4

Cyber Security
Basic Defenses and Attack Trends

Alvaro A. Cárdenas, Tanya Roosta, Gelareh Taban, and Shankar Sastry

4.1 Introduction

Our society, economy, and critical infrastructures have become largely dependent on computer networks and other information technology solutions. As our dependence on information technology increases, cyber attacks become more attractive, and potentially more disastrous.

Cyber attacks are cheaper, more convenient, and less risky than physical attacks: they require few expenses beyond a computer and an Internet connection, they are unconstrained by geography and distance, they are not physically dangerous for the attacker, and it is more difficult to identify and prosecute the culprits of a cyber attack. Furthermore, cyber attacks are easy to replicate. Once a single attacker writes a malicious program, several other people in any part of the world can reuse this program to attack other systems.

Given that attacks against information technology systems are very attractive, and that their numbers and sophistication are expected to keep increasing, we need to have the knowledge and tools for a successful defense.

Cyber security is the branch of security dealing with digital or information technology.[1] This chapter presents a selected overview on topics in cyber security.

[1] Throughout the chapter, we use the terms *security, information security*, and *computer security* interchangeably.

Following the subject of this book, we also explore the role of cyber security as part of the strategies for homeland security. Cyber security is an essential component in the protection of any nation. For example, the Department of Homeland Security (DHS) in the United States has a National Cyber Security Division (NCSD) with the following two objectives: (1) to build and maintain a cyberspace response system, and (2) to implement a cyber-risk management program for the protection of critical infrastructure.

The cyberspace response system unit is in charge of determining the actions that need to be taken when a cyber incident arises. The cyber-risk management program is in charge of assessing risks, prioritizing resources, and building a national awareness program in an effort to build stronger defenses against cyber attacks.

As we explore in Section 4.6.3 (with the recent attacks in Greece and Estonia), similar cyber security response programs can greatly benefit other countries, providing assistance and guidance for responding and handling cyber security incidents.

This chapter is divided into five sections:

- In Section 4.2 we describe the basic goals and terminologies used in information security to provide the necessary background for the subsequent sections.
- In Section 4.3 we give a brief overview of cryptography. Cryptography is an essential tool for securing communication by providing integrity, authentication, and confidentiality.
- In Section 4.4 we turn to network security. We show examples of widely used cryptographic protocols in Internet communication, such as IPsec and SSL. Cryptographic protocols, however, are only one link for achieving Internet security: firewalls are a complementary means for preventing attacks; intrusion detection systems are useful for detecting attacks; and honeypots provide security researchers the ability to study and understand the attacks.
- In Section 4.5, we focus on software security. We first summarize some of the most common problems and then give an overview of the ways security researchers attempt to prevent and limit the problems arising from malicious codes.
- Finally, in Section 4.6, we discuss some of the trends and threats of cyber security, and their relation to homeland security. We first discuss the growing cyber crime and botnet problem. Then, we discuss some of the threats to the communication infrastructure, such as worms and distributed denial of service (DDoS) attacks. We then finalize by discussing

some of the threats that may have a wider impact for homeland security: cyber espionage and attacks to the computer systems supporting and controlling critical infrastructures, such as electric power distribution, or water supply systems.

Cyber security is a very large field, therefore there are several topics that we are not able to cover in this short survey. Some topics missing in our survey include access control, usability, information flow control, security policies, secure operating systems, trusted computing, and advanced topics in cryptography, network security, and software security. We refer the reader interested in details on these topics to one of several computer security books [1–6].

4.2 Basic Concepts

A systematic study of the security of any system requires the description of three concepts: the security goals we want to achieve, the threats we expect to face, and the mechanisms and tools we can use to protect the system.

4.2.1 Common Security Goals

When a system is said to be "secure" it usually means that it has one or more of the following properties [7]:

- *Confidentiality* (or secrecy) refers to the concealment of information or resources from all but those who are authorized. A violation of confidentiality results in *disclosure*: a situation where an unauthorized party gets access to secret information.
- *Integrity* refers to the trustworthiness of data or resources. The goal of integrity is to prevent an attacker from tampering or corrupting the system's data or resources. A violation of integrity results in *deception*: a situation where a legitimate party receives false information and believes it to be true.
- *Availability* refers to the ability to use the information or resource desired. A violation of availability results in *denial of service* (DoS): the prevention (or "noticeable" delay) of authorized access to the information or resource.

Some security policies, such as preventing unauthorized users from using a resource (free-riding), are not directly covered by these three goals. However, most other security properties, such as privacy—the ability of a person to choose which personal details are to be kept confidential—or authentication—the verification of an identity (a subset of integrity)—rely on these three goals.

Integrity and availability are properties that we would like to have in almost every system. Even if we are not faced with a malicious entity, there are systems that can fail, corrupt data, and become unavailable because of design failures or accidents. Under these types of failures, integrity and availability become part of *reliability*.

What differentiates the fields of security and reliability is that in security we need to provide these goals under the presence of a malicious entity attacking the system of interest.

4.2.2 Threat Modeling

When people say "my system is secure" they usually mean "my system is secure as long as my threat model is satisfied in practice." Incorrect threat models (such as focusing on security against *outsiders* and doing little to prevent attacks by *insiders*) are common causes for breaches of security.

An essential part of threat modeling is identifying the entities or systems that we *trust*. Trusted systems are systems we rely on (i.e., trust is accepted dependence).

System designers should try to minimize the number of entities they trust; however, trust is essential for the security of any system. There is little chance of building a secure system if everyone is an enemy. A system, computer program, or security protocol is *trustworthy* if we have *evidence* to believe it can be trusted.

4.2.3 Security Analysis

Once the security goals and policies are defined, and the threat to the system has been assessed, we begin an iterative process of designing secure mechanisms and assessing new attack vectors against our mechanisms. Secure mechanisms in this context refer to the technology employed to enforce the intended security policy.

There are several ways to analyze the security of computer systems, some of which are software testing, red teams, certifications, and formal analysis.

In this chapter we focus on describing some traditional security mechanisms. We do not describe the different approaches for analyzing the security of a system. Instead, we intend to give a high-level view of security and the most common mechanisms used to enforce it.

4.3 Cryptography

Cryptographic protocols are a fundamental building block for securing computer systems. While cryptography is not a panacea—in practice, most security problems occur because of software or design bugs, human errors, or a bad security policy—it can be argued that without cryptographic algorithms, such as hash functions, digital signatures, key distribution, and encryption schemes, we would have very little chance of securing any distributed system.

Cryptography attempts to achieve *confidentiality*, *message integrity*, and *authentication* in an insecure network of computers, devices, and resources. A standard means to model the vulnerabilities associated with such a large network is the threat model proposed by Dolev and Yao [8]. In this threat model, all communications go through an attacker who can *eavesdrop*, *intercept*, *relay*, *modify*, *forge*, or *inject* any message. The attacker can also try to impersonate other parties in the system and send messages on their behalf.

Modern cryptography gives us the basic tools to achieve confidentiality with *encryption* algorithms, message integrity with *digital signatures* and *message authentication codes*, and authentication with protocols for authentication and key establishment.

In this section, we start by describing two fundamental concepts: hash functions and the differences between secret-key and public-key cryptography. Then, we give an overview of the techniques used to provide confidentiality, integrity, and authentication.

4.3.1 Hash Functions

An important primitive in cryptography is a hash function. The basic goal of a cryptographic hash function is to provide a *seemingly random* and *compact* representation (the hash value) of an arbitrary-length input string (which can be a document or a message). A hash function has two properties: (1) it is *one way* (it is *hard* to invert, where *hard* means it is *computationally infeasible*), and (2) it is *collision resistant* (it is *hard* to find two inputs that map to the same output).

Because the input to a hash function is any arbitrary-length string, and its output is a 160-bit binary string, there must be several collisions; however, in practice we should not be able to find any collisions of a well-designed hash function.

Hash functions have a variety of applications from integrity verification to randomization functions. Network administrators can store in a database the hash of the passwords instead of the raw passwords themselves. The *one-wayness* property of hash functions prevents an attacker from obtaining the passwords if the database is compromised [9]. Furthermore, hash functions are used in several cryptographic algorithms, such as message integrity codes, digital signatures, and encryption schemes (as we explain in the next sections).

Hash functions in practice are most susceptible to collision attacks. In this attack, the adversary tries to find two inputs to the hash function that map to the same output. If hash functions are used for signature schemes, a collision attack can allow an adversary to forge a signed message [10]. Most recently SHA-1, the most popular hash function at the moment, has been successfully attacked [11, 12]. Although these attacks have not yielded a practical use for the two inputs mapping to the same hash output, the potential for finding useful attacks is increasing. In order to avoid practical attacks on hash functions, in January 2007 NIST announced [13] a public competition for a new cryptographic hash function that would become the new federal information processing standard.

4.3.2 Secret-Key and Public-Key Cryptography

In an analogy to the way locks are opened with keys, cryptographers have used the idea of a *key* to refer to the information necessary to access cryptographically protected data. Modern cryptography follows Kerckhoffs' principle, which states that "the security of a system should depend on its key, not on its design remaining obscure"[14]. In short, the common practice in cryptography is not to *rely* on the secrecy of the algorithms, only on the secrecy of the *secret keys*.

There are two types of encryption algorithms, one which uses a secret key shared between the two communicating parties, and another one in which only one party knows the secret key, and everyone else (even the adversary) knows the public key (known as public-key cryptography).

Secret-key cryptography, also known as symmetric cryptography, involves the use of a single key shared between a pair of users. The fact that you need to share a secret key with every other party that you wish to communicate with makes secret-key cryptography cumbersome for several applications.

There are two main problems with the key management in symmetric key systems. First, since secrets are shared between *pairs* of users, a large system will contain a large number of secrets, which is hard to manage. The second problem is related to the initial sharing of secrets between users. In particular, the difficulty of establishing an initial secret key between two communicating parties, when a secure channel does not already exist between them, presents a chicken-and-egg problem. This problem is most commonly solved using key distribution centers (KDC), which are trusted intermediaries between communicating parties. By having trusted intermediaries, a party only needs to share a secret key with the KDC. Whenever two new parties need to communicate, they establish a secret key with the help of the KDC. Figure 4.1 provides a simplified overview of Kerberos, one of the most common key distribution protocols.

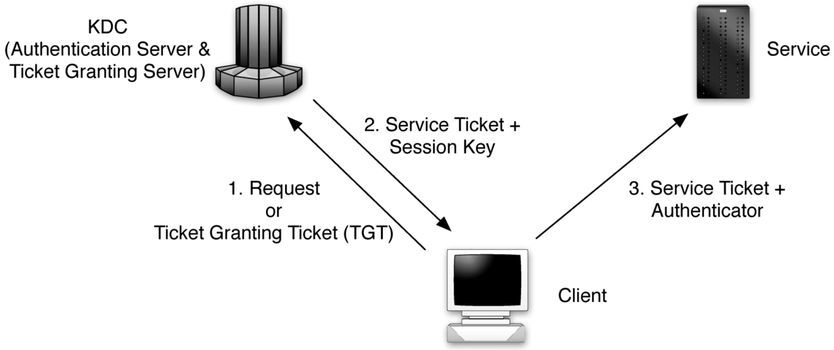

Figure 4.1 A simplified version of the Kerberos authentication system: both the client and a service share their own secret key with the KDC. To communicate securely with the service, the client sends a request to the KDC, which first authenticates the user. The KDC then selects a secret key to be shared between the client and the server (called a *session key*), and distributes it to the client and the service via a service ticket. Subsequent authentication can be streamlined by the use of ticket-granting tickets, issued by the KDC.

Using a KDC has two important shortcomings: first, the KDC introduces a single point of failure and if it crashes the whole system fails. Second, the security of the entire system breaks if the KDC is compromised. These two problems are solved through the use of public-key cryptography.

In *public-key cryptography*, also known as asymmetric cryptography, parties *do not share any secrets* and different keys are used for encrypting and decrypting. This is a particularly powerful primitive as it enables two parties to communicate secretly without having agreed on any secret information in advance.

In this setting, one party (the receiver) generates a pair of keys, called the public key and the secret key. The public key can then be made openly available so anyone can (for example) encrypt a message for the receiver. The receiver then uses its secret key to recover the message.

Public-key algorithms are less efficient than their secret-key counterparts; therefore, in practice public-key cryptography is often used in combination with secret-key cryptography. For example, in the pretty good privacy (PGP) set of algorithms for encrypting emails, a public key is used to encrypt a symmetric key. The symmetric key, in turn, is used to encrypt the bulk of the message.

Although public-key cryptography is computationally more intensive than secret-key cryptography, it requires simpler key management. However, a central problem in public-key cryptography is ensuring that a public-key is *authentic*; that is, we need to make sure that the public key we have was created by the party with whom we wish to communicate, and that it has not been modified or fabricated by a malicious party.

A public key infrastructure (PKI) provides the necessary services to distribute and manage *authentic* public keys. A trusted server called a certification authority (CA) issues a public-key certificate to each user in its system, certifying the user's public-key information. The main benefits of a CA are that it can operate *offline* and that its compromise does not lead to the compromise of the secrets of the existing users of the system. There are two main approaches to PKI: centralized (such as the X.509 model) and decentralized (such as the "web of trust" used in PGP) [15].

4.3.3 Confidentiality

Encryption schemes are used to protect messages against eavesdroppers, and therefore achieve message confidentiality. Specifically, the information we want to protect (commonly called the *plaintext*) is transformed with an encryption algorithm (also called a *cipher*) into mangled data (commonly called the *ciphertext*) such that it is unintelligible to anyone not possessing the secret key.

In general, the security of an encryption scheme depends on the difficulty of breaking the secrecy of the plaintext. The security level of an encryption scheme is defined with respect to the knowledge and the capability of the attacker. The weakest level of security is achieved against a ciphertext-only attacker who can only see the random ciphertexts. The strongest attacker has the capability to decrypt any ciphertext it wishes (except for the ciphertext the attacker is interested in breaking). For detailed definitions of these and other security notions refer to books on theoretical cryptography [5].

In practice, an encryption scheme needs to be only as secure as the system requires. In some cases, such as closed systems, a chosen ciphertext attacker cannot be realized and weaker notions of security are sufficient. Smart card systems, however, might require stronger notions of security as the attacker has more control over the input of the cryptosystem. We refer the interested reader to books on theoretical cryptography, such as [5], for detailed definitions of these and other security notions.

The most common examples of public-key encryption schemes are RSA [16, 17], and El-Gamal [18]. Both cryptosystems rely on exponentiation, which is a fairly expensive operation. More efficient schemes have been introduced that rely on elliptic curve cryptography.

Secret-key encryption algorithms can be divided into stream ciphers and block ciphers. Stream ciphers encrypt the bits of the message one at a time, and block ciphers take a number of bits and encrypt them as a single unit. The most common examples of secret-key algorithms include the data encryption standard (DES) and advanced encryption standard (AES) block ciphers, and the RC4 stream cipher.

4.3.4 Integrity

Data integrity techniques are used against unauthorized modification of messages. Specifically, the sender generates a code based on the message and transmits both the message and the code. The receiver then uses a verification algorithm that checks if the message has been altered in an unauthorized way during the transmission. The receiver can also verify that the message has indeed come from the claimed source.

4.3.4.1 Secret-key Cryptography

Data integrity in secret-key cryptography is achieved using message authentication codes (MACs). Given a key and a message, a MAC value is generated that protects the integrity of the message by allowing verifiers (who also possess the secret key) to detect any changes to the message content. MACs can also be used to provide authentication.

In general, there are two types of MAC schemes. An HMAC is based on keyed hash functions and is characterized by its efficiency. HMAC-SHA-1 and HMAC-MD5 are used within the IPsec and SSL protocols, respectively (see Section 4.4). Another type of MAC is generated based on block ciphers, such as CBC-MAC and OMAC [19].

4.3.4.2 Public-key Cryptography

Unlike secret-key cryptography, a ciphertext generated by a public-key encryption, accompanied by its associated plaintext, can provide data integrity for the plaintext and authentication of its origin. The integrity code of a message can only be generated by the owner, while the verification of the integrity check can be done by anybody (both properties of a signature). Therefore, the integrity check in public-key cryptography is called a *digital signature*. These characteristics allow for the provision of *non-repudiation*. Non-repudiation means that the owner cannot deny a connection with the message and is a necessary requirement for services such as electronic commerce. Examples of signature schemes include the digital signature standard (DSS) and RSA-PSS.

4.3.5 Authentication

There are two classes of authentication: data origin authentication and entity authentication.

Data origin authentication, also called message authentication, is the procedure whereby a message is transmitted from a purported transmitter (or origin) to a receiver who will validate the message upon reception. Specifically the receiver is concerned with establishing the identity of the message transmitter as well as the data integrity of the message subsequent to its transmission by the sender.

In entity authentication, which is concerned with validating a claimed identity of a transmitter, a "lively" correspondence is established between two parties, and a claimed identity of one of the parties is verified. Important properties of authentication include the establishment of message *freshness*: verifying whether data has been sent sufficiently recently, and user *liveness*: the lively correspondence of the communicating parties. The main techniques that handle user liveness include challenge–response mechanisms, time stamps, or freshness identifiers such as nonces.

User authentication can be divided into three categories [20]:

1. Knowledge-based authenticators ("what you know")—characterized by secrecy or obscurity, e.g., passwords, security questions such as mother's maiden name, and so forth.
2. Object-based authenticators ("what you have")—characterized by physical possession, e.g., security tokens, smart cards, and so forth.
3. ID-based authenticators ("who you are")—characterized by uniqueness to one person, e.g., a biometric such as a fingerprint or iris scan.

Different types of authenticators can be combined to enhance security. This is called *multi-factor authentication*. For example, the combination of a bank card plus a password (two-factor authentication) provides better security than either factors alone.

4.4 Network Security

In this section we give a brief overview of network security, focusing on cryptographic protocols used in the Internet for achieving secure services (such as online banking) and other security paradigms such as firewalls, intrusion detection systems, and honeypots.

The networking protocols used in the Internet can be viewed as a set of layers or protocol stack. Each layer is responsible for solving a different set of problems related to networking. The open systems interconnection (OSI) model describes a seven-layered network stack: physical layer, data link layer, network layer, transport layer, session layer, presentation layer, and application layer (shown in Figure 4.2).

In general, we are interested in providing end-to-end security, that is, ensuring that the communication between a client and a server in the Internet has been authenticated (both parties know who they are communicating with), is confidential, and has not been tampered with. In order to achieve these goals, protocols such as IPSec, SSL, and DNSSEC have been proposed.

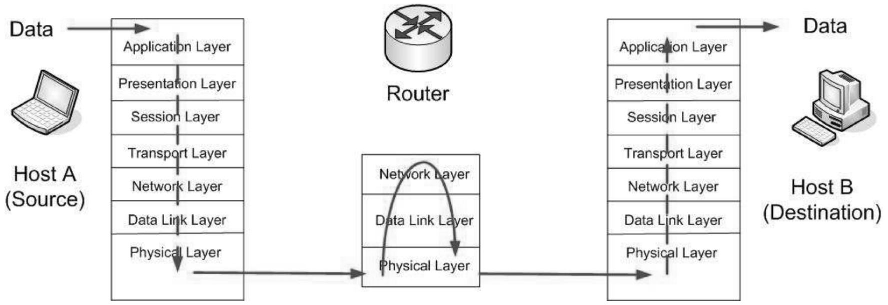

Figure 4.2 The network layer stack and routing of the data from a source to a destination.

IP Security (IPSec) The Internet protocol (IP) is the main protocol of the network layer: it provides the information needed for routing packets among routers and computers of the network. One of the shortcomings of the original IP protocol was that it lacked any kind of general-purpose mechanism for ensuring the authenticity and privacy of IP data while in transmission. Since IP data packets are generally routed between two devices, over unknown networks, any information included in the packets is subject to being intercepted or possibly changed. With the increased use of the Internet for critical applications, security enhancements for IP were needed. To this end, IPSec was developed to provide transparent end-to-end encryption of IP traffic and user data. IPSec is predominantly used in the commercial sector.

IPSec consists of two parts: the Internet key exchange (IKE) protocol, which provides mutual entity authentication and establishes a shared symmetric key, and the encapsulating security payload and authentication header (ESP/AH), which provides end-to-endconfidentiality and authentication.

One of the main uses of IPSec today is for the creation of a virtual private network (VPN). VPNs are generally used by enterprises to connect remote offices across the Internet. IPSec is used in VPNs for creating a secure channel across the Internet between a remote computer and a trusted network.

Secure Socket Layer (SSL) SSL was originally developed to provide end-to-end confidentiality, integrity, and authentication between two computers over the transmission control protocol (TCP), a protocol running at the transport layer. SSL and its successor, transport layer security (TLS), have been very popular, receiving the endorsement of several credit card companies and other financial institutions for commerce over the Internet.

SSL/TLS is commonly used with http to form https, a protocol used to secure Web pages. (Https is the protocol used when your browser shows a closed lock in one corner of the browser window, to indicate a secure connection.)

SSL is composed of a set of protocols: the protocol to ensure data security and integrity, called the SSL record protocol, and the protocols that are used for establishing an SSL connection. The latter consists of three sub-protocols: the SSL handshake protocol, the SSL change cipher protocol, and the SSL alert protocol.

The main difference between IPSec and SSL is the layer where they are implemented. The main motivation for IPSec is to avoid the modification of any layer on top of the network layer. IPSec is implemented in the operating systems, so no modifications are required from applications. The main motivation for SSL is to create a secure channel by creating (or modifying) the applications (as long as the application runs over TCP) without changing the infrastructure of the Internet or the operating system of the user.

Domain Name Service Security Extensions (DNSSEC) The domain name server (DNS) is the protocol that translates the human-readable host names into 32-bit Internet protocol (IP) addresses: it is essentially a "yellow book" for the Internet, telling routers to which IP address to direct packets when the user gives a name such as http://www.google.com.

Because DNS replies are not authenticated, an attacker may be able to send malicious DNS messages to impersonate an Internet server [21]. Although SSL can try to prevent this impersonation (the attacker's Web site should not have the secret key of the real Web site), there are many Web sites and other services that run without SSL, but that still need to be reached in a trustworthy manner.

In order to ensure the integrity of the DNS service, the IETF is currently working on DNSSEC.

Another major concern about DNS is its availability. Because a successful attack against the DNS service would create a significant communication disruption in the Internet, DNS has been the target of several DoS attacks. We explore these attacks in Section 4.6.3.

4.4.1 Firewalls, IDSs, and Honeypots

In this section we describe some very popular *non-cryptographic* tools in network security.

Firewalls A firewall is a software program or a hardware device sitting between the Internet and a private network that filters the incoming traffic to the private network [22]. If there are suspicious packets coming through the connection, the firewall prevents them from entering the private network. The concept of a firewall as an entity monitoring traffic between a local network and the Internet is shown in Figure 4.3.

A firewall can use three different approaches to filter traffic:

- *Packet filtering (inspection)*: in this method, packets are analyzed against a set of filters (rules). The packet filter is implemented at the network layer, and operates on the information available at this layer, such as

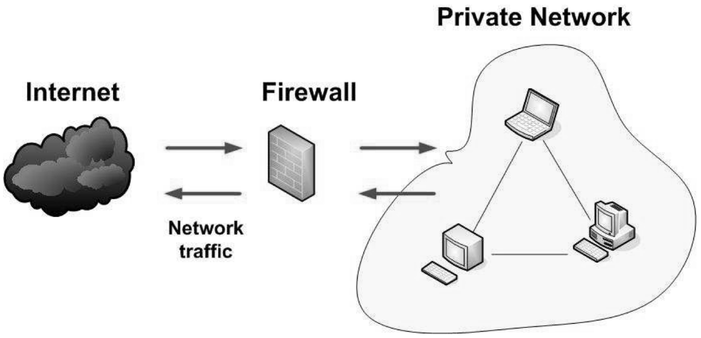

Figure 4.3 A firewall filters the traffic entering a private network from the Internet.

source address, destination address, and port information contained in each packet.

- *Application proxy*: the firewall is implemented at the application layer and processes the incoming packets all the way up to this layer. It acts as a proxy between the Internet and the private network, and verifies that the packets are clean in terms of both their origin and content. Application proxy firewall has a complete view of the network connections and application data, and as a result, it can filter bad data at different layers of the network.
- *Stateful packet filtering (inspection)*: this type of firewall adds states to the packet filter firewall. Instead of examining the contents of each packet on an individual basis, it compares certain key parts of the packet to a database of trusted information. A stateful packet filter is implemented at the transport layer. Therefore, it is not capable of monitoring the packets at the network or application layer. The advantage of this type of packet filtering is that it keeps a history of the transport layer connections, such as TCP or UDP connections. As a result, this type of firewall can help prevent attacks that exploit existing connections, or certain DoS attacks.

Intrusion Detection System (IDSs) Generally, an IDS consists of: sensors, analysis center, databases, and response center. IDSs tend to be classified in two types:

- *Signature-based detection*:[2] here, the system looks for the known attack patterns and tries to match the so-called *signature* of the attack to identify these intrusions. For example, failed login attempts could indicate a password-cracking attack. Signatures are essentially a *black-list* of activities that are not allowed in the network.

[2] This is also known as a misuse detection system.

- *Anomaly detection*: in this type of intrusion detection, the normal user behavior is defined and the system looks for any significant deviation from these established normal uses. The goal of anomaly detection is to create a *white-list* of the only activities that are allowed in the network.

One of the main problems with IDSs is their relatively large false alarm rates [23]. It is for this reason that most of the practical systems, such as *Snort*, are signature-based detectors, since anomaly detectors tend to generate more false alarms. Anomaly detection schemes are, however, a popular tool for detecting credit card fraud and other very specific scenarios.

Honeypots A honeypot is a network decoy set as a trap to detect and deflect attempts at unauthorized use of information systems. A honeypot is composed of a network site that is isolated, protected, and monitored. The network site appears to contain information or a resource that would be of value to attackers. Honeypots are used to distract adversaries from more valuable devices on a network, to provide early warning regarding new attacks and exploitation trends, and to allow in-depth examination of attackers' behavior. Most antivirus companies use honeypots to capture and study malware.

4.5 Software Security

Programming errors in computer programs can create software vulnerabilities. An attacker can compromise or obtain illegal access to databases, or cause a denial of service to computers running vulnerable software. To achieve these goals, attackers make use of an *exploit*, a computer code that takes advantage of the software vulnerability.

Software vulnerabilities are the most persistent problem in information security. It does not matter if the network security protocols or cryptographic algorithms are correct and theoretically secure if the software implementing them has errors. Ironically, even security software such as firewalls, IDSs, or implementations of security protocols (such as SSH) can open the door to attackers if they have software vulnerabilities. Therefore, preventing, detecting, and reacting to software vulnerabilities are the most active fields of information security.

In this section we give a small survey of *software security*: the study of the problems arising from programming errors and malicious software.

4.5.1 Software Vulnerabilities

In this section we briefly describe some of the main vulnerabilities in software. We start by describing early exploits, such as *buffer overflows* and *race conditions*. Then, we summarize two vulnerabilities that were discovered in the early parts of this decade and led to a wave of exploitable bugs being discovered in all kinds of programs: *format string vulnerabilities* and *integer overflows*. Finally, the increased focus on the World Wide Web in the last couple of years has pushed vulnerabilities like *SQL injection* and *cross site scripting* to become the most common reported vulnerabilities to date.

- **Buffer overflows**: A buffer overflow occurs when a computer instruction tries to store data that needs a larger computer memory space than what the program had allocated. The data that does not fit into the preallocated buffer overwrites something else in the computer memory.

  Buffer overflows have consistently been among the top reported vulnerabilities in the last two decades, and are the cause of most of the widely spread worms in the Internet. The most common type of buffer overflow is "smashing the stack" [24]. Pincus and Baker [25] provide a recent summary of this vulnerability and its variations.
- **Race conditions**: Race conditions occur when two processes compete to access the same resource before the other. One of the most common examples are the time-of-check-to-time-of-use (TOCTTOU) flaws. A TOCTTOU flaw occurs when a program checks for a particular characteristic of an object and then takes some action that assumes the characteristic still holds when in fact it does not [26].
- **Format string vulnerabilities**: Format string vulnerabilities were discovered in 2000 and immediately led to a wave of exploitable bugs being discovered in all kinds of programs [27]. These vulnerabilities arise from errors in the use of format functions (such as printf() in C). When an attacker is able to specify the format string, it can force the function to output values from parts of the computer memory that should not be available to the attacker.
- **Integer overflows**: Integer overflows occur because most programming languages give integer types a fixed maximum upper bound. When an attempt is made to store a value greater than this maximum value, an integer overflow occurs. One of the first and most significant integer overflow vulnerabilities was discovered in 2001, and allowed remote attackers to obtain full administrative privileges on an affected computer without any credentials [28].
- **Command injection**: This vulnerability occurs when data is interpreted as control data by a program. Structured Query Language (SQL) injection attacks (a particular type of command injection) allow attackers to bypass access control mechanisms of databases that support many Web applications [29]. The increase of Web applications has made SQL injection attacks widespread in the last couple of years. They currently rank among the top three vulnerabilities reported, along with buffer

overflows and cross site scripting attacks [30] (another attack to Web security).

- **Cross site scripting (XSS)**: Because Web browsers usually download and run (potentially untrusted) programs each time a user visits a Web site, browsers have implemented a *same-origin policy*, where the code downloaded from a Web site can only access the user's credentials of the Web site from where it was downloaded. In an XSS attack, an honest user is fooled to click on a malicious link that downloads a malicious program created by the attacker but that appears to have been generated from the correct Web site [31].

4.5.2 Malicious Software

Malicious software, or *malware*, is a generic term to identify computer programs with a harmful purpose. Malware usually exploits a program vulnerability to achieve its harmful purpose. There are several types of malware [1, 7, 32]:

- **Trojan**: (or Trojan horse) a program that appears to have a useful function, but also has a hidden and potentially malicious effect.
- **Computer virus**: a program that inserts itself into another program or file, and then performs a (generally) malicious action. A virus cannot run by itself, and requires that someone runs its host program.
- **Worm**: a computer program that can run independently, and copies itself from one computer to another.
- **Spyware**: software that gathers information of the users of a system without their knowledge.
- **Rootkit**: a set of tools that attempts to make malware invisible to the users of the system.

Although these definitions are generally accepted, sometimes there is no clear distinction between these concepts. For example, computer programs, such as email viruses, or email Trojans, depend on the user action to abet their propagation and are commonly referred to as worms in the media.

4.5.3 Defenses

Attempts to mitigate and solve the effects of vulnerabilities and malware can be classified as (1) detection of malicious code; (2) detection of software flaws; (3) reducing programming errors, which requires support by programming languages; (4) reducing the impact of programming errors by confining untrusted code; and (5) correcting the software flaw by patches.

Patching is a relatively straightforward technical problem. The main problems for patching systems are usually legal or economic or due to human factors. For example, patching may void certification of certain software used in critical applications, which is a legal concern. Therefore, we focus on the first four approaches in our discussion.

4.5.3.1 Detecting Malicious Code

IDSs and antivirus scanners are the most common tools for detecting attacks. IDSs focus on the detection of an exploit, or an interactive attacker, while antivirus scanners focus on the detection of malware.

Antivirus scanners are currently the most popular defense for malware. These malware detectors often look for known sequences of code (*signatures*). In order to obstruct code analysis, malware writers sometimes use *polymorphism*, *code obfuscation*, and *encryption* [33].

One of the fundamental results in the theory of malware asserts that it is undecidable whether an arbitrary program contains malware [34, 35]. Therefore, it is impossible to create an algorithm that can recognize all malicious logic instructions from normal program instructions. Any defense will be imprecise, allowing false negatives, meaning not recognizing malware, and/or false positives that are labeling non-malicious code as malware.

4.5.3.2 Software Testing

Finding software errors is very difficult. In his Turing Award lecture, Edsger W. Dijkstra stated, “Program testing can be quite effective for showing the presence of bugs, but it is hopelessly inadequate for showing their absence” [36]. A practical example given more than 10 years later by Thompson [37] shows that no amount of source-level verification or scrutiny can stop using untrusted code.

Moreover, Rice’s theorem, a basic theoretical result, states that there is no general computer algorithm that can classify code as safe or malicious with perfect accuracy. Software testing tools, therefore, produce false negatives (the program contains bugs that the tool does not report) or false positives (the tool reports bugs that the program does not contain).

Despite these negative results, approximate software testing algorithms have been very successful in finding several software flaws.

Automatic software testing can be divided into *static analysis*, analysis of programs without their execution [26, 38, 39, 40], and *dynamic analysis*, analysis of programs by executing them [41–45].

4.5.3.3 Type-safe Languages

Many vulnerabilities are caused because programmers have to manage the memory used by their programs. Type-safe programming languages avoid these vulnerabilities. A data type, such as the data type int in C, assigns meaning to values and variables in a program. Type safety is the property of a programming language that does not allow the programmer to treat a value as a type it does not belong to. Since a data type defines specific memory requirements for values and variables, type safety implies memory safety. A type-safe language prevents several memory management errors, such as buffer overflows. Java is a type-safe language, whereas C and C++ are not type-safe languages.

4.5.3.4 Confining Code

Most users and operating systems allow any program to read, modify, or delete any file in the computer. This allows an attacker to get full control of the system using any vulnerabilities of a program. Of interest is finding ways to limit the privileges that a program uses when running.

The *principle of least privilege*, by Saltzer and Schroeder [46], is an old design principle. The idea is to give a program only the privileges it needs to accomplish its task. By limiting the privileges, the damage is limited when the program is compromised.

Sandboxing is a way to limit the ability of untrusted code for doing harmful things. A sandbox executes code in a heavily restricted environment, limiting access to the file system or the ability to open network connections. Janus [47] is a research example of sandboxing: it monitors untrusted applications and disallows system calls that the untrusted code is not permitted to execute. A practical example of sandboxing is *Systrace* [48], a utility available in BSD, Linux, and Mac OS X that can limit how a computer program accesses the operating system. Another popular example is the Java applet. Applets are typically executed in a sandbox, preventing the untrusted code that is usually downloaded from Web sites from accessing information on the executing computer.

Java also uses *code-signing*, in which a code producer signs its code to provide a proof of trustworthiness. Based on the reputation of the code producer, Java (or the user) can decide whether the code can be trusted to be executed.

Proof carrying code is another technique to avoid trusting the producer of the code. Here, the user specifies a safety requirement, and the creator of the code must generate a proof that the code meets the desired safety properties and integrates them with the code [49].

4.6 Cyber Attack Trends, Threats, and Homeland Security

The Internet and information technology at large have become ingrained in our lives. This assimilation of information technology in our lives has made cyber attacks more attractive. There is a clear shift from cyber attacks carried out by a few technically savvy and curious hackers, to cyber attacks that can be mounted by a wide range of people or groups. The latter has clearly defined economic, political, religious, or national motivations.

In this section we discuss these trends by studying some recent examples. We start with cyber crime and botnets that are the driving force behind most of the Internet-based attacks. Worms and distributed denial of service (DDoS) attacks that threaten the communication infrastructure of the Internet are explained next. Finally, we describe cyber espionage and cyber attacks against the critical infrastructures, which are of paramount importance in homeland security.

4.6.1 Cyber Crime and Botnets

While the occasional widespread worm (Section 4.6.2) or significant DDoS attack (Section 4.6.3) steals the media attention, there is a much more prevalent group of attacks keeping a low profile since attention can bring down their profits. Malfeasants do not want people to realize that their computers have been Q1 compromised and are being used to host illegal content, to send spam, or to perform DDoS attacks.

There is a large underground market based on a wide variety of criminal activities. Theft of intellectual property, extortion based on the threat of DDoS attacks, fraud based on identity theft, credit card fraud, spamming, phishing, sales of *bots* (i.e., compromised computers), sales or rentals of *botnets* (i.e., networks of compromised computers under the control of the botnet owner), sales of stolen source code from software companies, and sales of malware or tools to create attacks (the vendor even provides technical support) are just a few examples of ways that they are making it easier for unskilled attackers to commit cyber crimes.

The prevalence of these activities and the large amount of compromised computers in the world can be recognized by the fact that the average compromised computer is being sold by an estimated average of 4 cents [50]. Q2

Botnets are one of the most coveted electronic goods. With several million messages sent per day, sending spam is the primary source of income for botnet operators. Botnets can be rented and sold for performing attacks, such as DDoS attacks (see Section 4.6.3). Other forms of attacks are, however, possible. For example, a botnet attack, targeting the user accounts in eBay and making fraudulent transactions, was detected in September 2007 [51].

Botnet operators are also very active in protecting their networks. They buy and steal botnets from each other, protect their compromised computers against compromise by other people, and actively attack security companies focusing on defeating spam. For example, in 2006, botnets launched DDoS attacks against BlueSecurity (forcing the company to abandon its antispam efforts), and other antispam and antimalware groups such as CastleCops, Spamhaus, and URIBL were victims of DDoS attacks in 2007. From the standpoint of those defending against cyber war, the problem is that the existence of lucrative spamming and underground criminal business keeps the botnets in operation and motivated to stick around and fight security companies [52].

At the time of this writing, the Storm botnet (first detected in January 2007) was considered to be the largest botnet identified by security researchers. The number of compromised computers it controls has been estimated to be from several hundred thousands up to 50 million [53]. Although estimating the size of a botnet is a very difficult task, and estimates themselves may not be very reliable [54], it is clear that the size of botnets is very large.

The Storm botnet is also highly resilient to efforts to take it down. Its command and control architecture is based on a peer-to-peer (P2P) network, with several redundant hosts spread among 384 providers in more than 50 countries. Furthermore, in order to hide infected machines and malicious Web sites, the Storm botnet uses a technique called *fast flux*, a method in which the DNS records of a Web site's domain name are constantly rotating and changing their IP addresses. Additionally, the infection code is polymorphic, so antivirus designers have a harder time crafting signatures against it. Storm's malware also tries to avoid infecting virtual machines and, therefore, security researchers have less opportunities to study the malicious code—virtual machines are typically used in honeypots; see Section 4.4.1.

4.6.2 Widely Spread Malware

In this section we summarize the recent worms, email viruses, and email Trojans, focusing on their side effects and their impact in different infrastructures.

The timeline of widespread worms and viruses did not see any major contenders to the original Internet worm of 1988 until 1999, when the Melissa email virus was released. Although the virus did not carry a malicious payload, it saturated email servers, creating an unintentional DoS attack. A similar email virus, the ILOVEYOU virus, was released a year later.

In 2001, three major Internet worms were released [55]: Code Red 1, Code Red 2, and Nimda. Code Red 1 infected approximately 360,000 computers in just six days, creating routing disruptions in the Internet. From the 20th until the 27th of each month it attempted to use all the compromised computers to launch a DDoS attack against the IP address of the White House. The White House responded to this threat by changing its IP address. Code Red 2 and Nimda showed improved spreading techniques over Code Red 1. Computers infected by Code Red 2 searched for vulnerable computers in their vicinity (local area networks), and Nimda had different methods to spread itself (IIS vulnerability, emails, and scanning for backdoors). Code Red 2 left a backdoor in infected computers, allowing an attacker to connect and control the computer at a later stage, thus having the potential of creating a botnet. Interestingly enough, Nimda used these backdoors to help its propagation.

Q3

The fastest spreading worm to date was released in 2003. Slammer compromised over 75,000 computers in just 10 minutes by exploiting (via UDP) a buffer overflow in Microsoft's SQL Server software. Due to this fast spread, the worm disrupted parts of the Internet, the phone service in Finland, airline reservation systems, credit card networks, 13,000 ATMs on the Bank of America network [56], and it shot down display systems at the Davis-Besse power plant in Ohio [57]. A couple of months later, the Blaster worm was released, and it was widely suspected of contributing to a power loss at a plant providing electricity to parts of New York state.

Although these worms created widespread damage, this damage was mostly a side effect of the spreading behavior of the worm. These attacks could have been much more damaging if they had included a malicious payload. In 2004, Witty carried such a destructive payload: it corrupted randomly the hard drive sections in compromised computers. But this was not the only accomplishment of Witty [58]. Witty showcased the increased sophistication of worms, since it started compromising hosts with a list of known vulnerable computers, and it was released just one day after the ISS firewall vulnerability was made public.

The current most widely spread computer infection is due to the Storm Trojan/virus. Storm has been used to construct the largest botnet to date. At the time of this writing, the botnet is still active and increasing. It spreads via emails directing users to a Web page hosting the malicious code, or to open a malicious Trojan that attempts to exploit several vulnerabilities in Windows computers. Storm also reflects the trend of creating worms and email Trojans for profit. The size of the Storm botnet should also be a concern for any nation, since the owner of the botnet can not only use it to commit fraud against hundreds of citizens in many countries, but it can potentially launch DDoS attacks several times larger than the ones against Estonia in 2006.

4.6.3 DDoS Attacks, Estonia, and Hacktivism

A DoS attack can be defined as an attack designed to disrupt, or completely deny, legitimate users' access to networks, servers, services, or other resources. In a distributed denial of service (DDoS) attack a botnet (or several botnets) launches a coordinated attack onto a single computer by flooding it with massive volumes of useless traffic. The flood of incoming messages to the target can occupy all the resources that the target could use to service legitimate traffic. Figure 4.4 illustrates this concept.

The first widely publicized DDoS incident occurred in 2000, when several of the most popular Web sites—Yahoo, Amazon.com, E-Trade, eBay, and CNN.com—fell victim to DDoS attacks.

Since then, DDoS attacks have become commonplace. The vast majority of DDoS attacks are not publicized and include a wide range of global victims,

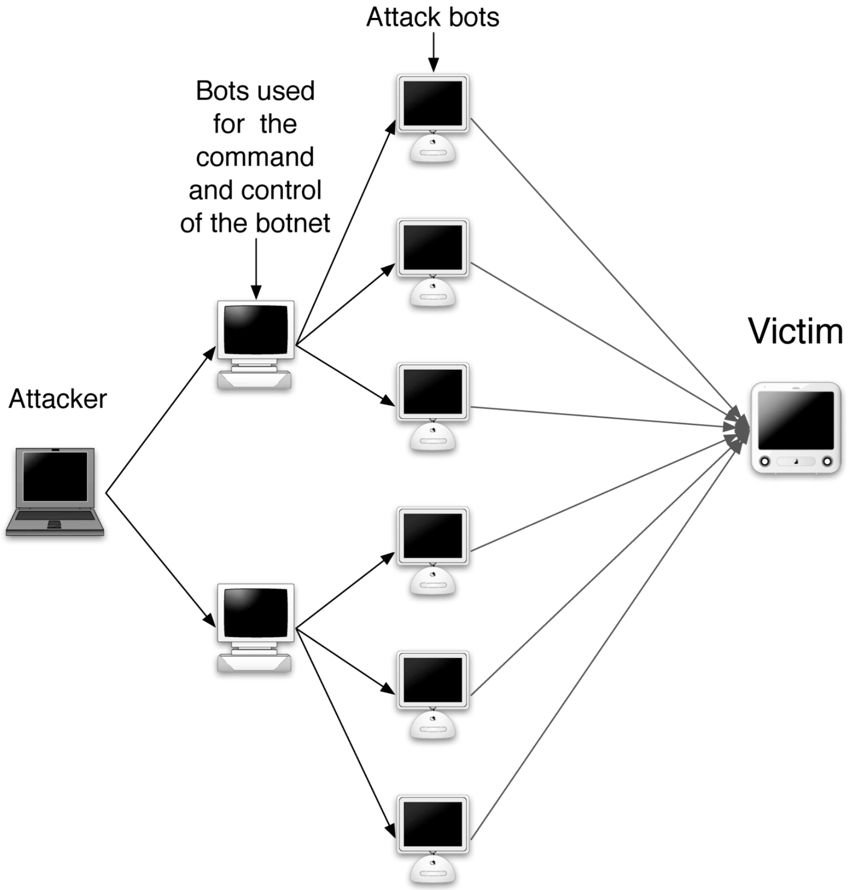

Figure 4.4 Using a (possibly rented) botnet, an attacker can launch a DDos attack on a desired victim by flooding it with useless connections.

from small commercial sites to educational institutions, public chat servers, and government organizations. In a recent study, Moore et al. [59] estimated that the vast majority of victims of the attacks were home users and small businesses rather than large corporations. Over three years, the researchers witnessed over 68,000 attacks, and yet this number is an underestimate, since they only monitored attacks where the bots used spoofed IP addresses (and spoofing IP addresses is not necessary in DDoS attacks).

Two of the most relevant DDoS attacks for the scope of this book are the DNS attacks in 2002 and 2007, and the attacks against Estonia in 2006.

Although the DDoS attacks against the DNS infrastructure in 2002 and 2007 had little impact on Internet users, they are relevant because of the potential consequences of a successful attack: if the attacks had been successful, the Internet communication infrastructure would have suffered a major disruption in service. However, since the DNS service is one of the most resilient services on the Internet, the amount of resources required to mount a successful attack is extraordinarily large. On the other hand, the attacks against Estonia were extremely successful by using even less resources than those required for the DNS attacks and much less than the resources used by botnets in their battles against antispam companies [52].

The cyber attacks against Estonia began as a retaliation to Estonia's plans for moving Soviet-era war memorials. The attacks were incited via Russian-language chat rooms: some people volunteered their computers by following the instructions on how to launch a ping attack and some other attackers defaced several Estonian Web sites. However, most of the damage came as DDoS attacks from botnets. The DDoS attacks blocked online access to banks and government Web sites, including the government ministries.

Estonia initially blamed Russia for the attacks, but the Kremlin denied the accusations. Most experts found the evidence of official Russian involvement weak, and concluded that the attacks in Estonia were the work of *hacktivists*, activists that use the Internet as a tool to advance their cause. At the end, Estonia accepted the idea that it was the work of individuals, but hopes Russia will cooperate in tracking the perpetrators.

Hacktivist activities are nothing new [60]. Some of the activities sparked by international tensions include the Web defacements between China and Taiwan in 1999, between Israel and Palestine in 2000, between China and the US after the bombing of the Chinese embassy in Belgrade in 1999, and in 2001 when one of the US spy planes collided with a Chinese airplane. In the recent Iraq war, there were a series of DDoS attacks against Al Jazeera's Web site, and the defacement of the Iraqi government Web site [61]. Although in several cases there is speculation about government involvement in these attacks, most of them are assumed to be part of independent groups.

However, even if these attacks were the product of independent groups, they should serve as a reminder of the fragility of the security of several Internet Web sites, and of the threat posed by large botnets. Based on estimated prices for renting botnets, the attacks on Estonia could have been carried by $100,000 [52], a sum within the reach of independent groups.

4.6.4 Cyber Espionage and the Athens Affair

Cyber espionage is another major threat for the security of a nation. It is natural to assume that several nations (or even independent groups) perform some level of espionage or intelligence gathering on each other by complex computer network attacks [62]. However, the details of these attacks are generally kept confidential.

One of the cases that researchers have been able to study and disclose publicly was the "Athens affair" [63], a case in which hackers broke into Vodafone's telephone network in Athens and subverted its built-in wiretapping features for their own purposes.

The cell phone bugging began sometime during the run-up to the August 2004 Olympic Games in Athens and remained undetected until January 24, 2005. The tapped cell phones included the prime minister of Greece, the defense and foreign affair ministers, top military and law enforcement officials, the Greek EU commissioner, activists, journalists, the mayor of Athens, and at least 100 other high-ranking dignitaries.

In 2005, while looking into what appeared to be a glitch, Ericsson found unauthorized software had been installed in two of Vodafone's central offices. The tapping is possible in the central offices because GSM calls are only encrypted between cell phones and the base station. However, the main reason the attack succeeded was because the rogue software used the lawful wiretapping mechanisms of Vodafone's digital switches to tap about 100 phones, a wiretapping mechanism that is supposed to be available only to the law enforcement agencies in Greece and with the appropriate warrants.

After the forensic investigation was performed on the rogue software, it was concluded that the developers who created this malware were experts. The intruders included a backdoor to install, operate, and update their wiretapping software without being detected by Vodafone or Ericsson. The software also included a rootkit that made it invisible to network operators and deleted all logs of its activities. Finally, the program was written in the PLEX language, a language where most of the information is available to only very trained individuals. Due to these facts, people have speculated that the culprits behind the attacks required the resources available only to insiders, or to a foreign spy agency.

The implications of this incident are far reaching and very relevant due to current developments in the US. First, the recent warrantless wiretapping advances by the US government on calls within the US have raised concerns because these procedures can in fact make the job of a hacker (or foreign spy agency) easier for listening to these conversations [64].

Additionally, the recently released information about DCSNet [65], the wiretapping network that makes it easy for FBI operatives to tap into conversations, has raised similar concerns. Critics argue that the lack of proper security measures for DCSNet means that there is no trustworthy way to prevent an attacker from exploiting the system; in other words, creating an infrastructure for wiretapping (in addition to recent warrantless wiretapping laws under the "Protect America Act") just makes the job of a malicious wiretapper easier [66].

4.6.5 Critical Infrastructure and Cyber Security

All of the critical infrastructures (energy, telecommunications, transportation, banking and finance, continuity of government services, water supply systems, gas and oil production, and emergency services) are dependent on the computer communication infrastructures. Moreover, the computer information infrastructures are themselves dependent on many of the critical infrastructures, such as electric power grid and telecommunications systems. A successful cyber attack on the supervisory control and data acquisition (SCADA) and other control systems for the critical infrastructures could have a significant impact on public health, economic losses, and potential loss of lives. Securing control systems in critical infrastructures is thus a national priority for the department of homeland security [67].

To date, there have been relatively few attacks on the critical infrastructures. However, security studies from the U.S. Department of Energy (DOE) and commercial security consultants have demonstrated the cyber vulnerabilities of control systems [67–69]. In one of the most recent demonstrations of the vulnerability of the critical infrastructure, a security researcher was able to break into a nuclear power station and within a week take over the control plant [70]. This is not the first time that a critical infrastructure has been penetrated. In 2000, a 48-year-old Australian man, who was fired from his job at a sewage-treatment plant, remotely accessed his workplace computers and poured toxic sludge into parks and rivers [70].

In January 2003, computers infected with the Slammer worm (SQL Server worm) shut down safety display systems at the Davis-Besse power plant in Oak Harbor, Ohio. A few months later, another computer virus was widely suspected by security researchers of leading to a power loss at a plant providing electricity to parts of New York state. A third incident was the power outage of August 2003 in the Midwest and Northeast of the United States, and Canada. Even though the incident was not an act of terrorism, it demonstrates the vulnerability of the electric power grid. In fact, some of the documents gathered from Al Qaeda in 2002, suggested that they were considering a cyber attack on the power grid.

Cyber security is one of the most fundamental aspects of critical infrastructure protection. We need to study and assess the threats and risks of possible cyber attacks, and implement suitable defenses.

4.7 Conclusions

There has been a lot of speculation about state-funded cyber militias and espionage, but without identifying the clear culprits of several of the recent attacks, there is no conclusive evidence for most claims. However, three things are clear: (1) most governments are investing in cyber warfare activities (in defense and offense), given that any conflict in the twenty-first century will necessarily involve the use of information technology, (2) any conflict, or international tension in the physical world, will have its counterpart effect in the Internet, and (3) there are large economic incentives for crime in the Internet. If we leave these activities unchecked, they will increase.

Although cyber war, cyber terrorism, and hacktivism pose real threats, they tend to attract most of the media attention and public opinion, leaving the thriving criminal activities on the Internet relatively unnoticed. Cyber crime is also expected to expand to new technologies: cell phones, personal digital assistants, music players, and embedded hardware can give rise to new vulnerabilities and risks.

Even though the threats and concerns about cyber security seem daunting, we believe that efforts to raise public awareness, new policy rules, investment in research for cyber security technologies, and an expansion in diplomacy and international cooperation for tracking and prosecuting criminals across different countries will greatly help in securing cyberspace. By providing a better cyber security, governments will be one step closer to providing homeland security to their citizens.

Acknowledgments

This work was supported in part by TRUST (Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award number CCF-0424422) and the following organizations: AFOSR (#FA9550-06-1-0244) Cisco, British Telecom, ESCHER, HP, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia, and United Technologies.

References

[1] M. Bishop. *Computer Security, Art and Science*. Addison-Wesley, 2003.

[2] C. Kaufman, R. Perlman, and M. Speciner. *Network Security:Private Communication in a Public World*. Prentice-Hall, 2nd ed, 2002.

[3] R. Anderson. *Security Engineering*. Wiley, 2001.

[4] D. Gollmann. *Computer Security*. Wiley, 2nd ed., 2006.

[5] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2007.

[6] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.

[7] R. Shirey. *RFC 2828—Internet Security Glossary*. http://www.faqs.org/rfcs/rfc2828.html, 2000.

[8] D. Dolev and A. C. Yao. "On the security of public key protocols." *IEEE 22nd Annual Symposium on Foundations of Computer Science*, pages 350–357, 1981.

[9] X. Boyen. "Halting password puzzles—hard-to-break encryption from human-memorable keys." *16th USENIX Security Symposium—SECURITY 2007*, pages 119–134, August 2007.

[10] A. Lenstra, X. Wang, and B. de Weger. Colliding X.509 certificates. Report 067, Cryptology ePrint Archive, 2005.

[11] C. De Cannière and C. Rechberger. "Finding SHA-1 characteristics: General results and applications." *Advances in Cryptology— Asiacrypt*, pp. 1–20, 2006.

Q4 [12] C. De Cannière and C. Rechberger. "SHA-1 collisions: Partial meaningful at no extra cost?" 2006.

[13] *NIST's Plan for New Cryptographic Hash Functions*. http://www.csrc.nist.gov/pki/HashWorkshop/index.html, January 24, 2007.

[14] A. Kerckhoffs. "La cryptographie militaire." *Journal des Sciences Militaires*, pages 5–38, January 1883.

[15] R. Perlman. "An overview of PKI trust models." *IEEE Network*, 13(6):38–43, November/December 1999.

[16] R. Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signatures and public-key cryptosystems." *Communications of the ACM*, 21(2):120–126, 1978.

[17] M. Bellare and P. Rogaway. "Optimal asymmetric encryption—How to encrypt with RSA." *Advances in Cryptology—Eurocrypt '94*, 1995.

[18] T. El-Gamal. "A public-key cryptosystem and a signature scheme based on discrete logarithms." *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[19] W. Mao. *Modern Cryptography: Theory and Practice*. Prentice-Hall, 2004.

[20] L. O'Gorman. "Comparing passwords, tokens, and biometrics for user authentication." *Proceedings of the IEEE*, 91(12):2021–2040, December 2003.

[21] S. M. Bellovin. "Using the domain name system for system break-ins." *Proceedings of the 5th USENIX Security Symposium*, 1995.

[22] W. Cheswick, S. M. Bellovin, and A. D. Rubin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley Professional Computing Series, 2002.

[23] A. A. Cárdenas, J. S. Baras, and K. Seamon. "A framework for the evaluation of intrusion detection systems." *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 63–77, Oakland, May 2006.

[24] Aleph One. Smashing the stack for fun and profit. *Phrack Magazine*, 7(49), 1996.

[25] J. Pincus and B. Baker. "Beyond stack smashing: Recent advances in exploiting buffer overruns." *IEEE Security & Privacy Magazine*, 2(4):20–27, July–August 2004.

[26] M. Bishop and M. Dilger. "Checking for race conditions in file accesses." *Computing Systems*, 9(2):131–152, 1996.

[27] Scut/Team Teso. *Exploiting Format String Vulnerabilities*. http://julianor.tripod.com/teso-fs1-1.pdf, March 2001.

[28] D. Ahmand. "The rising threat of vulnerabilities due to integer errors." *IEEE Security & Privacy Magazine*, 1(4), July–August 2003.

[29] Z. Su and G. Wassermann. "The essence of command injection attacks in Web applications." *33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 372–382, 2006.

[30] S. Christey and R. A. Martin. *Vulnerability Type Distributions in* CVE. http://cwe.mitre.org/documents/vuln-trends/index.html, May 22, 2007.

[31] A. Klein. "Cross site scripting explained." *Sanctum White Paper*, May 2002.

[32] E. H. Spafford. "A failure to learn from the past." *Proceedings of the 19th Annual Computer Security Applications Conference*, pp. 217–231, December 8–12, 2003.

[33] A. Shevchenko. *The Evolution of Self-Defense Technologies in Malware*. http://www.viruslist.com/analysis?pubid=204791949, July 2007.

[34] L. Adleman. "An abstract theory of computer viruses." *Advances in Cryptology—Proceedings of CRYPTO '88*, pp. 354–374.

[35] F. Cohen. "Computational aspects of computer viruses." *Computers and Security*, 8(4):325–344, November 1989.

[36] E. W. Dijkstra. *The Humble Programmer*. ACM Turing Lecture, 1972.

[37] K. Thompson. "Reflections on trusting trust." *Communications of the ACM*, 27(8):761–763, August 1984.

[38] B. Chess and G. McGraw. "Static analysis for security." *IEEE Security & Privacy Magazine*, 2(6):76–79, November 2004.

Q5 [39] D. Wagner, et al. "A first step towards automated detection of buffer overrun vulnerabilities." *Proceedings of the 7th Networking and Distributed System Security Symposium (NDSS)*, pp. 3–17, February 200.

[40] J. Yang, et al. "MECA: an extensible expressive system and language for statically checking security properties." *Proceedings of the 10th ACM conference on Computer and Communications Security*, pp. 321–324, 2003.

[41] A. Orso and A. Zeller, (eds.) *Fifth International Workshop on Dynamic Analysis (WODA 2007)*, 2007.

[42] B. Arkin, S. Stender, and G. McGraw. "Software penetration testing." *IEEE Security & Privacy Magazine*, 3(1):84–87, January 2005.

[43] Y. Xie and A. Aiken. "Static detection of security vulnerabilities in scripting languages." *15th USENIX Security Symposium*, pages 179–192, 2006.

[44] V. B. Livshits and M. S. Lam. "Finding security vulnerabilities in Java applications with static analysis." *Proceedings of the 14th USENIX Security Symposium*, pages 271–286, 2005.

[45] E. Haugh and M. Bishop. "Testing C programs for buffer overflow vulnerabilities." *Proceedings of the 10th Network and Distributed System Security Symposium (NDSS)*, pages 123–130, February 2003.

[46] J. H. Saltzer and M. D. Schroeder. "The protection of information in computer systems." *Proceedings of the IEEE*, 63(9):1278–1308, September 1975.

[47] I. Goldberg, et al. "A secure environment for untrusted helper applications: Confining the wily hacker. *Proceedings of the 6th USENIX Security Symposium,* pages 1–13, July 1996.

[48] N. Provos. "Improving host security with system call policies." *Proceedings of the 12th USENIX Security Symposium*, August 2003.

[49] G. Necula. "Proof-carrying code." *24th ACM SIGPLAN-SIGACT, Symposium on Principles of Programming Languages*, pp. 106–119, January 1997.

[50] R. Thomas and J. Martin. "The underground economy: Priceless." *Login: The USENIX Magazine*, 31(6):7–16, 2006.

[51] G. Keizer. *Botnet Steals eBay Accounts. PC World*, http://www.pcworld.com/article/id,136729-c,onlinesecurity/article.html, September 4, 2007.

[52] M. Lesk. "The new front line. Estonia under cyberassault." *IEEE Security & Privacy Magazine*, 5(4):76–79, July–August 2007.

[53] S. Gaudin. *Storm Worm Botnet More Powerful Than Top Supercomputers. iTnews*, http://www.itnews.com.au/News/60752,storm-worm-botnetmore-powerful-than-top-supercomputers.aspx, September 7, 2007.

[54] M. Abu Rajab, et al. "My botnet is bigger than yours (maybe, better than yours): Why size estimates remain challenging." *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots '07)*, April 10, 2007.

[55] S. Staniford, V. Paxson, and N. Weaver. "How to own the Internet in your spare time." *Proceedings of the 11th USENIX Security Symposium*, pages 149–167, 2002.

[56] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. "Inside the slammer worm." *IEEE Security & Privacy Magazine*, 1(4):33–39, July 2003.

[57] K. Poulsen. *Slammer Worm Crashed Ohio Nuke Plant Network*. Security Focus, http://www.securityfocus.com/news/6767, September 2003.

[58] C. Shannon and D. Moore. "The spread of the Witty worm." *IEEE Security & Privacy Magazine*, 2(4):46–50, July–August 2004.

[59] D. Moore, C. Shannon, D. Brown, G. M. Voelker, and S. Savage. "Inferring Internet denial-of-service activity." *ACM Transactions on Computer Systems*, May 2006.

[60] D. Denning. "Cyberwarriors: Activists and terrorists turn to cyberspace." *Harvard International Review*, 23(2), 2001.

Q6 [61] "Hacktivists attack websites in war protests." *Network Security*, (4):2, April 2003.

Q7 [62] D. Sevastopulo. *Chinese Military Hacked into Pentagon. Finantial Times*, http://www.ft.com/cms/s/0/9dba9ba2-5a3b-11dc-9bcd-0000779fd2ac.html, September 3, 2007.

[63] V. Prevelakis and D. Spinelis. "The Athens affair." *IEEE Spectrum*, 44(7):26–33, July 2007.

[64] S. Landau. "A gateway for hackers." *Washington Post*, p. A17, August 9, 2007.

[65] R. Singel. *Point, Click... Eavesdrop: How the FBI Wiretap Net Operates. Wired Magazine*, http://www.wired.com/politics/security/news/2007/08/wiretap, August 29, 2007.

Q8 [66] S. Bellovin, et al. "*Risking Communications Security: Potential Hazards of the "Protect America Act*." Technical Report, http://www.crypto.com/papers/paa-comsec-draft.pdf, October 2007.

[67] United States Government Accountability Office. “Critical infrastructure protection. Multiple efforts to secure control systems are under way, but challenges remain.” Technical Report GAO-07-1036, Report to Congressional Requesters, 2007.

[68] J. Eisenhauer, et al. *Roadmap to Secure Control Systems in the Energy Sector*. Energetics Incorporated. Sponsored by the U.S. Department of Energy and the U.S. Department of Homeland Security, January 2006.

[69] R. J. Turk. “Cyber incidents involving control systems.” Technical Report INL/EXT-05-00671, Idaho National Laboratory, October 2005.

[70] A. Greenberg. *America’s Hackable Backbone. Forbes*, http://www.forbes.com/logistics/2007/08/22/scada-hackers-infrastructuretech-security-cx ag 0822hack.html, August 2007.