of Virtual Private Networks to secure transactions being routed across the Internet. However, for e-business to be successful, there are still significant issues that need to be addressed, such as the speed and reliability of traffic movements across the Internet, and key management protocols.

Digital signatures provide the best way to ensure trusted authentication, integrity, confidentiality, non-repudiation and time-date stamping of an electronically filled documents because all of these services can be automated and provided with exact full credibility. These security features are particularly important for electronic payments. To make it work however, a trusted third party need to bind the digital signature to the person claiming to have sent the document. This is the role of the public key infrastructure (PKI). PKI will manage those certificates to assure the recipient that the digital signature attached to that document belongs to a particular person.

9.2. Commercial Off-the-Shelf Components

"COTS components are more susceptible to vulnerabilities than custom code because the "hacker" has access to the same third-party knowledge about the component as the system integrator." [HCPI98]

In theory, the security mechanisms employed in building information systems should be a function of the environment and the perceived threat[1]. In practice, to keep the manufacturing costs of such information system down, most of the producers use commercial off-the-shelf (COTS) components. As a result, COTS producers dictate the available security mechanisms. Because most of COTS are conceived for broad range of systems, their security is not tailored to

[1] As previously mentioned, the military systems need more than risk management.

meet specific needs. While volume of such components makes them cheap and readily available, they are very difficult to integrate into secure and reliable system. Consequently, the information system architect has to determine how to best use the generic COTS security mechanisms and how to enhance these mechanisms to achieve desirable level of security. Often COTS components are not thoroughly analyzed for security vulnerabilities and this makes security architect's task even more difficult [HCP198].

When building a computer system you need to design its architecture, build your own components and purchase other ones, integrate and test the system. Purchased components would consist of COTS and third-party products. Integration of the components into a trustworthy system is made more difficult because of purchased components and in particular COTS ones. The COTS general-purpose components such as compilers and routers have well defined interfaces that support application programming interface (API) or standard protocols. While desired functional properties are predictable, the trustworthy properties are difficult to anticipate. Probably the biggest uncertainty about using COTS products in the system with respect to security is the cascading effect, when flaw in one of such components is causing breach in another component.

The features of the COTS products are prioritized according to existing and projected customer demand, the actual competition and forecast technology directions. It is often the case, especially for software products, that to beat the competition with the earliest arrival on the market, the product gets a multitude of functional features, but suffers in the trustworthiness. To cut down on time-to-market, software developers reduce to a minimum all activities that are not related to actual coding. This results in very poor documentation, especially about system requirements specification. For trustworthy commercial products, the requirements specification needs to address all the possible attacks on the system trustworthiness, what defenses are available in COTS products and how these attacks will propagate through the system. If you assume that all COTS components are completely vulnerable and bear all the defenses on locally developed components, your product might be prohibitively expensive.

COTS software components developed during PC era tended (and still do) to ignore trustworthiness aspect in favor of features, impressive user interface, user-friendliness and short deployment time. In some cases, the products are so successful that they are used in settings not intended by developer. For example, Web browser developers did not foresee the popularity of

their application and widespread use in security sensitive areas, such as banking or e-commerce. When PCs were used as isolated desktops, the damage resulted from unreliable applications could be fixed and tolerated by the users. Such failures might have resulted in losing valuable data, but were not life-critical and these failures had no way of propagating to other machines. The trustworthiness of PC applications is well summarized by shrink-wrap licenses – end user license agreements (EULA), whose primary features are disclaimers of any responsibility by the application developers (see 9.3.3).

PC software developers realize the importance of market share for their financial success and survival. The early applications become de facto standards upon which other applications are built and measured against. To shorten the time to market, developers release beta versions to interested groups of users, who actually perform testing for the developers. Even then, product versions are released with non-critical bugs a.k.a. known issues and the users have to procure updates and patches later on. While this practice could be tolerated in the isolated PCs environment, it is intolerable to employ such COTS into high consequence information systems.

9.3. Obstacles to Security

Security, often viewed as burdensome to users frequently hurting the performance of the system, becomes appreciated only when it fails. Users seem unable to value security mechanisms, except when they experience a significant damage from the incidents. Hence, the problems here are societal and while the industry could foster highly secure systems, it is not likely to happen in advance of directly experienced or largely publicized incidents to the information systems. This vulnerability could be lessened to some extent, by user education, standards, procedures and law controls.

Furthermore, the external pressures such as the organizational culture and specificity of the working environment can aggravate security problems in information systems. For example, it is useful from a security point of view, to shorten the time of inactivity on the terminal to automatically log off the user, to limit the probability of its being used by unauthorized users. However, this requirement conflicts with the need to immediate access to patient information in emergency rooms.

Some of the security obstacles can be resolved through practical management controls, but the managers must have sensible tools and robust framework provided by technical controls.
For example, most organizations secure their intranet using firewalls. There is still a problem when two geographically different divisions of the same organization are connected over the Internet. Both divisions have secure firewalls, but each division will be insecure when they have to communicate over the Internet. A secure channel between the two firewalls is required.

9.3.1. Software Life Cycle

There are well-known benefits of following established methods for developing software products. However, it is often the case where such methods are not employed; some of them are viewed as unnecessary and misdirected activities, since they do not involve coding and are thought to increase time to market. The most neglected activity in project development is the documentation, especially on the system requirements specification. Specification for the system requirements results from the discussion between departments or groups participating in the system development and should include trustworthy requirements. The difficulty of expressing trustworthy requirements is compounded by the problem of predicting the effects of certain features. For example, the client might insist on a remote access to the information in a way that is incompatible with keeping integrity and secrecy of the communication.

9.3.2. TCP/IP

TCP/IP, as it exists today, is lacking security. This lack of security has lead directly to the development of tools and techniques that exploit TCP/IP weaknesses. Examples of such attacks include SYN flooding, IP spoofing and connection hijacking. Some of these flaws can be fixed with TCP Wrappers, Kerberos and Simple Key-Management for Internet Protocols (SKIP). However, these tools are not always in widespread use. Your host may implement them, but hosts you want to communicate with may not. Thus, most communication on today's Internet is still unsecured. Migrating to a new protocol suite, such as IPv6, may be the only way to fix the flaws in TCP/IP.

9.3.3. Liability Disclaimers

PCs are not bundled with specialized applications, so the buyer of such application has to contact directly the developer of the application, because PC and software vendors will most likely ignore any requests for compensation. The developer is protected by the license agreement and the most a user can get is a free update or patch to the application. It is worth noting that such license disclaimers are void in some countries or states, so the developer could be sued when the application usage causes any damage. Though, there is still to be seen any lawsuit brought against Microsoft for Hotmail security gaff[1] in August 1999.

Software products have a very odd product quality model. It is unlike a car, a telephone or a house. If you buy a product and get harmed because of a manufacturer's defect, you can (successfully) sue. Car makers cannot get away with building vehicles that explode on impact, contractors cannot get away with selling house with floors falling down. The constructor cannot say thing like, "Sorry. Wait for house 1.1. It will have stable floors." Software is different. It is sold without any claims whatsoever. Your word processor can accidentally corrupt your files and you have no recourse. Your firewall can turn out to be completely ineffectual, hardly better than having nothing at all and yet, you can do nothing about it. Microsoft never bothered to apologize for Hotmail bug, let alone provide any compensation for the subscribers.

Software manufacturers do not have to produce a quality product because there is no liability. The effect of this for security is that manufacturers do not have to make products that are actually secure, because no one can sue them if they make a bunch of false claims of security.

9.4. Open Problems

There is no such thing as the security problem; the problems of information security are complex and plentiful. It is of utter importance to first identify them, to assess the threats and then to prioritize the work that needs to be done. However, you should realize that the most challenging security problems are caused by all-to-all connectivity and the explosive growth of the Internet.

[1] The Hotmail exploit apparently took advantage of a bug in the start script that processed a login session between a Web browser and a server. As a result, anyone with simple HTML code, posted on hacking-related Web sites, was able to connect to a Hotmail server by typing in a user name without requiring a password.

The Internet is an international collection of data networks, presently based on TCP/IP protocols. It has no central authority. Although, Cobb states that the Internet in the U.S. is "...essentially operated by the U.S. government" [Cobb90:360-361], we do not share this opinion. This is untrue today and it was very unlikely when the Cobb's book was published in 1990. The majority of the Internet is owned, managed and legally controlled by entities other than the federal government. Although the government has to take the lead obviously, its role is more like a cheerleader or regulator to these other entities.

In fact, the only central authority is the Network Information Center, whose authority, by consensus, not law, governs mapping of IP addresses and domain names in the Internet Domain Name Service (DNS). Besides this limited mandate, there is no controlling body on the Internet (so far). Internet standards, generated by the Internet Engineering Task Force (IETF), have no force of law and are enforced only by mutual consent and the need for mutual interoperability. It is impressive that the Internet is flourishing in such environment.

Many attacks in the past have disrupted parts of the Internet. The worm released by Robert Morris Jr., in 1988 halted almost 6 000 computers in a matter of hours and made them unusable for days. In May 2000, ILOVEYOU virus apparently, caused billions of dollars in damage [Yudk00].

Consequently, here are some questions on security of the Internet:

1. What would be the costs of halting, say 25% of the network for 24 hours?
2. Is it possible to totally "break" the Internet?
3. Who can prevent it from happening?
4. Can we make it as robust and secure as it used to be?
5. Who is responsible for making it secure?
6. Is anyone listening? Does anyone care? Do you?

All these questions are very important and the answers to them should be provided by a separate study.

9.5. Remarks

Coming to the end of this thesis, we would like to reflect on its real contributions to the area of information security. There has been no attempt to find a solution to a particular problem, rather to provide a broad perspective of information security, identify technical solutions to the existing and very likely future problems, evaluate existing technical solutions and provide recommendations where the improvements can be made.

We emphasized that technical controls are only part of information system security. These controls are very important and interesting from a research point of view, because certain improvements in technical controls, take some load off the non-technical controls. We underlined the importance of the software life cycle, prudent engineering, significance of quality of service and user education in secure information systems.

We have learned that security is not a product, but a process. The only way to effectively do business in an insecure world is to recognize the inherent insecurity in the products and move these insecurities to places in the system where they are no longer vulnerable to a specific threat.

We underlined that perfect security is unachievable. For the military, it has to be as perfect as possible, (almost) regardless of the costs. For the business organizations, security involves risk management. For example, the credit card industry estimates the losses due to fraud. They know that losses from phone credit card transactions are about five times the losses from face-to-face transactions (when the card is present). Losses from the Internet transactions are many times those of phone transactions hence, no wonder they are the driving force behind SET.

The role of the firewall is evolving with the changing usages of the Internet as illustrated by the emergence of intranets and the need to protect internal in addition to external boundaries. Firewalls and other security measures have to be bolted on, because adequate security is not built in; however the long-term solution, e.g. IPv6, has to integrate security. Encryption plays a central role in this infrastructure. There are signs of major structural changes in the firewall market over the next couple of years. Firewall vendors are starting to co-operate and standards are emerging. Both of these factors should benefit organizations that currently struggle to choose the right product.

We have discussed the code-signing solution, the sandbox approach and the combination of the two. Furthermore, combining these with firewalling could provide an extra layer of security. The proof-carrying code and software fault isolation (see chapter 10 for more details) approaches are still being researched, but their ability to enforce some security policies will provide us valuable tools for protection against untrusted code.

Today, we are only beginning to realize the security implications of mobile code. As new security techniques are introduced, new bugs and exploits are found that circumvent these techniques. It is therefore important to continue research in this field in order to find an efficient and secure method of running mobile code.

We discussed the danger of building a system with COTS components. It is likely that information systems will be built from COTS operating systems and other applications to reduce cost, development time and risk of project timely delivery. These other applications include network protocols, database systems, compilers, e-mail, Web servers and other system tools. It is difficult to assemble a secure system by using COTS software components, because of limited access to internals of such components or controls over their design.

COTS software components tend to ignore trustworthiness aspect in favor of features, impressive user interface, user-friendliness and short deployment time. The trustworthiness of PC applications is well summarized by shrink-wrap licenses – end user license agreements (EULA), which primary features are disclaimers of any responsibility by the application developers.

We underscored the importance of user education in secure information systems. People seem unable to value security mechanisms, except when they experience a significant damage from the incidents. While the industry could foster highly secure systems, it is not likely to happen in advance of directly experienced incidents to the information systems. If the goal of secure information system is to be fulfilled, it requires pooling and organization of involved parties, including users.

Lastly we have stated big, open problems, all involving the Internet. The challenges here are political at least as much as they are technical. Regulation and new law in some form are required.

We have made every effort to provide practical assistance to those seeking protection for their information, to analyze existing technical controls, to point out weaknesses of today's technologies and to provide recommendations for the research. However, we realize that information security is a very complex undertaking and it is impossible to explore every possible set of threats, particularly those that have not yet materialized but which new developments in science and technology have made possible.

We would like to terminate this thesis with the analysis of the emerging security technologies as well as the recommendations for users and system builders. You will find these in the next chapter.

Chapter 10

Recommendations

At present, operating systems (OS) do not provide protected execution environments for processes. This imposes limits on what applications can do to protect themselves against users and against other applications. The fundamental insecurity of most operating systems undermines efforts to develop trustworthy applications. We should question claimed security by the applications, even if they offer evident security functionality. For example, Web browsers now incorporate cryptographic mechanisms to protect against wiretapping attacks. However, the keys are optionally protected through encryption with user-selected password and stored in a file system, managed by an insecure OS. Hence, neither cryptography, nor other application level mechanism will provide protection in the face of insecure OS. We need more research to find out how responsibility for secure environment should be split between OS and the applications.

For some applications, security properties are best enforced through cryptographic means. For example, security for e-mails entails confidentiality, sender authentication, message integrity and possibly non-repudiation with proof of submission and/or receipt. And because implementing cryptographic protocols is subtle, a number of efforts are made to free application developer from this task. The Internet Emergency Task Force (IETF) has developed series of specifications for making simplified, cryptographically protected (stream or message) communications available using the Generic Security Services Application Programming Interface (GSSAPI). Intel's multi-layered Common Data Security Architecture (CDSA) API aims to provide an integrated framework for cryptography, key and certificate management, and related services. CDSA has been submitted to the Open Software Foundation[1] for adoption as a standard.

The motivation for a robust, broadly available, multi-platform, industry standard security infrastructure is clear. The definition of such an infrastructure however, must accommodate the emerging Internet and intranet business opportunities and address the requirements unique to the most popular client systems, namely: personal computers (PCs) and networked application servers.

[1] Open Software Foundation merged with X/Open Company Ltd. in 1996 to form The Open Group.

10.1. Fine-Grained Access Control

Fine-grained access control (FGAC) enforces security policies at a low level of granularity [Raph99]. You can use it, for example, to limit an access to a particular record in the database for an authorized user. Hence the application can enforce much finer level of access to the database than operating system (OS). FGAC permits access control in accordance with principle of least privilege, which is an extremely effective defense against a large variety of attacks, including many that could be conveyed using foreign code. FGAC facilitates this defense, since it lets you confine accesses made by each software module to the minimal set of resources.

The access control mechanisms that are usually provided by operating systems (OSs) are too coarse and concern only certain resources. Almost no facilities are provided for controlling access to abstractions implemented above the OS level, including accesses that might be sensitive to the state of the resources being controlled and/or the state of the module requesting the access. A limited form of FGAC is available for Java programs running under JDK[1] security architecture, but state-sensitive access are not easily supported there and the technology is limited to programs written in the single programming language.

The downside of FGAC is the management complexity. Once FGAC support is in place, users and system administrators must configure access for all the resources. Based on experience with both discretionary and mandatory access policies, the setting of all permissions for FGAC, could be overwhelming. Then, if you are a programmer, you must either build suitable mechanisms or learn existing mechanisms when enforcing any particular application security policy.

There will always be more applications than OSs, applications will evolve faster and will be developed by a wider range of vendors. That is why the research on computer security in the past focused on securing OSs. Unfortunately, these efforts have been largely unsuccessful. It might be because modern applications tend to involve the security policies defined in terms of application-level abstractions [Melo99] rather than OS ones. Thus, it is obvious that enforcing security will be a responsibility shared between the OS and the applications. Research is needed to understand how the responsibilities might best be partitioned, which OS mechanisms can assist in application-level security implementation and how to specify and implement security policies within applications.

[1] JDK stands for Java Development Kit. Currently, JDK 1.3 is available.

10.2. Software Fault Isolation

Until recently, almost all operating system and hardware-implemented security mechanisms, have involved monitoring system execution. Actions whose execution would violate the security policy are intercepted and aborted, and all other actions are executed normally. An alternative approach is to execute only those programs that cannot violate the security policies.
It can be achieved in two ways:

- Modify a program before execution begins. This way it might be possible to prevent program behavior from violating the security policy being enforced.
- Analyze a program and prove that no program behavior will violate the security policy.

Both schemes depend on analysis techniques developed by programming language researchers and each requires incorporating program analysis into the trusted computing base.

The idea of program rewriting to enforce security was first proposed in connection with memory safety, a security policy stipulating that memory access are confined to specified regions of memory. The obvious approach of adding a test and conditional jump before each machine language instruction that reads, writes or jumps to memory can slow down the execution and renders it impractical. Software fault isolation (SFI) modifies instructions and addresses by AND-ing and OR-ing masks, so that they do not reference memory outside the specified regions [WLAG93]. The behavior of programs that never attempt illegal memory access is unaffected by the modifications and programs that would violate memory safety end up accessing legal addresses instead. Note that the use of program modifications to enforce security policies is not limited to memory safety. Any security policy that monitoring can enforce, can be enforced by using a generalization of SFI [Schf98].

10.3. Proof-carrying Code

Proof-carrying Code (PCC) applies mathematical techniques and formal methods to produce a proof so that a program can be trusted in advance of its use. The proof is constructed such that the operating system can detect any attempt to tamper with either the proof or the code. A program with PCC is executed only if an accompanying proof establishes that the security policies of interest will not be violated [Necu97]. When the program is loaded, a proof examiner attests that the (untrusted) code is safe to execute. This has potential in the system with dynamic trusted

computing base such as extensible operating systems and the Internet browsers capable of downloading code.

However, the feasibility of proof generation will depend on what security policy is being enforced. Initial version of PCC focused on ensuring that programs do not violate memory safety or attempt operations that violate type declarations. Nevertheless, the richer security policies can certainly be handled by PCC approach, granted we have more elaborate proof-generation and proof-checking methods available.

There are many advantages of PCC over related techniques. Because the untrusted code is verified statically, before being executed, we not only save execution time but we detect potentially hazardous operations early, thus saving the resources that might have been acquired by the rogue process.

SFI and PCC are very active areas of research. So far, each has been tried solely on relatively small examples and only on a few kinds of security policies. Both presume that an entire system will be subject to analysis, which is probably wrong for COTS products. There is a great deal of research to be done before we fully realize the ultimate impact of this technology, the practicality and limits of these approaches. Part of the research involves programming language semantics, compilers, sophisticated security policies and automated deduction. Other study involves testing the approaches for efficiency and interoperability.

SFI and PCC represent a new approach to the enforcement of some security policies, an approach in which programming language technology is leveraged to obtain mechanisms that are more efficient and better suited for the application-level security. Most programming today is done in high-level typed languages and good use might be made of the structural and type information that high-level languages provide. Moreover, certain security policies, like information-flow restrictions, cannot be enforced by monitoring execution but can be enforced by analyzing entire program prior to execution. Any security policies that can be enforced by a secure OS or by the use of hardware memory protection can be achieved by SFI or PCC [Schf98].

10.4. Recommendations for Users

Prudent information system users should take reasonable precautions to protect their information stored on local and network computer devices. In addition they should protect their information in transit over the Internet. The reasonable level of precaution will be different for every user and it depends on value of the information that needs the protection. (See chapter 2.2.) Here is a list of precautions that you, as a user of the information system, can employ immediately without much of an expense[1]:

- Assess your risks.
- Employ a good password [RuGa91:61] & [DdDp98:159-165] for the access controls to your information system.
- Learn basic security features of your information system. This should include password management, file encryption and file sharing permissions.
- Encrypt sensitive files, even when you store them off-line.
- Verify permissions on files that you want to share with others.
- Never send sensitive information across the Internet, such as social insurance number (SIN), credit card number, your address and phone number, unless the Web site is using secure communication protocol.
- Make use of encryption applications, such as Pretty Good Privacy (PGP), for confidential e-mails. (Refer to Appendix E and [Zimm95] for more details on PGP.)
- Back up important files[2]. This is your single best protection against data loss [Cobb90:285].
- Keep your anti-virus software up to date and make sure you understand which files and when are scanned with this software.
- Learn security features of your Web browser, such as disabling cookies, executing Java applets and ActiveX controls.
- Know what Web sites you are visiting.
- Use up-to-date browsers and keep lookout for security alerts.
- Delete all cookies after you are finished browsing the Web.
- Terminate browser application after the Web session or clear the cache of visited sites.
- While unattended, lock (through software or physical key) the shared workstation.

[1] Actually, there is no expense, except for the time.
[2] This thesis was backed up on three PCs and several floppy disks.

Because almost all Internet traffic is sent "in the clear" it can be read by software on any network device through which the network packets are routed. The sensitive information traveling across the Internet is user name, password, IP address and sensitive data files. A solution to this problem is to have these data sent across the Internet in a secure manner, that is, by encrypting them first. Currently, you cannot do much about this situation, except for frequently modifying your password and using Web sites that employ secure communication protocol. What is required is for system builders (and suppliers) to provide a more secure procedure for logging in across the network, as discussed in chapter 6.

10.5. Recommendations for Systems Builders

Suppliers of information systems should ensure that their products conveniently provide the users with capability to take the precautions described in the previous section.

There are two problems related to passwords. First, user name, password and IP address combinations are sometimes sent in the clear across the Internet. Packet sniffers [Dend99:184,185] may be used to read these combinations. Systems builders should provide protocols and software that encrypt these data at the source, or provide alternative systems that do not require passwords to be sent in the clear across the Internet. The second password problem that continues to be a concern is access to password files for password cracking. Systems builders should provide protocols and software that prevent access to files of encrypted passwords, or provide an alternative system that does not require encrypted passwords to be stored in files on systems accessible across the Internet.

Here are our recommendations for information system builders:

- Make sure the systems are delivered with reasonably secure settings. Provide a simple interface for administrators to modify these settings, if necessary.
- Speed up development of protocols and applications that provide practical privacy for the Internet users, for example, e-mail users.
- Set up the framework for successful e-commerce. This will involve large-scale deployment of PKI, IPSec protocol, e-cash and digital signatures.
- Accelerate development of authentication protocols for e-commerce.
- Offer protocols and applications for electronic funds transfer and e-cash.

- Supply protocols and applications that encrypt user name, password and IP address at the source, or offer an alternative system that does not require sensitive information to be sent in the clear across the Internet.
- Harden the systems against denial-of-service attacks.
- Develop applications for intrusion detection, monitoring of the attacks and identifying the intruders.
- Provide alternative authentication protocols that do no require access to files with encrypted password, accessible across the Internet.
- Develop system that will resist viruses and will recover from computer and connection failures.
- Deliver protocols for automatic transfer of copyright on payment of a fee.

Authentication is the most important technology for e-commerce. It should be impossible for the attacker to masquerade an authorized user. It should be unachievable to alter e-mail address, digital signatures or the content of any document. It should be unfeasible for the attacker to spoof IP address. These technologies exist today; they just are not used in practice.

Today, it is easy to get an unlicensed material from the Internet and many people routinely copy and distribute files, including licensed software, without permission of the author. There have been a lot of discussions on protocols for automatic transfer of copyright on payment of a fee, but none has been implemented yet.

References

1. [ABHK96] Derek Atkins, Paul Buis, Chris Hare, Robert Kelley, Carey Nachenberg, Anthony B. Nelson, Paul Phillips, Tim Ritchey and William Steen "Internet Security Professional Reference", New Riders, Macmillan Computer Publishing, 1996

2. [AbNe94] Martin Abadi and Roger Needham "Prudent Engineering Practice for Cryptographic Protocols", IEEE Transactions on Software Engineering, Vol. 22, No. 1, January 1996 (First appeared as SRC report 125 in June 1994.)

3. [Anon98] Anonymous "Maximum Security : A Hacker's Guide to Protecting Your Internet Site and Network", Sams, Macmillan Computer Publishing, 1998

4. [Appe99] Andrew W. Appel "Protection against untrusted code", 1999 (http://www-4.ibm.com/software/developer/library/untrusted-code/)

5. [Arno93] N. Derek Arnold "UNIX security : a practical tutorial", McGraw-Hill,1993

6. [Bell89] Steven M. Bellovin "Security Problems in the TCP/IP Protocol Suite", Computer Communications Review, Vol. 19, No. 2, pp. 32-48, April 89

7. [BeMe92] Steven M. Bellovin and M. Merritt "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks", Proceedings of the IEEE Symposium on Security and Privacy, pp.72-84, 1992

8. [Bent00] John T. Bentivoglio "Symposium on Healthcare Internet and E-commerce: Legal, Regulatory and Ethical Issues", March 2000 (http://www.usdoj.gov/criminal/cybercrime/healthsp.htm)

9. [BePa73] D.E. Bell and Leonard J. LaPadula "Secure Computer Systems: Mathematical Foundations", Mitre Report MTR 2547, Vol. 2, 1973

10. [BiSh92] E. Biham and A. Shamir "Differential Cryptanalysis of the Full 16-round DES" Advances in Cryptology - Crypto '92. Springer-Verlag, pp. 487-496, 1992

11. [BKRo94] M. Bellare, J. Killian and P. Rogaway "The Security of the Cipher Block Chaining", Advances in Cryptology - Crypto '94, Springer-Verlag, pp. 341-358, 1994

12. [Brun00] Mark Brunker "Vast online credit card theft revealed", March 2000 (http://www.msnbc.com/news/382561.asp?cp1=1)

13. [CERT96] Computer Emergency Response Team (CERT) "TCP SYN Flooding and IP Spoofing Attacks", Sept. 1996. CA-96:21 (http://www.cert.org/advisories/CA-96.21.tcp_syn_flooding.html)

14. [ChBe94] William R. Cheswick, Steven M. Bellovin "Firewalls and Internet Security : Repelling the Wily Hacker", Addison-Wesley Publishing Company, 1994.

15. [ChZw95] Brent D. Chapman and Elizabeth D. Zwicky "Internet Security: Building Internet Firewalls", O'Reilly & Associates, 1995

16. [Cisc1]Cisco Systems Inc. "Defining Strategies to Protect Against UDP Diagnostic Port Denial of Service Attacks", http://www.cisco.com/warp/public/707/3.html

17. [Cisc2] Cisco Systems Inc. "Configuring TCP Intercept (Prevent Denial-of-Service Attacks)", http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/secur_c/scprt3/scdenial.htm

18. [Cisc96] Cisco Systems Inc. "Defining Strategies to Protect Against TCP SYN Denial of Service Attacks", September 1996 (http://www.cisco.com/warp/public/707/4.html)

19. [Cobb90] Steven Cobb "The Steven Cobb complete book of PC and LAN Security", Windcrest, 1990

20. [CSA92] Canadian Standards Association "Information Processing Systems-Open Systems Interconnection-Basic Reference Model-Part 2: Security Architecture (CAN/CSA-Z243.100.2-92)

21. [DdDp98] Dorothy E. Denning and Peter J. Denning "Internet Besieged: Countering Cyberspace Scofflaws", Addison-Wesley, 1998

22. [Dend82] Dorothy E. Denning "Cryptography and data security", Addison-Wesley, 1982

23. [Dend99] Dorothy E. Denning "Information Warfare And Security", Addison-Wesley, 1999

24. [Denp91] Peter J. Denning "Computers Under Attack: Intruders, Worms, and Viruses", Addison-Wesley, 1991

25. [DiHe76] W. Diffie and M.E. Hellman "New Directions in Cryptography", IEEE Transactions on Information Theory, pp.109-112, November1976

26. [DoD85] Department of Defense Trusted Computer System Evaluation Criteria, DoD 5200.28-STD, December 1985

27. [Elli70] J.H. Ellis "The history of Non-Secret Encryption" http://www.cesg.gov.uk/about/nsecret.htm

28. [EWSh97] M. David Ermann, Mary B. Williams, Michele S. Shauf "Computers, Ethics, and Society", Oxford University Press, 1997

29. [FaSp94] Daniel Farmer, Eugene H. Spafford "The COPS Security Checker System", Purdue University Technical Report CSD-TR-993, January 22, 1994

30. [FeMa98] E. Feustel and T. Mayfield "The DGSA: Unmet Information Security Challenges for Operating System Designers", Operating Systems Review, Vol. 32, pp. 3-22, 1998

31. [Ferg98] P. Ferguson "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", Network Working Group, RFC 2267, January 1998

32. [Ford94] Warwick Ford "Computer communications security : principles, standard protocols, and techniques", Prentice Hall, 1994

33. [Garf00] Simson Garfinkel "Database Nation : The Death of Privacy in the 21st Century", O'Reilly & Associates, January 2000

34. [GaSp96] Simson Garfinkel and Gene Spafford, "Practical UNIX and Internet Security" Second Edition, O'Reilly & Associates, Inc., 1996.

35. [GLNS93] Li Gong, M.A. Lomas, R.M. Needham and J.H. Saltzer "Protecting Poorly Chosen Secrets from Guessing Attacks", IEEE Journal on Selected Areas in Communications, Vol. 11, No. 5, pp. 648-656, 1993

36. [Goll99] Dieter Gollmann "Computer Security", John Wiley & Sons, 1999

37. [Graf96] Mark Graff "Sun Security Bulletin 00136", Oct 1996 (http://sunsolve1.Sun.COM/pub-cgi/retrieve.pl?doctype=coll&doc=secbull/136&type=0&nav=sec.sba)

38. [Hall94] Neil Haller "The S/Key One-time Password System", Proceedings of the ISOC Symposium on Network and Distributed System Security, 1994

39. [HBHo95]Arthur E.Hutt, Seymour Bosworth, Douglas B.Hoyt "Computer Security Handbook", third edition, Wiley, 1995

40. [HCPl98] Scott A. Hissam, David Carney and Daniel Plakosh "DoD Security Needs and COTS-Based Systems", September 1998 (http://www.sei.cmu.edu/cbs/papers/monographs/dod-security-needs/dod-security-needs.htm#section2)

41. [John96] William Johnston, "Unique DOE Security Requirements in the NGI" (A summary of "Public Key Infrastructure for DOE Security Research: Finding from U.S. Department of Energy, Joint Energy Research / Defense Programs Computing – related Security Research Requirements – Workshop – II, Dec 11-13, 1996, Albuquerque, New Mexico."

42. [Jonc95] Laurent Joncheray "A Simple Active Attack Against TCP", Proceedings of the 5th USENIX UNIX Security Symposium, Salt Lake City, Utah, 1995

43. [KaLa86] Richard Y. Kain and Carl W. Landwehr "On Access Checking in Capability-based Systems", Proceedings of the IEEE Symposium on Security and Privacy, pp. 95-100, 1986

44. [Kenn85] J.J. Kenny "Data privacy and security", Pergamon Infotech, 1985

45. [Leit97] Felix von Leitner "Chaos Computer Club Clarifications", www.tbtf.com/resource/felix.html, February 17, 1997

46. [Lewi98] Peter H. Lewis "Threat to Corporate Computers Is Often Enemy Within", The New York Times, March 2, 1998

47. [Losh95] Pete Loshin "Electronic commerce : on-line ordering and digital money", Charles River Media, 1995

48. [LRWo92] X. Lai, R.A. Rueppel, and J. Woollven "A fast cryptographic checksum algorithm based on stream ciphers", Advances in Cryptology – Auscrypt '92, Springer-Verlag (1992), pp. 339-348.

49. [McCa97] McCarthy M. Linda "Stories from the Trenches", Sun Microsystems Press, 1997

50. [McFe97] Gary McGraw, Edward W. Felten "Java Security", Wiley Computer Publishing, 1997

51. [Merk78] Ralph Merkle "Secure Communications Over Insecure Channels", CACM April 1978, pages 294-299. Submitted in 1975

52. [Melo99] Steven Meloan "FINE-GRAINED SECURITY: The Key to Network Safety: Strength in Flexibility", November 1999 (http://java.sun.com/features/1999/11/security.html)

53. [Mill98] Stewart S. Miller "Windows NT security guide", Digital Press, 1998

54. [Morr85] Robert T. Morris "A Weakness in the 4.2 BSD UNIX TCP/IP Software", Computing Science Technical Report 117. AT&T Laboratories, Murray Hill, N.J., 1985

55. [MOVa96] Alfred J. Menenzes, J. Paul C. Van Oorschot and Scott A. Vanstone "Handbook of Applied Cryptography", CRC Press, October, 1996

56. [NCES98] National Center for Education Statistics "Safeguarding your Technology", 1998 (http://nces.ed.gov/pubs98/safetech/)

57. [Necu97] George C. Necula "Proof-carrying Code", Proceedings of the 24th Symposium on Principles of Programming Languages, pp. 106-119, 1997

58. [NeTs94] Clifford B. Neuman and Theodore Ts'o "Kerberos: An Authentication Service for Computer Networks", IEEE Communications Magazine, 32, 1994. http://gost.isi.edu/publications/kerberos-neuman-tso.html

59. [NIST85] National Institute of Standards and Technology (NIST), FIPS Publication 113: Computer Data Authentication, 1985. (http://www.itl.nist.gov/fipspubs/fip113.htm)

60. [Perk98] Charles E. Perkins "Mobile IP Design Principles and Practices", Addison-Wesley, 1998

61. [Raph99] Denis Raphaely "Oracle8i Application Developer's Guide – Fundamentals", Oracle Corporation, 1999 (http://oradoc.photo.net/ora81/DOC/server.815/a68003/toc.htm)

62. [Schb96] Bruce Schneier "Applied Cryptography", Wiley, 1996

63. [Schf98] Fred B. Schneider "Enforceable Security Policies". http://cs-tr.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/TR98-1664, 1998

64. [Shan48] C.E. Shannon "Mathematical Theory of Communication", Bell System Technical Journal, vol. 27, pp. 379-423, July 1948

65. [Shan49] C.E. Shannon "Communication Theory of Secrecy Systems" Bell System Technical Journal, Vol. 28, pp. 656-715, October 1949

66. [Shan93] C.E. Shannon "Claude Elwood Shannon : collected papers", Institute of Electrical and Electronics Engineers, 1993

67. [Simo96] Fred Simonds "Network security: Data and Voice communications", McGraw-Hill, 1996

68. [SKKS96] Christoph L. Schuba, Ivan V. Krsul, Markus G. Kuhn, Eugene H. Spafford, Aurobindo Sundaram, Diego Zamboni, Purdue University "Analysis of a Denial of Service Attack on TCP", 1996 (http://www.cs.purdue.edu/homes/clay/netsecpapers.html)

69. [SpLa89] David L.Spooner, Carl Landwehr "Database security, III: status and prospects", Elsevier Science, 1989

70. [Spaf92] Eugene Spafford "OPUS: Preventing weak password choices", Computers & Security 11, No. 2, pp. 273-278, 1992

71. [Stei98] Steve G. Steinberg "Deflating this month's overblown memes." Wired Magazine, February, 1998. (http://www.wired.com/wired/archive/6.02/hypelist.html)

72. [WrSt94] Gary R. Wright and W. Richard Stevens "TCP/IP Illustrated", Volume 2, The Implementation, Addison-Wesley, 1994

73. [Stin95] Douglas R. Stinson "Cryptography Theory and Practice", CRC Press, 1995

74. [Stol89] Clifford Stoll "The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage", Doubleday, 1989

75. [Summ97] Rita C. Summers "Secure Computing", McGraw-Hill, 1997

76. [THMH99] Mary Thompson, William Johnston, Srilekha Mudumbai, Gary Hoo, Keith Jackson, Abdelilah Essiari "Certificate-based Access Control for Widely Distributed Resources", proceedings of the Eighth Usenix Security Symposium, Aug. 1999 (http://www.usenix.org/publications/library/proceedings/sec99/thompson.html)

77. [WLAG93] Robert Wahbe, Steven Lucco, Thomas E. Anderson and Susan L. Graham "Efficient Software-based Fault Isolation", Proceedings of the 14th ACM Symposium on Operating System Principles, pp. 203-216, 1993

78. [Yudk00] Chaim Yudkowsky "The `ILOVEYOU' virus offers us numerous lessons" (http://www.bizjournals.com/louisville/stories/2000/06/05/smallb7.html)

79. [Zimm95] Philip Zimmermann "The official PGP user's guide", MIT Press, 1995

Acronyms

ACM
Association for Computing Machinery

AES
Advanced Encryption Standard

ANI
Automatic Number Identification

ANSI
American National Standards Institute

ARPA
Advanced Research Projects Agency

CGI
Common Gateway Interface

CISSP
Certified Information System Security Professional

COMPSEC
Computer Security

COMSEC
Communications Security

CPU
Central Processing Unit

CSL
Computer Systems Laboratory

CTCPEC
Canadian Trusted Computer Product Evaluation Criteria

DES
Data Encryption Standard

DNS
Domain Name Service

DSS
Digital Signature Standard

EDI
Electronic Data Interchange

EES
Escrowed Encryption Standard

FIPS
Federal Information Processing Standards

FIRST
Forum for Incident Response and Security Teams

FTP
File Transfer Protocol

HTML
Hypertext Markup Language

HTTP
Hypertext Transfer Protocol

ICMP
Internet Control Message Protocol

IDEA
International Data Encryption Algorithm

IEEE
Institute of Electrical and Electronics Engineers

IETF
Internet Engineering Task Force

IP
Internet Protocol

$(ISC)^2$
International Information Systems Security Certification Consortium

ISO
International Organization for Standardization

ITSEC
Information Technology Security Evaluation Criteria

LAN
Local Area Network

MAC
Message Authentication Code

MTA
Message Transfer Agent

Multics
Multiplexed Information and Computing Service

NCSC
National Computer Security Center

NFS
Network File System

NIST
National Institute of Standards and Technology

NNTP
Network News Transfer Protocol

NSA
National Security Agency

OECD
Organization for Economic Cooperation and Development

OSI
Open Systems Interconnection

PC
Personal Computer

PEM
Privacy Enhanced Mail

PKI
Public-Key Infrastructure

RPC
Remote Procedure Call

RSVP
Resource Reservation Protocol

SEC
Securities and Exchange Commission

SET
Secure ElectronicTransactions is the credit card transaction protocol

SMTP
Simple Mail Transfer Protocol

SNMP
Simple Network Management Protocol

SSL
Secure Socket Layer

TCB
Trusted Computing Base

TCP
Transmission Control Protocol

TCSEC
Trusted Computer System Evaluation Criteria

TEMPEST
Transient Electromagnetic Pulse Standard

TSD
Telephone Security Device

UDP
User Datagram Protocol

URL
Uniform Resource Locators

WAN
Wide Area Network

Appendix A – Hacker

The definitions of what constitutes a "hacker" abound and these definitions vary according to the socio-political position of the defining group or individual. The hackers are either viewed as computer enthusiasts who have an ardent interest in learning about computer systems and how to use them in innovative ways, or malicious hackers who deliberately crash systems and delete files [Cobb90:485-486]. The hacker can be portrayed as either a hero or a villain depending on your point of view. In the absence of the single definition, both images are correct. You are left to decide which image is of a true hacker.

Often, the media reports reinforce the stereotypical image of the hacker as a teenage loner, devoid of social skills, who is often petty and malicious in their actions and hold absolutely no morals or ethics whatsoever. Reports like these simply perpetuate the popular image of the lonesome computer criminal, without making crucial divisions between the anarchists and the explorers.

Yes, there are hackers who crash systems intentionally, but they certainly do not comprise the overwhelming majority of hackers; they are in fact only a small percentage. Many hackers, as is their primary intention, go completely unnoticed on the systems they choose to hack and are never discovered.

It seems however, that the public pays a lot of attention to hyped media reports on computer security breaches. For example, the arrest of Kevin Mitnick on Feb. 14, 1995 was one of the most celebrated arrests in recent U.S. history. Kevin Mitnick faced 23 counts of computer and telecommunications fraud. Since the early 1980s, Mitnick had managed to break into computers of software giants like Digital Equipment and had cracked a number of phone systems. He managed to obtain the user lists for Netcom, the largest Internet service provider in the U.S. These lists contained personal information including passwords, phone and credit-card numbers.

However, unlike those of some better-known hackers, Mitnick's motives were not larcenous. He bilked phone companies by illegally commandeering phone lines and pilfered private e-mail from Internet users. It was all illegal, but hardly a threat to national security. Some hackers, like infamous Justin Petersen, did the stolen credit-card numbers for profit and won cars and money from radio contests by hijacking the phone system. In comparison, Mitnick was small potatoes. He got in trouble, because one of the computers he cracked belonged to a San-Diego

computational physicist and security expert, Tsutomu Shimomura. Shimomura later said that catching Mitnick was a matter of honor, and it is easy to believe him. It does not look good when someone breaks into the computer of a security expert who has worked for the National Security Agency (NSA) and the United States air force. Shimomura is every bit as much of a hacker as his quarry. The difference is that he uses his powers and knowledge for good instead of evil, although it is questionable whether working for the shadowy NSA necessarily puts him on the side of good.

Appendix B – Electromagnetic Radiation

In April 1985, a Dutch scientist Wim van Eck published a paper, which explained how electromagnetic radiation from video display devices could be used for eavesdropping. An appropriately outfitted van can park near the site and remotely pick up all of the keystrokes and messages displayed on computer monitor. This would compromise all the passwords, messages, etc.

Many machines, such as computers, terminals and modems emit radiation that can be picked up at some distance. If radiation from the terminal can be picked up and recorded, this can be converted into the data that are transmitted.

Military and intelligence agencies use trucks with sensitive antennas and processing equipment to test the security of their own installations, and take special precautions to screen rooms and maintain physical control over areas from which radiation could be recorded. Using special equipment, they can analyze the emitted radiation generated by computer equipment and determine the cause of the radiation.

This kind of eavesdropping is not widely used since other forms of eavesdropping are much easier. However, if no protective measures are taken against it, it does not need to be excessively expensive. As with all eavesdropping, the intruder may have to listen for a long time before anything worthwhile is intercepted. Electromagnetic eavesdropping is complicated by the fact that the signals obtained are often confused, incomplete, or mixed with noise. The radiation from central processing unit (CPU) itself is generally not worth recording because it is too complex to be unscrambled without great expense. On the other hand, to utilize radiation from a terminal is not too expensive. The radiation from a terminal can be converted into data without expensive equipment, providing the receiver is close enough. Some devices transmit the same signal repetitively. Most visual display terminals, for example, continually scan the screen, refreshing the image. A signal containing the data on the screen may be retransmitted hundreds of times. Repetition makes it possible to reconstruct a very weak or noisy signal a long way from the terminal. However, the cost of the necessary equipment is high. Vital data such as security codes would not be displayed on the screen and they will not be retransmitted repetitively.

The first line of defense against electromagnetic eavesdropping lies with the equipment manufacturer. Machines can be designed so that the level of radiation they emit is not high. Then, several other measures can be taken to protect information, if electromagnetic radiation is a serious concern. You should ensure that inexpensive detection equipment could not come close enough to the terminals, to cause trouble. Locate terminals handling sensitive data toward center of the building, rather than against the outside walls. Often the best location to pick up radiation is the adjacent office to where a terminal is used. It is advisable not to place a sensitive terminal in the office next to one owned by a different organization (especially competitor).
It is often difficult to tell where the radiation will be detectable without measuring it. Simple physical changes can substantially lower the detectable emissions, such as moving the terminal to the other side of the room, erecting electrical screening, such as the metal mesh of window screens, moving cables or conduits, grounding all metalwork.

However, before intruder needs to use electromagnetic eavesdropping method, there are many simpler means of eavesdropping open, which include:

- Wastebasket
- Visual Eavesdropping
- Bugging devices

Appendix C – Trusted Computer Systems

Trusted Computer System Evaluation Criteria (TCSEC) document was first issued in 1983 and then updated in 1985 and issued as a Department of Defense (DoD) standard.
TCSEC is an outgrowth of Project MAC of Massachusetts Institute of Technology (MIT) that was working on the development of computer operating systems in the late 1960s and early 1970s. At the time, the National Bureau of Standards (now known as NIST) contracted MIT and Honeywell Corp. to use Project MAC as a basis to develop a set of criteria for evaluation of systems requiring high trust.
This produced the Multics Operating System and the TCSEC. Multics was utilized to test the security concepts embodied in TCSEC and DoD confidentiality oriented policy to ensure against unauthorized disclosure of sensitive information.

The TCSEC known as the Orange Book because it was published with a distinctive orange cover classifies computer systems into several categories, based on their security requirements. Although originally written for military systems, the security classifications are now broadly used within computer industry.
The TCSEC approach has been criticized for its narrow policy goals, since it mainly targets monolithic, multi-user, centralized operating systems. To supplement TCSEC on databases and networks, Trusted Database Interpretation or Trusted Network Interpretation should be used.

The National Computer Security Center (NCSC) is part of the National Security Agency (NSA) and the DoD. The NCSC is responsible for establishing policy and procedures for securing information deemed of value to the U.S. government. NCSC has developed, promoted and published several standards, including TCSEC.

The TCSEC standard establishes a basis for evaluating operating system security. The NCSC applies tests to software and hardware configurations, submitted by vendors and rates them in accordance with the TCSEC criteria. The application of these procedures assures both the government and the commercial buyer that the products meet designated levels of performance and expectation.

The TCSEC divides systems into four hierarchical divisions and the requirements for each group are divided into the following categories: Security Policy, Marking, Identification, Accountability, Assurance and Continuous Protection.

The TCSEC security categories range from D (Minimal Protection) to A (Verified Protection) as follows:

- Division D: Minimal Protection
- Division C: Discretionary Protection Class C1 and C2
- Division B: Mandatory Protection Class B1, B2 and B3
- Division A: Verified Protection

The product is classified as Division D if it does not comply with any other category, or has failed to receive a higher classification. DOS-based PCs fall into this category.

The principal difference between C and B class systems is in the area of access control. In C2, object owners decide who can access their object, to what degree (Discretionary Access Control). In B, objects have a security classification independent of the owner. So if a user receives a copy of an object marked "secret", the user cannot change the classification of this object, or give it to others who are not cleared to access "secret" objects (Mandatory Access Control). B class systems need "data labeling" to implement MAC.

Division A contains all level B3 features and requirements and provides high degree of assurance that implementation is correct through formal design specifications and formal verification procedures.

Appendix D – Kerberos

Kerberos[1] is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography. It verifies that a user requesting access to resource is not an imposter. To accomplish this, it encrypts passwords transmitted around networks and uses a database, an authentication server and ticket-granting server. These components reside on a single, physically secure device that is separate from other server on the network.

When you log on to a server, using a password and requesting a service on a host computer, your password is transmitted to the authentication server. The Kerberos server authenticates you by looking up information in the Kerberos database. If you have access rights, the authentication server creates a ticket for the ticket-granting server. This ticket is a credential used to authenticate one principal[2] to another. This ticket contains user name, the name of the ticket-granting server, the time, a time limit for which the ticket is valid, your network address and a randomly generated private session key. The information is encrypted using DES with a key shared by authentication and ticket-granting servers. The encrypted information is sent back to you along with a copy of the random session key, again encrypted using private key shared by the Kerberos server and you. Upon receiving this information, your password is converted to DES key and used to decrypt the response from the Kerberos server. Upon successful decryption, your password and DES key are erased while the ticket and the session key is stored.

To gain access to the host computer with the requested service requires that the ticket and an authenticator (encrypted by the session key sent by the Kerberos server) is sent by the user to the server, which decrypts the ticket using the common key it shares with the Kerberos server. By comparing the information in the ticket to that in the authenticator and the address, the server can permit or deny access to the desired service.

[1] Sometimes referred to as Kerboros [Stin95:269]
[2] Principal is a client or server instance that participates in a network communication.

Appendix E – Pretty Good Privacy

Pretty Good Privacy (PGP) from Philip Zimmermann's Pretty Good Software[1], is a program for encrypting files and electronic mail. Unlike most commercial software packages such as Microsoft Word, it uses "strong" encryption[2]. PGP combines public-key cryptography with conventional cryptography and works on virtually every platform. PGP is fast, offers sophisticated key management, digital signatures and data compression.

PGP combines the convenience (of key distribution) of the RSA public-key cryptosystem with the speed of conventional cryptography. For conventional encryption, it uses an algorithm called IDEA (see 5.2.2 for details) in 64-bit CFB mode. Because the public key encryption algorithm is much slower than conventional single-key encryption, encryption is better accomplished by using a high-quality fast conventional single-key encryption algorithm to encipher the message.

The keys are kept in the key certificates. A key file or a key ring contains one or more of these key certificates. The public/secret key pair is derived from large truly random numbers deduced mainly from measuring the intervals between the user's keystrokes with a fast timer. The software asks you to enter some random text to help it accumulate some random bits for the keys. Some of the randomness is derived from the unpredictability of the typed content. The generated key pair is placed on your public and secret key rings. The public key file can be sent to the others, for inclusion in their public key rings. Each secret key is also encrypted with its own pass-phrase[3], in case it falls into the wrong hands. This pass-phrase is needed every time a secret key is used. The pass-phrase must not be lost, since there is no way to recover it. Of course, the secret key should never be disclosed to anyone else. You should always keep physical control of the secret key and do not risk exposing it by storing it on a remote timesharing computer. You should keep it on your own personal computer.

While PGP has the ability to conceal, it is in the area of identity that this software makes its most positive contribution. The absence of real bodies leaves a place for ambiguity on everyone who interacts on the Net. To this dilemma, PGP provides an unambiguous solution: digital signature.

---

[1] Philip Zimmermann is a software engineer consultant, specializing in embedded real-time systems, cryptography, authentication and data communications.
[2] Longer the encryption key, stronger the encryption will be. However, longer keys make the application slower. It is up to the user to decide which key-length is appropriate.
[3] Pass-phrase is like a password, except that it can be a whole phrase or sentence with many words, spaces, punctuation, or anything else. The pass phrase is case-sensitive.

To create a digital signature, PGP encrypts the message (message digest) with the secret key. The message digest is a compact (128 bit) "fingerprint" of the message, which is calculated by the sender's software. The sender's secret key is used to encrypt the message digest and an electronic timestamp is added, forming a digital signature, or signature certificate. PGP uses 128-bit message digests to form signatures. Documents are signed by prefixing them with the signature certificates, which contain the sender's key ID[1], a message digest of the document signed with the secret key and a timestamp of when the signature was made. The receiver's software automatically looks up the sender's public key and user ID in the receiver's public key ring to verify the signature. PGP uses MD5 Message Digest Algorithm, placed in the public domain by RSA Data Security Inc.

In public-key cryptosystem, the public keys must be protected from tampering, to make sure that a public key really belongs to whom it appears to belong to. A public key downloaded from a bulletin board must not be trusted, unless, it is signed by a trusted person. It is your responsibility to protect the public key ring from tampering. You should maintain the physical control of the public key ring, preferably by storing it on your personal computer, just as in case of the secret key. This is to protect it from tampering, not from disclosure. The trusted backup copy of the public and secret key ring should be kept on a write-protected media.

PGP keeps track of which keys on your public key ring are properly certified with signatures of introducers, you trust. All you have to do is tell PGP which people are trusted as introducers, and certify their keys with your own ultimately trusted key. PGP can take it from there, automatically validating any other keys that have been signed by designated introducers. Since your public key is used as final authority to, directly, or indirectly certify all the other keys on your key ring, it is the most important key to protect from tampering. To detect any tampering of the ultimately trusted public key, PGP can be set up to automatically compare the public key against a backup copy on write-protected media.

PGP generally assumes the user will maintain physical security over the system and the key rings, as well as the copy of PGP itself. If an intruder can tamper with the disk, then in theory he can tamper with PGP itself, rendering moot the safeguards PGP may have to detect tampering with keys.

---

[1] Internally, the keys are referenced by key ID. Key ID is derived from public key.

Keys that have been certified by a trusted introducer are deemed valid by PGP. The keys belonging to trusted introducers must themselves be certified either by you or by other trusted introducers. PGP also allows for the possibility of the user having several levels of trust for people to act as introducers. You can designate a person to PGP as unknown, untrusted, marginally trusted, or completely trusted to certify other public key. This trust information is stored on your key ring, but PGP will not copy the trust information along with the key, because your private opinions on trust are regarded as confidential.

This unique, grass-root approach contrasts sharply with standard public key management schemes, such as Internet Privacy Enhanced Mail (PEM), which is based on centralized control and mandatory centralized trust. PGP's decentralized probabilistic method for determining public key legitimacy is the centerpiece of its key management architecture.
The decentralized non-institutional approach PGP uses to manage public keys has its benefits, but unfortunately, this also means PGP cannot rely on a single centralized list of which keys have been compromised. This makes it a bit harder to contain the damage of a secret key compromise. The only way is to spread the word and hope everyone hears it. If the worst case happens - the secret key and pass-phrase are both compromised (hopefully it can be found out somehow) - a "key compromise" certificate has to be issued. Then this compromise certificate must somehow be sent to everyone else. Their own PGP software will install this key compromise certificate on their public key ring and will automatically prevent them from accidentally using the compromised public key ever again. You may use the same mechanism to revoke the key for some other reasons than the compromise of a secret key.

If you lose the secret key, or if the secret key is destroyed, you cannot revoke it, because the secret key must be used to do it, and the user does not have it anymore. A future version of PGP will offer a more secure means of revoking keys in these circumstances, allowing trusted introducers to certify that a public key has been revoked. But for now, you will have to get the word through whatever informal means, asking users to "disable" the user's public key on their own individual public key rings.

Although Zimmermann was the author of PGP version 1.0, major parts of later versions were implemented by an international collaborative effort, under Zimmermann's design guidance. The development of PGP has turned into a remarkable social phenomenon, whose unique political appeal has inspired the collective efforts of an ever-growing number of volunteer programmers.

PGP does not have any glaring weaknesses. The crypto algorithms were developed by people at high levels of civilian academia and have been individually subject to extensive peer review. Source code is available to facilitate peer review of PGP and to help dispel the fears of some users. It is reasonably well researched and has been years in the making.

Potential vulnerabilities of PGP, that you should be aware of, include compromising the pass-phrase or secret key, public key tampering, deleted files that are still somewhere on the disk, viruses and Trojan horses, breaches in the physical security, electromagnetic emissions, exposure on multi-user systems, traffic analysis and perhaps even direct cryptoanalysis. A somewhat obscure vulnerability of PGP involves dishonest users creating bogus timestamps on their own public key certificates and signatures. This may have some legal or financial benefit to the user, by creating some kind of loophole that might allow the repudiation of signature. A remedy for this could involve some trustworthy Certifying Authority or notary that would create notarized signatures with a trustworthy timestamp. This might not necessarily require a centralized authority. The notary would sign the public key certificate and the trusted timestamp in the notary's signature would have some legal significance. This certificate could be kept in a special certificate log, controlled by the notary. This log could be available to anyone to read.

PGP is not designed to protect the data while it is in plaintext form on a compromised system. Nor can it prevent an intruder from using sophisticated measures to read the secret key while it is being used. You have to recognize these risks on multi-user systems, and adjust the expectations and behavior accordingly.

In summary, without good cryptographic protection of the data communications, it may have been practically effortless and perhaps even routine for an opponent to intercept the messages, especially those sent through a modem or e-mail system. If PGP is used and reasonable precautions are followed, the attacker will have to expend far more effort and expense to violate the privacy. If you protect yourself against the simplest attacks, and you feel confident that your privacy is not going to be violated by a determined and highly resourceful attacker, then you will probably be safe using PGP.