34

Network Security and Secure Applications

34.1 Introduction 34-1
34.2 Security Attacks and Security Properties 34-2
34.3 Security Mechanisms 34-3
Attack Prevention • Attack Avoidance • Secret Key Cryptography • Public Key Cryptography • Authentication and Digital Signatures • Attack and Intrusion Detection
34.4 Secure Network Protocols 34-9
34.5 Secure Applications 34-11
34.6 Summary 34-12

Christopher Kruegel
University of California

34.1 Introduction

In order to provide useful services or to allow people to perform tasks more conveniently, computer systems are attached to networks and get interconnected. This resulted in the worldwide collection of local and wide-area networks known as the Internet. Unfortunately, the extended access possibilities also entail increased security risks as it opens additional avenues for an attacker. For a closed, local system, the attacker was required to be physically present at the network in order to perform unauthorized actions. In the networked case, each host that can send packets to the victim can be potentially utilized. As certain services (such as web or name servers) need to be publicly available, each machine on the Internet might be the originator of malicious activity. This fact makes attacks very likely to happen on a regularly basis.

The following text attempts to give a systematic overview of security requirements of Internet-based systems and potential means to satisfy them. We define properties of a secure system and provide a classification of potential threats to them. We also introduce mechanisms to defend against attacks that attempt to violate desired properties. The most widely used means to secure application data against tampering and eavesdropping, the Secure Sockets Layer (SSL) and its successor, the Transport Layer Security (TLS) protocol, are discussed. Finally, we briefly describe popular application programs that can act as building blocks for securing custom applications.

Before one can evaluate attacks against a system and decide on appropriate mechanisms against them, it is necessary to specify a *security policy* [21]. A security policy defines the desired properties for each part of a secure computer system. It is a decision that has to take into account the value of the assets that should be protected, the expected threats, and the cost of proper protection mechanisms. A security policy that is sufficient for the data of a normal user at home may not be sufficient for bank applications, as these systems are obviously a more likely target and have to protect more valuable resources. Although

0-8493-1985-4/05/$0.00+$1.50

often neglected, the formulation of an adequate security policy is a prerequisite before one can identify threats and appropriate mechanisms to tackle them.

34.2 Security Attacks and Security Properties

For the following discussion, we assume that the function of a system that is the target of an attack is to provide information. In general, there is a flow of data from a source (e.g., host, file, memory) to a destination (e.g., remote host, other file, user) over a communication channel (e.g., wire, data bus). The task of the security system is to restrict access to this information to only those parties (persons or processes) that are authorized to have access according to the security policy in use. In the case of an automation system that is remotely connected to the Internet, the information flow is from/to a control application that manages sensors and actuators via communication lines of the public Internet and the network of the automation system (e.g., a field-bus).

The normal information flow and several categories of attacks that target it are shown in Figure 34.1 and explained below (according to [20]).

1. *Interruption*: An asset of the system gets destroyed or becomes unavailable. This attack targets the source or the communication channel and prevents information from reaching its intended target (e.g., cut the wire, overload the link so that the information gets dropped because of congestion). Attacks in this category attempt to perform a kind of *denial-of-service(DOS).*
2. *Interception*: An unauthorized party gains access to the information by eavesdropping into the communication channel (e.g., wiretapping).
3. *Modification*: The information is not only intercepted, but modified by an unauthorized party while in transit from the source to the destination. By tampering with the information, it is actively altered (e.g., modifying message content).
4. *Fabrication*: An attacker inserts counterfeit objects into the system without having the sender doing anything. When a previously intercepted object is inserted, this processes is called *replaying.* When the attacker pretends to be the legitimate source and inserts his desired information, the attack is called *masquerading* (e.g., replay an authentication message, add records to a file).

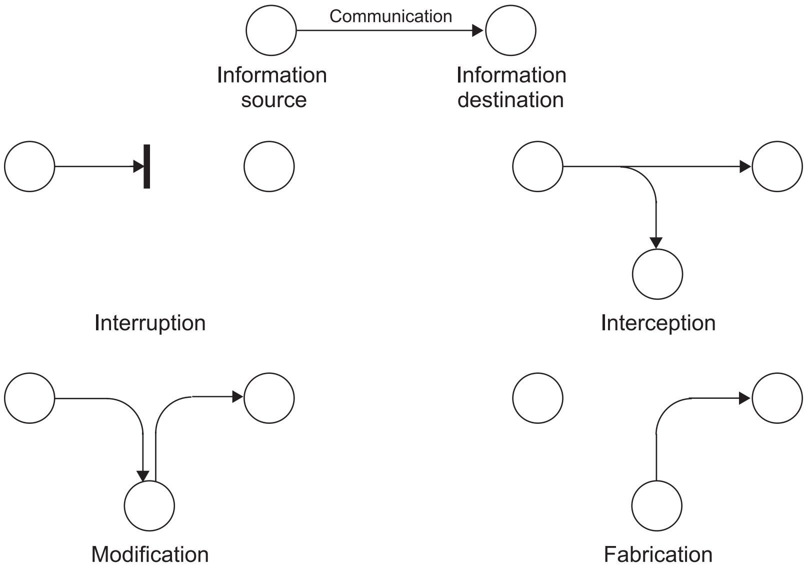

FIGURE 34.1 Security attacks.

The four classes of attacks listed above violate different security properties of the computer system. A security property describes a desired feature of a system with regard to a certain type of attack. A common classification following [5, 11] is listed below.

Confidentiality: This property covers the protection of transmitted data against its release to nonauthorized parties. In addition to the protection of the content itself, the information flow should also be resistant against traffic analysis. Traffic analysis is used to gather information other than the transmitted values themselves from the data flow (e.g., timing data, frequency of messages).

Authentication: Authentication is concerned with making sure that the information is authentic. A system implementing the authentication property assures the recipient that the data are from the source that it claims to be. The system must make sure that no third party can masquerade successfully as another source.

Nonrepudiation: This property describes the feature that prevents either the sender or receiver from denying a transmitted message. When a message has been transferred, the sender can prove that it has been received. Similarly, the receiver can prove that the message has actually been sent.

Availability: Availability characterizes a system whose resources are always ready to be used. Whenever information needs to be transmitted, the communication channel is available and the receiver can cope with the incoming data. This property makes sure that attacks cannot prevent resources from being used for their intended purpose.

Integrity: Integrity protects transmitted information against modifications. This property assures that a single message reaches the receiver as it has left the sender, but integrity also extends to a stream of messages. It means that no messages are lost, duplicated, or reordered and it makes sure that messages cannot be replayed. As destruction is also covered under this property, all data must arrive at the receiver. Integrity is not only important as a security property but also as a property for network protocols. Message integrity must also be ensured in case of random faults, and not just in the case of malicious modifications.

34.3 Security Mechanisms

Different security mechanisms can be used to enforce the security properties defined in a given security policy. Depending on the anticipated attacks, different means have to be applied to satisfy the desired properties. We divide these measures against attacks into three different classes: attack prevention, attack avoidance, and attack detection.

Attack Prevention

Attack prevention is a class of security mechanisms that contains ways of preventing or defending against certain attacks before they can actually reach and affect the target. An important element in this category is access control, a mechanism that can be applied at different levels such as the operating system, the network, or the application layer.

Access control [21] limits and regulates the access to critical resources. This is done by identifying or authenticating the party that requests a resource and checking its permissions against the rights specified for the demanded object. It is assumed that an attacker is not legitimately permitted to use the target object and is therefore denied access to the resource. As access is a prerequisite for an attack, any possible interference is prevented.

The most common form of access control used in multi-user computer systems are access control lists for resources that are based on the user identity of the process that attempts to use them. The identity of a user is determined by an initial authentication process that usually requires a name and a password. The login process retrieves the stored copy of the password corresponding to the user name and compares it with the presented one. When both match, the system grants the user the appropriate user credentials. When a resource should be accessed, the system looks up the user and group in the access control list and grants or denies access as appropriate. An example of this kind of access control is a secure web server. A secure web server delivers certain resources only to clients that have authenticated themselves and that possess sufficient credentials for the desired resource. The authentication process is usually handled by the web client such as the `Microsoft Internet Explorer`, or `Mozilla` by prompting the user for his name and password.

The most important access control system at the network layer is a firewall [4]. The idea of a firewall is based on the separation of a trusted inside network of computers under single administrative control from a potential hostile outside network. The firewall is a central choke point that allows enforcement of access control for services that may run at the inside or outside. The firewall prevents attacks from the outside against the machines in the inside network by denying connection attempts from unauthorized parties located outside. In addition, a firewall may also be utilized to prevent users behind the firewall from using certain services that are outside (e.g., surfing web sites containing pornographic material).

For certain installations, a single firewall is not suitable. Networks that consist of several server machines that need to be publicly accessible and workstations that should be completely protected against connections from the outside would benefit from a separation between these two groups. When an attacker compromises a server machine behind a single firewall, all other machines can be attacked from this new base without restrictions. To prevent this, one can use two firewalls and the concept of a *demilitarized zone (DMZ)* [4] in between as shown in Figure 34.2.

In this setup, one firewall separates the outside network from a segment (DMZ) with the server machines, while a second one separates this area from the rest of the network. The second firewall can be configured in a way that denies all incoming connection attempts. Whenever an intruder compromises a server, he is now unable to immediately attack a workstation located in the inside network.

The following design goals for firewalls are identified in [4]:

1. All traffic from inside to outside, and *vice versa,* must pass through the firewall. This is achieved by physically blocking all access to the internal network except via the firewall.
2. Only authorized traffic, as defined by the local security policy, will be allowed to pass.
3. The firewall itself should be immune to penetration. This implies the use of a trusted system with a secure operating system. A trusted, secure operating system is often purpose-built, has heightened security features, and only provides the minimal functionality necessary to run the desired applications.

These goals can be reached by using a number of general techniques for controlling access. The most common is called *service control* and determines Internet services that can be accessed. Traffic on the Internet is currently filtered on the basis of IP addresses and TCP/UDP port numbers. In addition, there may be proxy software that receives and interprets each service request before passing it on. *Direction control* is a simple mechanism to control the direction in which particular service requests may be initiated and permitted to flow through. *User control* grants access to a service based on user credentials similar to the technique used in a multiuser operating system. Controlling external users requires secure authentication over the network (e.g., such as provided in IPSec [9]). A more declarative approach in contrast to the operational variants mentioned above is *behavior control.* This technique determines how particular services are used. It may be utilized to filter e-mail to eliminate spam or to allow external access to only part of the local web pages.

A summary of capabilities and limitations of firewalls is given in [20]. The following benefits can be expected:

- A firewall defines a single choke point that keeps unauthorized users out of the protected network. The use of such a point also simplifies security management.
- It provides a location for monitoring security-related events. Audits, logs, and alarms can be implemented on the firewall directly. In addition, it forms a convenient platform for some nonsecurity-related functions such as address translation and network management.

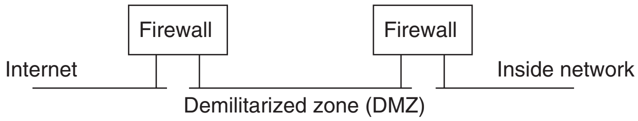

FIGURE 34.2 Demilitarized zone.

- A firewall may serve as a platform to implement a virtual private network (e.g., by using IPSec).

The list below enumerates the limits of the firewall access control mechanism.

- A firewall cannot protect against attacks that bypass it, for example, via a direct dial-up link from the protected network to an Internet Service Provider (ISP). It also does not protect against internal threats from an inside hacker or an insider cooperating with an outside attacker.
- A firewall does not help when attacks are against targets whose access has to be permitted.
- It cannot protect against the transfer of virus-infected programs or files. It would be impossible, in practice, for the firewall to scan all incoming files and e-mails for viruses.

Firewalls can be divided into two main categories. A *Packet-Filtering Router,* or short packet filter, is an extended router that applies certain rules to the packets that are forwarded. Usually, traffic in each direction (in- and outgoing) is checked against a rule set, which determines whether a packet is permitted to continue or should be dropped. The packet filter rules operate on the header fields used by the underlying communication protocols, for the Internet almost always IP, TCP, and UDP. Packet filters have the advantage that they are cheap as they can often be built on existing hardware. In addition, they offer a good performance for high traffic loads. An example for a packet filter is the `iptables` package, which is implemented as part of the `Linux` 2.4 routing software.

A different approach is followed by an *Application-Level Gateway,* also called proxy server. This type of firewall does not forward packets on the network layer but acts as a relay on the application level. The user contacts the gateway, which in turn opens a connection to the intended target (on behalf of the user). A gateway completely separates the inside and outside networks at the network level and only provides a certain set of application services. This allows authentication of the user who requests a connection and session-oriented scanning of the exchanged traffic up to the application-level data. This feature makes application gateways more secure than packet filters and offers a broader range of log facilities. On the downside, the overhead of such a setup may cause performance problems under heavy load.

Attack Avoidance

Security mechanisms in this category assume that an intruder may access the desired resource but the information is modified in a way that makes it unusable for the attacker. The information is preprocessed at the sender before it is transmitted over the communication channel and postprocessed at the receiver. While the information is transported over the communication channel, it resists attacks by being nearly useless for an intruder. One notable exception are attacks against the availability of the information as an attacker could still interrupt the message. During the processing step at the receiver, modifications or errors that might have previously occurred can be detected (usually because the information cannot be correctly reconstructed). When no modification has taken place, the information at the receiver is identical to the one at the sender before the preprocessing step.

The most important member in this category is cryptography, which is defined as the science of keeping messages secure [16]. It allows the sender to transform information into a random data stream from the point of view of an attacker but to have it recovered by an authorized receiver (see Figure 34.3).

The original message is called *plain text* (sometimes clear text). The process of converting it through the application of some transformation rules into a format that hides its substance is called *encryption.* The corresponding disguised message is denoted *cipher text* and the operation of turning it back into clear text is called *decryption.* It is important to notice that the conversion from plain to cipher text has to be loss-less in order to be able to recover the original message at the receiver under all circumstances.

The transformation rules are described by a cryptographic algorithm. The function of this algorithm is based on two main principles: *substitution* and *transposition.* In the case of substitution, each element of the plain text (e.g., bit, block) is mapped into another element of the used alphabet. Transposition describes the process where elements of the plain text are rearranged. Most systems involve multiple steps (called rounds) of transposition and substitution to be more resistant against cryptanalysis. Cryptanalysis is the science of breaking the cipher, that is, discovering the substance of the message behind its disguise.

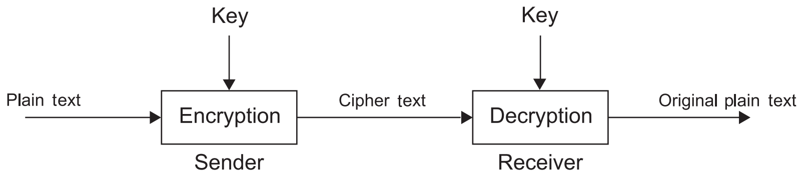

FIGURE 34.3 Encryption and decryption.

When the transformation rules process the input elements one at a time, the mechanism is called a *stream cipher;* in case of operating on fixed-sized input blocks, it is called a *block cipher.*

If the security of an algorithm is based on keeping the way how the algorithm works (i.e., the transformation rules) secret, it is called a restricted algorithm. Those algorithms are no longer of any interest today because they do not allow standardization or public quality control. In addition, when a large group of users is involved, such an approach cannot be used. A single person leaving the group makes it necessary for everyone else to change the algorithm.

Modern cryptosystems solve this problem by basing the ability of the receiver to recover encrypted information on the fact that he possesses a secret piece of information (usually called the *key).* Both encryption and decryption functions have to use a key and they are heavily dependent on it. When the security of the cryptosystem is completely based on the security of the key, the algorithm itself may be revealed. Although the security does not rely on the fact that the algorithm is unknown, the cryptographic function itself and the used key together with its length must be chosen with care. A common assumption is that the attacker has the fastest commercially available hardware at his disposal in his attempt to break the cipher text.

The most common attack, called *known plain text attack,* is executed by obtaining cipher text together with its corresponding plain text. The encryption algorithm must be so complex that even if the code breaker is equipped with plenty of such pairs and powerful machines, it is infeasible for him to retrieve the key. An attack is infeasible when the cost of breaking the cipher exceeds the value of the information or the time it takes to break it exceeds the lifespan of the information.

Given pairs of corresponding cipher and plain text, it is obvious that a simple key guessing algorithm will succeed after some time. The approach of successively trying different key values until the correct one is found is called *brute force* attack because no information about the algorithm is utilized whatsoever. In order to be useful, it is a necessary condition for an encryption algorithm that brute force attacks are infeasible.

Depending on the keys that are used, one can distinguish two major cryptographic approaches — public and secret key cryptosystems.

Secret Key Cryptography

This is the kind of cryptography that has been used for the transmission of secret information for centuries, long before the advent of computers. These algorithms require that the sender and the receiver agree on a key before communication is started.

It is common for this variant (which is also called single key or symmetric encryption) that a single secret key is shared between the sender and the receiver. It needs to be communicated in a secure way before the actual encrypted communication can start and has to remain secret as long as the information is to remain secret. Encryption is achieved by applying an agreed function to the plain text using the secret key. Decryption is performed by applying the inverse function using the same key.

The classic example of a secret key block cipher that is widely deployed today is the Data Encryption Standard (DES) [6]. DES was developed in 1977 by IBM and was adopted as a standard by the US government for administrative and business use. Recently, it has been replaced by the Advanced Encryption Standard (AES — Rijndael) [1]. It is a block cipher that operates on 64-bit plain text blocks and utilizes a key with 56-bits length. The algorithm uses 16 rounds that are key dependent. During each round, 48 key bits are selected and combined with the block that is encrypted. Then, the resulting block is piped through a substitution and a permutation phase (which use known values and are independent of the key) to make cryptanalysis harder.

Although there is no known weakness of the DES algorithm itself, its security has been much debated. The small key length makes brute force attacks possible and several cases have occurred where DES-protected information has been cracked. A suggested improvement called 3DES uses three rounds of the simple DES with three different keys. This extends the key length to 168 bits while still resting on the very secure DES base.

A well-known stream cipher that has been debated recently is RC4 [14], which has been developed by RSA. It is used to secure the transmission in wireless networks that follow the IEEE 802.11 standard and forms the core of the WEP (wired equivalent protection) mechanism. Although the cipher itself has not been broken, current implementations are flawed and reduce the security of RC4 down to a level where the used key can be recovered by statistical analysis within a few hours.

Public Key Cryptography

Since the advent of public key cryptography, the knowledge of the key that is used to encrypt a plain text also allowed the inverse process, the decryption of the cipher text. In 1976, this paradigm of cryptography was changed by Diffie and Hellman [7] when they described their public key approach. Public key cryptography utilizes two different keys, one called the *public key,* and the other called the *private key.* The public key is used to encrypt a message, while the corresponding private key is used to do the opposite. Their innovation was the fact that it is infeasible to retrieve the private key given the public key. This makes it possible to remove the weakness of secure key transmission from the sender to the receiver. The receiver can simply generate his public/private key pair and announce the public key without fear. Anyone can obtain this key and use it to encrypt messages that only the receiver with his private key is able to decrypt.

Mathematically, the process is based on the *trap door* of *one-way* functions. A one-way function is a function that is easy to compute but very hard to inverse. This means that given *x,* it is easy to determine *f(x)* but given *f(x)* it is hard to obtain *x.* Hard is defined as computationally infeasible in the context of cryptographically strong one-way functions. Although it is obvious that some functions are easier to compute than their inverse (e.g., square of a value in contrast to its square root), there is no mathematical proof or definition of one-way functions. There are a number of problems that are considered difficult enough to act as one-way functions but it is more an agreement among crypto analysts than a rigorously defined set (e.g., factorization of large numbers). A one-way function is not directly usable for cryptography, but it becomes so when a trap door exists. A trap door is a mechanism that allows one to easily calculate *x* from *f(x)* when an additional information *y* is provided.

A common misunderstanding about public key cryptography is thinking that it makes secret key systems obsolete, either because it is more secure or because it does not have the problem of secretly exchanging keys. As the security of a cryptosystem depends on the length of the used key and the utilized transformation rules, there is no automatic advantage of one approach over the other. Although the key exchange problem is elegantly solved with a public key, the process itself is very slow and has its own problems. Secret key systems are usually a factor of 1000 (see [16] for exact numbers) faster than their public key counterparts. Therefore, most communication is stilled secured using secret key systems and public key systems are only utilized for exchanging the secret key for later communication. This hybrid approach is the common design to benefit from the high speed of conventional cryptography (which is often implemented directly in hardware) and from a secure key exchange.

A problem in public key systems is the authenticity of the public key. An attacker may offer the sender his own public key and pretend that it origins from the legitimate receiver. The sender then uses the faked public key to perform his encryption and the attacker can simply decrypt the message using his private key. In order to thwart an attacker that attempts to substitute his public key for the victim's one, *certificates* are used. A certificate combines user information with the user's public key and the digital signature of a trusted third party that guarantees that the key belongs to the mentioned person. The trusted third party is usually called a *certification authority (CA).* The certificate of a CA itself is usually verified by a higher-level CA that confirms that the CA's certificate is genuine and contains its public key. The chain of third parties that verify their respective lower-level CAs has to end at a certain point, which is called the root CA. A user that wants to verify the authenticity of a public key and all involved CAs needs to obtain the self-signed certificate of the root CA via an external channel. Web browsers (e.g., `Netscape Navigator, Internet Explorer`) usually ship with a number of certificates of globally known root CAs. A framework that implements the distribution of certificates is called a public key infrastructure (PKI). An important protocol for key management is X.509 [23]. Another important issue is revocation, the invalidation of a certificate when the key has been compromised.

The best-known public key algorithm and textbook classic is RSA [15], named after its inventors Rivest, Shamir, and Adleman at MIT. It is a block cipher that is still utilized for the majority of current systems, although the key length has been increased over recent years. This has put a heavier processing load on applications, a burden that has ramifications especially for sites doing electronic commerce. A competitive approach that promises similar security as RSA using far smaller key lengths is elliptic curve cryptography. However, as these systems are new and have not been subject to sustained cryptanalysis, the confidence level in them is not yet as high as in RSA.

Authentication and Digital Signatures

An interesting and important feature of public key cryptography is its possible use for authentication. In addition to making the information unusable for attackers, a sender may utilize cryptography to prove his identity to the receiver. This feature is realized by *digital signatures.* A digital signature must have similar properties as a normal handwritten signature. It should be hard to forge and it has to be bound to a certain document. In addition, one has to make sure that a valid signature cannot be used by an attacker to replay the same (or different) messages at a later time.

A way to realize such a digital signature is by using the sender's private key to encrypt a message. When the receiver is capable of successfully decrypting the cipher text with the sender's public key, he can be sure that the message is authentic. This approach obviously requires a cryptosystem that allows encryption with the private key, but many (such as RSA) offer this option. It is easy for a receiver to verify that a message has been successfully decrypted when the plain text is in a human readable format. For binary data, a checksum or similar integrity checking footer can be added to verify a successful decryption. Replay attacks are prevented by adding a time-stamp to the message (e.g., `Kerberos` [10] uses time-stamps to prevent that messages to the ticket granting service are replayed).

Usually, the storage and processing overhead for encrypting a whole document is too high to be practical. This is solved by one-way hash functions. These are functions that map the content of a message onto a short value (called *message digest).* Similar to one-way functions, it is difficult to create a message when given only the hash value itself. Instead of encrypting the whole message, it is enough to simply encrypt the message digest and send it together with the original message. The receiver can then apply the known hash function (e.g., MD5 [13]) to the document and compare it to the decrypted digest. When both values match, the messages is authentic.

Attack and Intrusion Detection

Attack detection assumes that an attacker can obtain access to his desired targets and is successful in violating a given security policy. Mechanisms in this class are based on the optimistic assumption that most of the time the information is transferred without interference. When undesired actions occur, attack detection has the task of reporting that something went wrong and then to react in an appropriate way. In addition, it is often desirable to identify the exact type of attack. An important facet of attack detection is recovery. Often, it is enough to just report that malicious activity has been found, but some systems require that the effect of the attack has to be reverted or that an ongoing and discovered attack is stopped. On the one hand, attack detection has the advantage that it operates under the worst-case assumption that the attacker gains access to the communication channel and is able to use or modify the resource. On the other hand, detection is not effective in providing confidentiality of information. When the security policy specifies that interception of information has a serious security impact, then attack detection is not an applicable mechanism. The most important members of the attack detection class, which have received an increasing amount of attention in the last few years, are intrusion detection systems (aka IDS).

Intrusion Detection [2, 3] is the process of identifying and responding to malicious activities targeted at computing and network resources. This definition introduces the notion of intrusion detection as a process, which involves technology, people, and tools. An IDS basically monitors and collects data from a target system that should be protected, processes and correlates the gathered information, and initiate responses, when evidence for an intrusion is detected. IDS are traditionally classified as anomaly- or signature-based. Signature-based systems act similar to virus scanners and look for known, suspicious patterns in their input data. Anomaly-based systems watch for deviations of actual from expected behavior and classify all "abnormal" activities as malicious.

The advantage of signature-based designs is the fact that they can identify attacks with an acceptable accuracy and tend to produce fewer false alarms (i.e., classifying an action as malicious when in fact it is not) than their anomaly-based cousins. The systems are more intuitive to build and easier to install and configure, especially in large production networks. Because of this, nearly all commercial systems and most deployed installations utilize signature-based detection. Although anomaly-based variants offer the advantage of being able to find prior unknown intrusions, the costs of having to deal with an order of magnitude more false alarms is often prohibitive.

Depending on their source of input data, IDS can be classified as either network- or host-based. Network-based systems collect data from network traffic (e.g., packets by network interfaces in promiscuous mode), while host-based systems monitor events at the operating system level such as system calls or receive input from applications (e.g., via log files). Host-based designs can collect high-quality data directly from the affected system and are not influenced by encrypted network traffic. Nevertheless, they often seriously impact the performance of the machines they are running on. Network-based IDS, on the other hand, can be set up in a nonintrusive manner — often as an appliance box without interfering with the existing infrastructure. In many cases, this makes them the preferred choice.

As many vendors and research centers have developed their own intrusion detection system versions, the IETF has created the intrusion detection working group [8] to coordinate international standardization efforts. The aim is to allow intrusion detection systems to share information and to communicate via well-defined interfaces by proposing a generic architectural description and a message specification and exchange format (IDMEF).

34.4 Secure Network Protocols

After the general concepts and mechanisms of network security have been introduced, the following section concentrates on two actual instances of secure network protocols: the SSL [18] and the TLS [22] protocol.

The idea of secure network protocols is to create an additional layer between the application and the transport/network layer to provide services for a secure end-to-end communication channel.

TCP/IP are almost always used as transport/network layer protocols on the Internet and their task is to provide a reliable end-to-end connection between remote tasks on different machines that intend to communicate. The services on that level are usually directly utilized by application protocols to exchange data, for example, Hypertext Transfer Protocol (HTTP) for web services. Unfortunately, the network layer transmits these data unencrypted, leaving it vulnerable to eavesdropping or tampering attacks. In addition, the authentication mechanisms of TCP/IP are only minimal, thereby allowing a malicious user to hijack connections and redirect traffic to his machine as well as to impersonate legitimate services. These threats are mitigated by secure network protocols that provide privacy and data integrity between two communicating applications by creating an encrypted and authenticated channel.

SSL has emerged as the de-facto standard for secure network protocols. Originally developed by Netscape, its latest version SSL 3.0 is also the base for the standard proposed by the IETF under the name TLS. Both protocols are quite similar and share common ideas, but they unfortunately cannot interoperate. The following discussion will mainly concentrate on SSL and only briefly explain the extensions implemented in TLS.

The SSL protocol [19] usually runs above TCP/IP (although it could use any transport protocol) and below higher-level protocols such as HTTP. It uses TCP/IP on behalf of the higher-level protocols, and in the process allows an SSL-enabled server to authenticate itself to an SSL-enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection. These capabilities address fundamental concerns about communication over the Internet and other TCP/IP networks and give protection against message tampering, eavesdropping, and spoofing.

SSL server authentication allows a user to confirm a server's identity. SSL-enabled client software can use standard techniques of public-key cryptography to check that a server's certificate and public key are valid and have been issued by a CA listed in the client's list of trusted CAs. This confirmation might be important if the user, for example, is sending a credit card number over the network and wants to check the receiving server's identity.

SSL client authentication allows a server to confirm a user's identity. Using the same techniques as those used for server authentication, SSL-enabled server software can check that a client's certificate and public key are valid and have been issued by a CA listed in the server's list of trusted CAs. This confirmation might be important if the server, for example, is a bank sending confidential financial information to a customer and wants to verify the recipient's identity.

An encrypted SSL connection requires all information sent between a client and a server to be encrypted by the sending software and decrypted by the receiving software, thus providing a high degree of confidentiality. Confidentiality is important for both parties to any private transaction. In addition, all data sent over an encrypted SSL connection are protected by a mechanism for detecting tampering — that is, for automatically determining whether the data have been altered in transit.

SSL uses X.509 certificates for authentication, RSA as its public-key cipher, and one of RC4-128, RC2-128, DES, Triple DES, or IDEA as its bulk symmetric cipher. The SSL protocol includes two subprotocols: the SSL Record Protocol and the SSL Handshake Protocol. The SSL Record Protocol simply defines the format used to transmit data. The SSL Handshake Protocol (using the SSL Record Protocol) is utilized to exchange a series of messages between an SSL-enabled server and an SSL-enabled client when they first establish an SSL connection. This exchange of messages is designed to facilitate the following actions.

- Authenticate the server to the client.
- Allow the client and server to select the cryptographic algorithms, or ciphers, that they both support.
- Optionally authenticate the client to the server.
- Use public-key encryption techniques to generate shared secrets.
- Establish an encrypted SSL connection based on the previously exchanged shared secret.

The SSL Handshake Protocol is composed of two phases. Phase 1 deals with the selection of a cipher, the exchange of a secret key, and the authentication of the server. Phase 2 handles client authentication, if requested and finishes the handshaking. After the handshake stage is complete, the data transfer between client and server begins. All messages during handshaking and after, are sent over the SSL Record Protocol layer. Optionally, session identifiers can be used to reestablish a secure connection that has been previously set up.

Figure 34.4 lists in a slightly simplified form the messages that are exchanged between the client C and the server S during a handshake when neither client authentication nor session identifiers are involved. In this figure, {`data`} key means that data have been encrypted with key.

The message exchanges shows that the client first sends a challenge to the server, which responds with an X.509 certificate containing its public key. The client then creates a secret key and uses RSA with the server's public key to encrypt it, sending the result back to the server. Only the server is capable of decrypting that message with its private key and can retrieve the shared, secret key. In order to prove to the client that the secret key has been successfully decrypted, the server encrypts the client's challenge with the secret key and returns it. When the client is able to decrypt this message and successfully retrieves

Message type	Direction	Data transferred
client-hello	C > S	challenge-data and supported ciphers
server-hello	C < S	server-certificate and ciphers both support
client-master-key	C > S	chosen cipher and {secret-key}server-public-key
server-verify	C < S	{challenge-data}secret-key

FIGURE 34.4 SSL handshake message exchange.

the original challenge by using the secret key, it can be certain that the server has access to the private key corresponding to its certificate. From this point on, all communication is encrypted using the chosen cipher and the shared secret key.

TLS uses the same two protocols shown above and a similar handshake mechanism. Nevertheless, the algorithms for calculating message authentication codes (MACs) and secret keys have been modified to make them cryptographically more secure. In addition, the constraints on padding a message up to the next block size have been relaxed for TLS. This leads to an incompatibility between both protocols.

SSL/TLS is widely used to secure web and mail traffic. HTTP as well as the current mail protocols IMAP (Internet Message Access Protocol) and POP3 (post office protocol, version 3) transmit user credential information as well as application data unencrypted. By building them on top of a secure network protocol such as SSL/TLS, they can benefit from secured channels without modifications. The secure communication protocols simply utilize different well-known destination ports (443 for HTTPS, 993 for IMAPS, and 995 for POP3S) than their insecure cousins.

34.5 Secure Applications

A variety of popular tools that allow access to remote hosts (such as telnet, rsh, and rlogin) or that provide means for file transfer (such as rcp or ftp) exchange user credentials and data in plain text. This makes them vulnerable to eavesdropping, tampering, and spoofing attacks. Although the tools mentioned above could have also been built upon SSL/TLS, a different protocol suite called Secure Shell (SSH) [17] has been developed, which follows partial overlapping goals. The SSH Transport and User Authentication protocols have features similar to those of SSL/TLS. However, they are different in the following ways:

- TLS server authentication is optional and the protocol supports fully anonymous operation, in which neither side is authenticated. As such connections are inherently vulnerable to man-in-the-middle attacks, SSH requires server authentication.
- TLS does not provide the range of client authentication options that SSH does — public-key via RSA is the only option.
- Most importantly, TLS does not have the extra features provided by the SSH Connection Protocol.

The SSH Connection Protocol uses the underlying connection, aka secure tunnel, which has been established by the SSH Transport and User Authentication protocols between two hosts. It provides interactive login sessions, remote execution of commands, and forwarded TCP/IP as well as XI1 connections. All these terminal sessions and forwarded connections are realized as different logical channels that may be opened by either side on top of the secure tunnel. Channels are flow-controlled, which means that no data may be sent to a channel until a message is received to indicate that window space is available.

The current version of the SSH protocol is SSH 2. It represents a complete rewrite of SSH 1 and improves some of its structural weaknesses. As it encrypts packets in a different way and has abandoned the notion of server and host keys in favor of host keys only, the protocols are incompatible. For applications built from scratch, SSH 2 should always be the preferred choice.

Using the means of logical channels for interactive login sessions and remote execution, a complete replacement for telnet, rsh, and rlogin could be easily implemented. A popular site that lists open-source implementations, which are freely available for many different platforms, can be found under [12]. Recently, a secure file transfer (sftp) application has been developed that makes the use of regular FTP-based programs obsolete.

Notice that it is possible to tunnel arbitrary application traffic over a connection that has been previously set up by the SSH protocols. Similar to SSL/TLS, web and mail traffic could be securely transmitted over an SSH connection before reaching the server port at the destination host. The difference is that SSH requires that a secure tunnel is created in advance, which is bound to a certain port at the destination host. The setup of this secure channel, however, requires that the client that is initiating the connection has to log into the server. Usually, this makes it necessary that the user has an account at the destination host. After the tunnel has been established, all traffic sent into by the client gets forwarded to the desired port at the target machine. Obviously, the connection is encrypted. In contrast to this SSL/TLS connects directly to a certain point without prior logging into the destination host. The encryption is set up directly between the client and the service listening at the destination port without a prior redirection via the SSH server.

The technique of tunneling application traffic is often utilized for mail transactions when the mail server does not support SSL/TLS directly (as users have accounts at the mail server anyway), but it is less common for web traffic.

34.6 Summary

This chapter discusses security threats that systems face when they are connected to the Internet.

In order to achieve the security properties that are required by the security policy in use, three different classes of mechanisms can be adopted. The first is attack prevention, which attempts to stop the attacker before it can reach its desired goals. Such techniques fall into the category of access control and firewalls. The second approach aims to make the data unusable for unauthorized persons by applying cryptographic means. Secret key as well as public keys mechanism can be utilized. The third class of mechanisms contains attack detection approaches. They attempt to detect malicious behavior and recover after undesired activity has been identified.

The text also covers secure network protocols and applications. SSL/TLS as well as SSH are introduced and its most common fields of operations are highlighted. These protocols form the base of securing traffic that is sent over the Internet on behalf of a variety of different applications.

References

[1] Advanced Encryption Standard (AES), National Institute of Standards and Technology, U.S. Department of Commerce, FIPS 197, 2001.

[2] E. Amoroso, *Intrusion Detection — An Introduction to Internet Surveillance, Correlation, Trace Back, and Response,* Intrusion.Net Books, New Jersey, U.S.A., 1999.

[3] R. Bace, *Intrusion Detection,* Macmillan Technical Publishing, Indianapolis, U.S.A., 2000.

[4] W.R. Cheswick and S. M. Bellovin, *Firewalls and Internet Security,* Addison-Wesley, Reading, MA, U.S.A., 1994.

[5] G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems — Concepts and Design,* 2nd ed., Addison-Wesley, Harlow, England, 1996.

[6] Data Encryption Standard (DES), National Bureau of Standards, U.S. Department of Commerce, FIPS 46-3, 1977.

[7] W. Diffie and M. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory,* IT-22, 644–654, 1976.

[8] Intrusion Detection Working Group, http://www.ietf.org/ids.by.wg/idwg.html.

[9] IP Security Protocol, `http://www.ietf.org/html.charters/ipsec-charter.html,` 2002.

[10] J. Kohl, B. Neuman, and T. 'T'so, The evolution of the Kerberos authentication system, *Distributed Open Systems,* 78 – 94, 1994.

AQ1

[11] S. Northcutt, *Network Intrusion Detection — An Analyst's Handbook,* New Riders, Indianapolis, U.S.A., 1999.

[12] OpenSSH: Free SSH tool suite, `http://www.openssh.org`.

[13] R.L. Rivest, The MD5 message-digest algorithm, Technical report, Internet Request for Comments (RFC) 1321, 1992.

[14] R.L. Rivest, The RC4 encryption algorithm, Technical report, RSA Data Security Inc., 1992.

[15] R.L. Rivest, A. Shamir, and L.A. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM,* 21,120–126, 1978.

[16] B. Schneier, *Applied Cryptography,* 2nd ed., John Wiley & Sons, Inc., New York, U.S.A., 1996.

[17] Secure Shell (secsh), `http://www.ietf.org/html.charters/secsh-charter.html,` 2002.

[18] Secure Socket Layer, `http://wp.netscape.com/eng/ss13/`, 1996.

[19] Introduction to Secure Socket Layer, `http://developer.netscape.com/docs/manuals/security/sslin/contents.htm`, 1996.

[20] W. Stallings, *Network Security Essentials — Applications and Standards,* Prentice-Hall, Englewood Cliffs, NJ, U.S.A., 2000.

[21] A.S. Tanenbaum and M. van Steen, *Distributed Systems — Principles and Paradigms,* Prentice-Hall, Englewood Cliffs, NJ, U.S.A., 2002.

[22] Transport Layer Security, `http://www.ietf.org/html.charters/tsl-charter.html`, 2002.

[23] Public-Key Infrastructure X.509, `http://www.ietf.org/html.charters/pkix-charter.html`, 2002.

Manuscript Queries

Title: The Industrial Information Technology Handbook
CHAPTER 34

Query Number	Query
AQ: 1	Please provide volume no.