



realtimerepublishers.comtm

The Definitive Guidetm To

Information Theft Prevention

Blue  **Coat**[®]

Dan Sullivan

Chapter 3: Overview of Key Technologies to Prevent Information Theft	43
Preventing Theft by Preserving Confidentiality	43
Preserving Confidentiality with Encryption	44
Symmetric Key Cryptography	45
Asymmetric Key Cryptography	46
Combining Symmetric and Asymmetric Key Cryptography.....	47
Preserving Message Integrity.....	48
Preserving Data Integrity by Controlling Access	50
Identifying Users.....	51
Authenticating Users.....	51
User Authorizations	51
Auditing	54
Maintaining Availability and Integrity by Protecting the Infrastructure	54
Maintaining a Secure Network	54
Perimeter Defenses: Firewalls	55
IPSs	56
Network-Based IPSs	56
Host-Based IPSs.....	56
Signature-Based Detection.....	57
Behavior-Based Detection	57
Content Filtering and Anti-Malware Systems	57
Evolving Nature of Threats.....	57
Anti-Malware: Protecting Against Malicious Software	58
Beyond Malicious Programs: Filtering Inappropriate Content.....	60
Maintaining Secure Servers, Desktops, and Other Devices	61
The Role of Patch Management in Systems Security.....	61
Vulnerability Testing	62
Hardening OSs	63
Securing Applications.....	63
Managing Client Security	63
Summary	65
Download Additional eBooks from Realtime Nexus!	65

Copyright Statement

© 2006 Realtimepublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimepublishers.com, Inc. (the “Materials”) and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtimepublishers.com, Inc or its web site sponsors. In no event shall Realtimepublishers.com, Inc. or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

Realtimepublishers.com and the Realtimepublishers logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners.

If you have any questions about these terms, or if you would like information about licensing materials from Realtimepublishers.com, please contact us via e-mail at info@realtimepublishers.com.

[**Editor's Note:** This eBook was downloaded from Realtime Nexus—The Digital Library. All leading technology guides from Realtimepublishers can be found at <http://nexus.realtimepublishers.com>.]

Chapter 3: Overview of Key Technologies to Prevent Information Theft

A common practice in information security is to implement security in-depth, that is, with multiple layers using a mix of technologies. This practice has been adopted because, as is often the case in information technology (IT), there is no single solution that solves all problems in this domain. This chapter examines the major technologies that have a role in supporting information security and on-demand security in particular. The technologies are grouped according to the following high-level security objectives:

- Preventing information theft by preserving confidentiality
- Preserving data integrity by controlling access
- Maintaining availability and integrity by protecting the infrastructure
- Handling special issues in managing client security

Each of these areas presents particular challenges that must be addressed to maintain a secure infrastructure.



It should be noted, though, that the groupings used here are based on organizational objectives and not technical differences; technologies discussed in one section often play a role in other areas as well. For example, encryption is discussed as a technology for protecting confidentiality but is also a fundamental component of authentication systems used to preserve data integrity.

Preventing Theft by Preserving Confidentiality

Maintaining confidentiality is a fundamental objective of information security. There are many reasons for keeping information confidential, ranging from personal privacy—for example, in the case of healthcare information—to national security, such as with military secrets. The form confidential information takes on can also vary; examples include:

- Emails between business partners that contain proprietary information about a new product design
- Hospital databases of patient clinical records
- Credit card information transmitted between a retailer and a credit card processing center
- Documents containing information about sensitive business negotiations

These examples include information that is either transmitted over networks, such as emails and credit card transactions, or stored for extended periods, such as database records and files. In both cases, users would want to make sure the confidential information is not disclosed.

They would also want to ensure that it is not tampered with. For example, email messages should not be changed en route to their destination, and database records should not be altered by unauthorized personnel or processes. Concerns about maintaining the integrity of information often occur alongside concerns about confidentiality. Fortunately, the fundamental techniques used to preserve confidentiality are closely related to maintaining integrity of information.

Three core topics related to confidentiality and to some aspects of integrity are:

- Encryption
- Message digests
- Key management

Encryption includes techniques for masking information so that anyone other than the intended recipient cannot read the information. Message digests use specialized mathematical functions to determine whether a message has been tampered with en route. Key management is the practice of creating, storing, and sharing data used to encode messages, known as cryptographic keys.

Preserving Confidentiality with Encryption

The art of encoding messages has a long history. Ancient Hebrews, Greeks, and Romans used a variety of crude encoding methods. The atbash method used by Hebrews was based on substituting a letter with its corresponding letter in a reverse order alphabet. For example, the source word “security” would encode as “hvxfigb” using the following source and target alphabets:

Source: abcdefghijklmnopqrstuvwxyz

Target: zyxwvutsrqponmlkjihgfedcba

The Caesar Substitution cipher was similar but rather than creating the target alphabet by reversing the source alphabet, one would simply shift the letters a fixed number of positions. For example a 3-shift would yield the following mappings:

Source: abcdefghijklmnopqrstuvwxyz

Target: xyzabcdefghijklmnopqrstu

Although these techniques are easily applied, they are also easily circumvented. Modern techniques are more resilient to cracking, or cryptanalysis, but are more complex. When using cryptographic systems to encode messages, one must balance the strength of an encryption technique with the time and resources required to encrypt the message.

Regardless of the specific technique used, the basic workflow of encryption is shown in Figure 3.1.

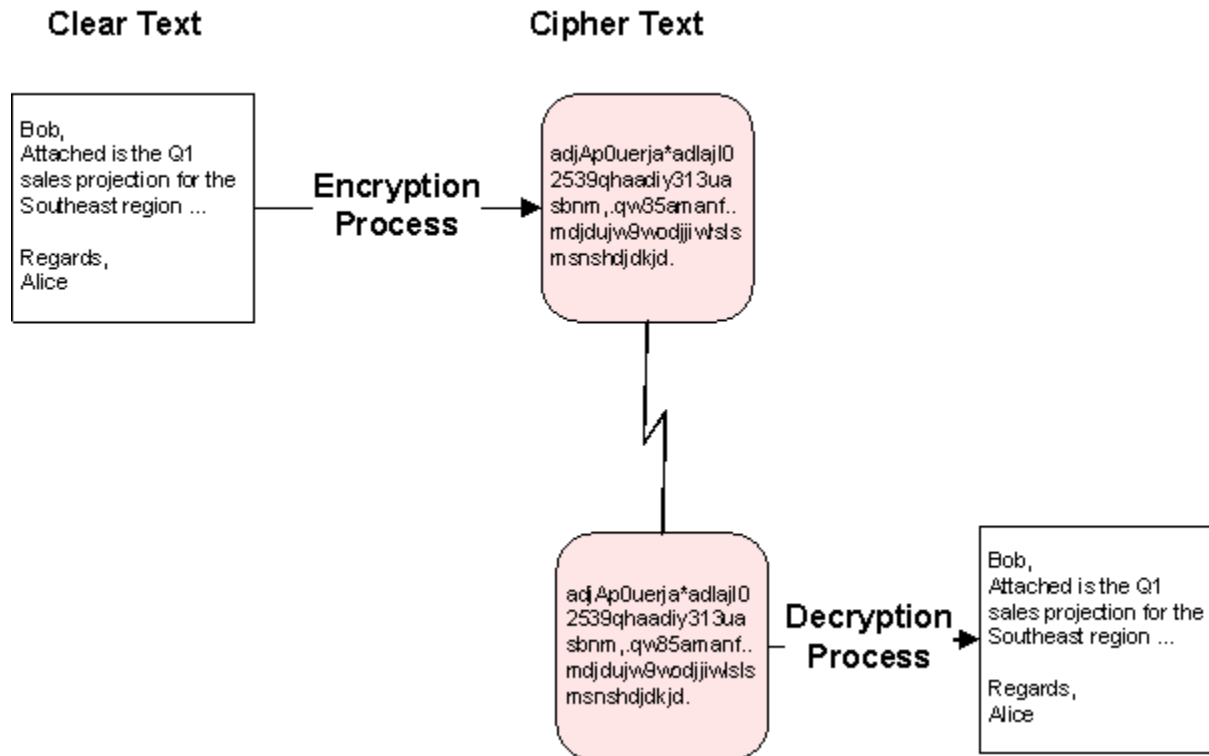



Figure 3.1: Encryption follows the same general set of steps regardless of the specific technique used.

Modern cryptography is based on algorithms that use keys, or sequences of bits that control how an encryption algorithm encodes a message. There are two categories of algorithms: symmetric and asymmetric cryptographic algorithms.


 Confidentiality of a message is ensured by protecting the secrecy of the key, not the algorithm. A long-held principle in security is that confidentiality should be based on keeping the key, but not the algorithm, secret. Publicly known algorithms are widely scrutinized and are less likely to contain subtle errors that could enable code cracking than algorithms that are not broadly analyzed.

Symmetric Key Cryptography

Symmetric algorithms use the same key to encode a message as well as decode it. This is analogous to a house key. Whether you are locking or unlocking a door, the same key is used. One of the advantages of symmetric key algorithms is that they are quite fast compared with asymmetric algorithms—sometimes up to 10 times faster. The chief drawback with symmetric key cryptography is that keys must be exchanged first before secure communications can occur.

For example, if Alice wanted to send encrypted emails to Bob using a symmetric key algorithm, she would have to send Bob the key. She could not send it through email in an unencrypted message because someone could intercept the message, steal the key, then decrypt any captured traffic between Alice and Bob that used the key. Instead, another channel, such as a physically transmitted message, would have to be used.

Another drawback is that if Alice wanted to send messages to Charles and did not want Bob to be able to read the messages, she would need a separate key to communicate with Charles. Clearly the number of keys one has to manage can grow rapidly and quickly reach the point of being impractically difficult to manage.

 In general, if a group of N different people need to communicate with encrypted messages, they will need a total of $N * (N-1)/2$ different keys; 5 people would need 10 keys, 20 people would need 190 keys, and 100 would need 4950 keys.

Although the speed of symmetric key cryptography is desirable, the key management problems limit its usefulness. Asymmetric key cryptography was developed to solve those problems.

Asymmetric Key Cryptography

Asymmetric key cryptography uses two keys instead of one. The first key is used to encrypt a message and the second is used to decrypt a message. If one tries to decrypt a message with the same key that encrypted it, the process will not work. It is like locking a door with one key but needing another key to open the door. As Figure 3.2 shows, the process of encoding a message is the same in both symmetric and asymmetric cryptography; the only difference is whether one or two distinct keys are used.

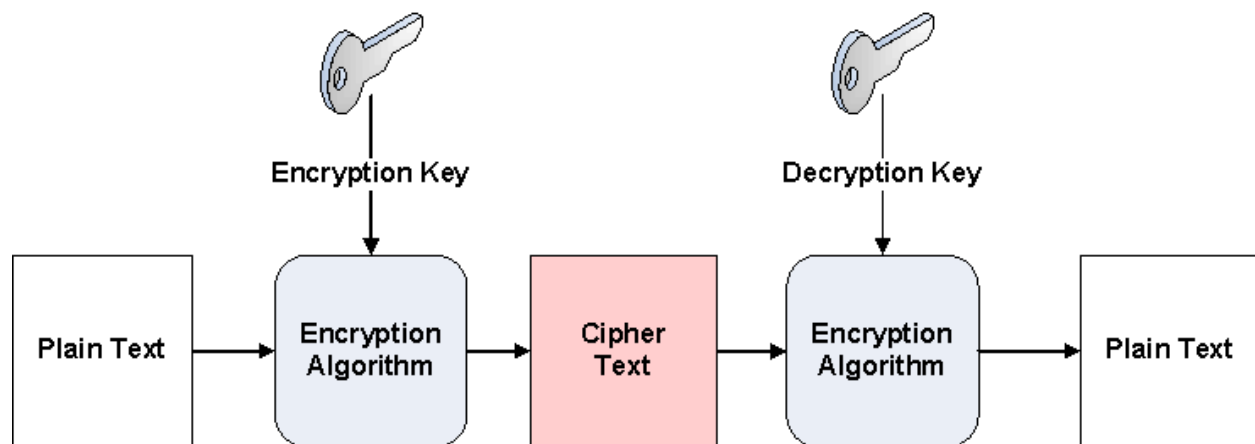


Figure 3.2: In symmetric key cryptography, the same key is used for both encryption and decryption.

This slight difference in the process has wide ranging effects on the ease of key management. Now, rather than generating a new key every time someone wants to securely communicate with someone else, a person can have one set of keys. The decryption key is kept private and used only by the person to decrypt messages sent to them. The second key is shared with anyone who might want to communicate with that person.

For example, suppose Alice, Bob, and Charles all want to communicate securely with each other. They each acquire a pair of encryption keys. One key is kept private and the other is made public. When Alice sends Bob a message, she uses Bob's public key to encrypt the message. Bob then uses his private key to decrypt the message. Since only Bob knows his private key, no one else can decrypt the message, not even Alice who just sent the message. Alice and Bob do not have to worry about Charles intercepting the message and decoding it because Charles only has access to Bob's public key, which cannot be used to decode the message.

In addition to keeping messages secure, asymmetric cryptography ensures non-repudiation. If a message is successfully decrypted using a person's private key, the message must have been encrypted using that same person's public key. Assuming the private key is secure and unknown to anyone else, only that person could have sent the message. Thus, the recipient can be sure of the origin of the message.

The major drawback of asymmetric key cryptography (also known as public key cryptography) is that it is much slower to encrypt and decrypt messages than doing so with symmetric key cryptography. However, using both symmetric and asymmetric key cryptography together, you can have the best of both worlds.

Encryption Algorithms

Some of the most well known and commonly used encryption algorithms include:

Symmetric Key Algorithms

- Data Encryption Standard (DES) was an early, widely adopted algorithm
- Triple-DES (3DES) was created to address vulnerabilities in DES
- International Data Encryption Algorithm (IDEA) is used in PGP encryption software
- Blowfish, a replacement for DES and IDEA, was designed by Bruce Schneier
- RC5 is a patented algorithm developed by RSA Data Security

Asymmetric Key Algorithms

- RSA is the most widely used asymmetric algorithm
- El Gamal is used for digital signatures and key exchanges
- Elliptic Curve algorithm is a very efficient asymmetric algorithm

Combining Symmetric and Asymmetric Key Cryptography

The strengths of the two cryptography types can be combined to solve the problem of securely sharing symmetric keys and enabling the higher performance of symmetric key systems. The procedure uses asymmetric cryptography to securely exchange a symmetric key, which is then used for the rest of the communication session.

The basic procedure is as follows:

- Alice initiates a communication channel with Bob and sends Bob her public key.
- Bob generates a symmetric key to use for this communication session, known as the session key.
- Bob encrypts the session key with Alice's public key and sends it to Alice.
- Alice decrypts the session key with her private key.
- Alice and Bob now use the session key with a symmetric key cryptography system to communicate.

This type of technique is used with Secure Sockets Layer (SSL) communications and works with other session-oriented communication patterns.

 For a detailed explanation of cryptographic algorithms, see Bruce Schneier's *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (Wiley, 1995).

Preserving Message Integrity

In addition to ensuring confidential messages remain private, parties to communication will want to ensure a message is not tampered with en route from the sender to the receiver. Two techniques, known as message authentication codes (MACs) and digital signatures, address this problem.

Consider the following scenario depicted in Figure 3.3. If Alice sends Bob a message stating a company's earnings are expected to grow 10 percent this year, and Charles intercepts the message and changes it to say the company's earnings will drop 5 percent this year, how would Bob know the message was altered?

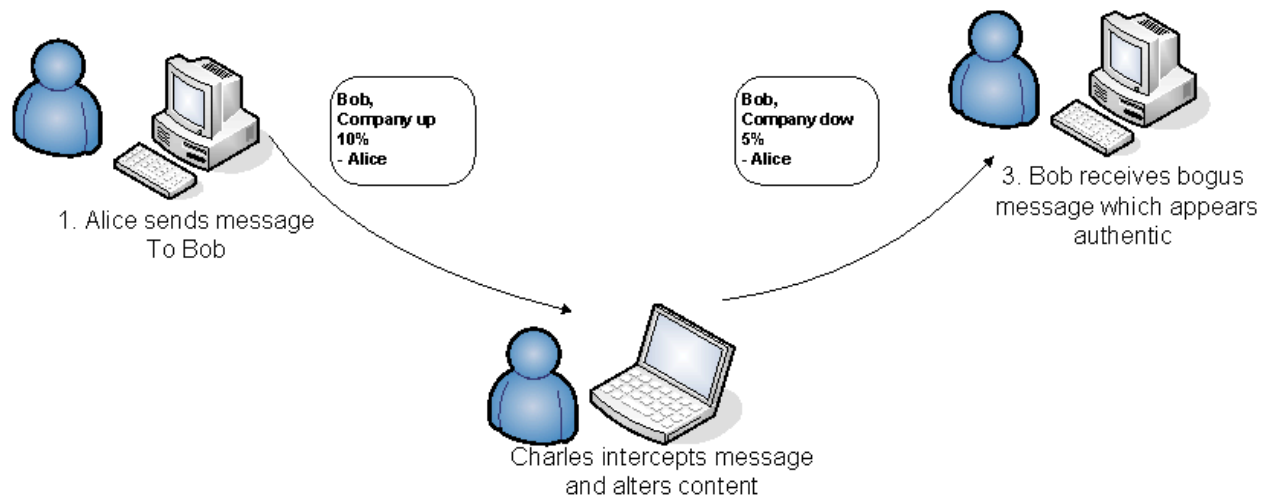


Figure 3.3: Man-in-the-middle attacks can alter the content without detection unless additional information, such as a message digest, is provided along with the message.

One way to prevent this type of a man-in-the-middle attack is to use a message digest calculated with a hashing algorithm. A message digest is a fixed-length string of characters that is generated by a mathematical function that takes a message as input. The input message can be of any length, but the message digest will always be the same length. As Table 3.1 shows, changing just a single character in a message will cause the calculated message digest to change.

Message	Message Digest
This is a sample text for a message digest example 1.	889e994466b1dd2ab5adeb62c5a2dd30
This is a sample text for a message digest example 2.	eee37f4ae076ed63e4c2adda6b7f2385

Table 3.1: A simple, single-character change in a message results in a completely different message digest.

 The message digests were calculated using the MD5 algorithm implemented in MD5Sums program by Jem Berkes, which is freely available at <http://www.pc-tools.net/win32/md5sums/>.

Of course, using this technique would be of little use if one could easily change a message in such a way as to generate the identical message digests. The algorithms used for message digests are designed to make it infeasible to detect two messages with the same message digests. (When this happens, it is known as a collision.) Another potential problem is that anyone intercepting the message could alter the message and calculate a new message digest; the recipient would have no way to tell the message had been altered.

MACs

One way to avoid undetected man-in-the-middle attacks is to use a MAC. A MAC uses a hashing function, but instead of using just the message as input, the MAC takes the message plus a symmetric key concatenated to the message.

Assuming Alice wants to send another message to Bob using a MAC, the steps are:

- Alice writes the message and concatenates a symmetric key.
- Alice calculates a message authentication code for the message plus a key using a hashing algorithm.
- The message authentication code is attached to the message and sent to Bob.
- Bob receives the message and concatenates the symmetric key.
- Bob generates a message authentication code and compares it with the one sent by Alice; if they are identical, the message has not been changed.

This technique assumes that Alice and Bob have securely exchanged a symmetric key and that only they have that key. MACs have similar problems to those presented by using symmetric keys to encrypt messages. Again, asymmetric cryptography addresses those problems.

Digital Signatures

To avoid the need to exchange symmetric keys, asymmetric cryptography can be used with a technique that is known as digital signatures. (A digital signature is an encrypted string of characters and should not be confused with a digital image of a person's signature.) The following process illustrates how Alice can send a message to Bob and ensure any changes could be detected:

- Alice writes a message and calculates a hash value for the message.
- The hash value is encrypted with Alice's private key to create a digital signature.
- Alice sends the message plus the digital signature to Bob.
- After receiving the message and digital signature, Bob calculates the hash value of the message.
- Bob decrypts the encrypted hash value Alice sent along with the message using Alice's public key.
- If the hash value Bob calculate matches the one sent by Alice, the message has not been changed.


If someone intercepted the message and altered the contents, the hash value generated by the receiver would not match the hash value sent along with the message. The only way to encrypt a new hash value that could be decrypted using Alice's public key would require Alice's private key. Any other encryption key would not decrypt properly. As long as Alice keeps her private key to herself, no one can impersonate her. This process provides for authentication as well as non-repudiation.

As Table 3.2 shows, by combining a number of cryptographic techniques, one can have a number of different aspects of secure communication.

Aspect of Secure Communication	Techniques
Confidentiality	Encryption
Integrity	Hashing functions
Integrity, non-repudiation, and authentication	Digital signatures
Integrity, non-repudiation, authentication, and confidentiality	Encryption and digital signatures

Table 3.2: Aspects of secure communications are realized with a variety of cryptographic techniques.

Cryptography provides several tools for protecting the confidentiality and integrity of data. But cryptography alone will not provide a secure information infrastructure. The systems that run cryptographic programs must be protected, access to unencrypted information must be controlled, and rules governing who performs operations on data and systems must be enforced. Access control systems have been created for these purposes and more.

 Bruce Schneier's *Secrets and Lies: Digital Security in a Networked World* (Wiley, 2004) provides an excellent discussion of the difficulties of applying security technologies, such as cryptography, in real-world systems.

Preserving Data Integrity by Controlling Access

Access controls are mechanisms put in place to ultimately ensure that only persons (or programs) allowed to view, change, and delete information are able to do so. The practice of access controls consists of four operations:

- Identifying users
- Proving that users are who they say they are, known as authentication
- Determining what actions a user is allowed to perform, known as authorization
- Recording information about operations performed, known as auditing

Each of these operations provides a critical part of the access control process.

Identifying Users

When someone wants to gain access to a system, they must identify themselves. Usernames are the most common method of identification. They are the most convenient and the least-expensive method to implement. They are vulnerable to disclosure, so secure environments use other techniques, either instead of usernames or in addition to usernames.

Other techniques for identifying users include biometrics and token devices. Biometrics use a physical feature of a person to identify them. These include fingerprints, hand geometry, retina scan, iris scan, voice print, facial scan, and handwriting dynamics. Tokens are devices that someone must possess to gain access to a system, such as a magnetic card that must be swiped through a reader. Identification is closely tied to authentication.

Authenticating Users

Authenticating users entails having them prove who they say they are. Weak authentication depends on passwords, which can be easily forgotten or disclosed. Strong authentication methods depend on two or more mechanisms, often referred to as something you know, something you have, or something you are.

For example, a bank may require strong authentication to access a wire transfer system. A user may have to provide a username and password as well as swipe a magnetic card or put a finger on a fingerprint reader to gain access to the system. Once a person has been identified and their identity is verified through authentication, the next step in access controls is to determine what the person is allowed to do.

User Authorizations

Authorizations dictate what type of access a user has to a resource, such as a file, a server, a database table, or an application. File access controls are probably the most familiar access control in an operating system (OS). For example, Figure 3.4 shows the basic file access controls on a Windows system.

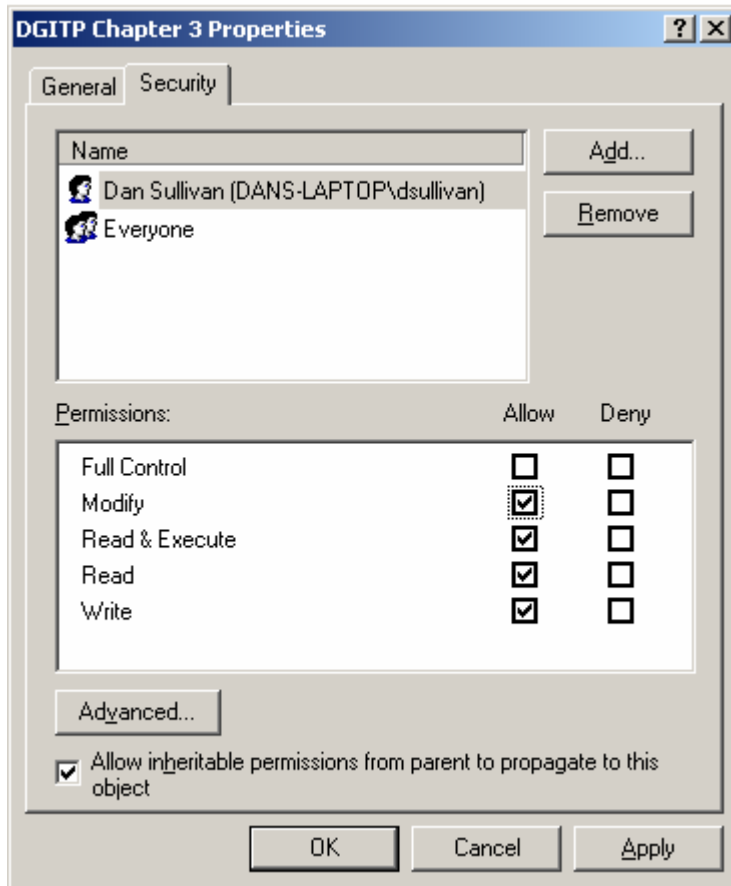


Figure 3.4: Basic file access controls on a Windows system.

Access control lists (ACLs) for a given user and a given object, such as a file, define what operations can be performed. The minimal set of operations listed in Figure 3.4 is often enough to specify appropriate control levels. In some cases, finer-grained access controls are required. In the case of Windows OSs, access controls can be specified using the advanced access controls shown in Figure 3.5.

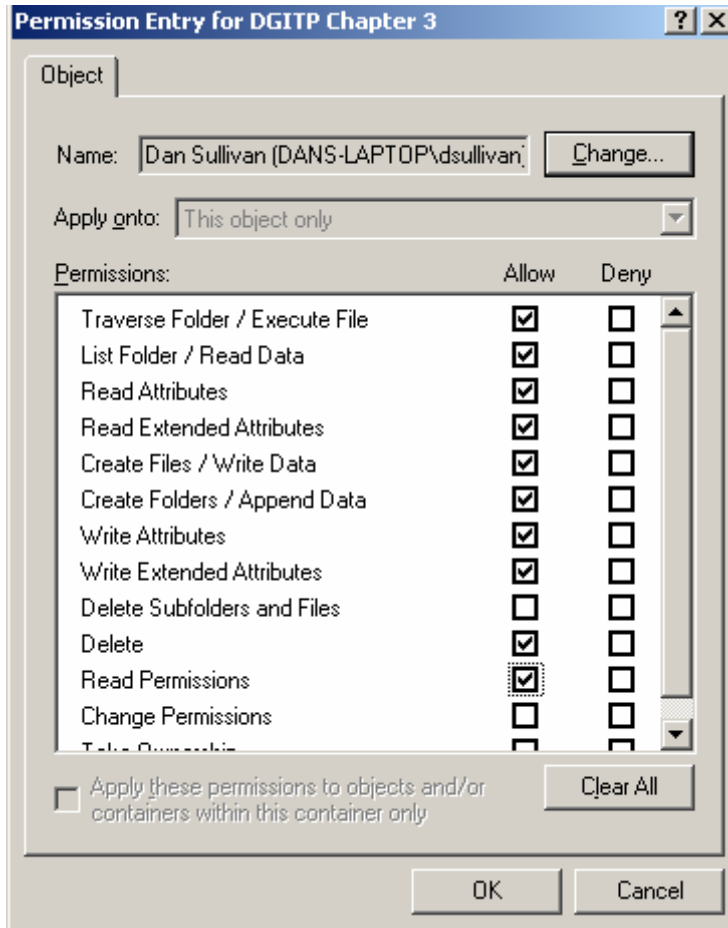


Figure 3.5: Advanced access controls provide greater control over objects.

Specifying fine-grained and even basic access controls for every file and every individual user can quickly become a tedious and time-consuming task. Access controls are typically managed in groups based on one or more attributes:

- Roles in an organization—such as job function
- Groups—such as project teams that are made up of individuals with different roles but need access to the same resources
- Location—for example, only employees in the Boston office will have access to file servers in that office
- Time of day—for example, accounts payables clerks can issue checks to vendors but only during normal business hours
- Transaction types—such as allowing a user to delete some records in a database but not others

A best practice for access controls is known as the principle of least privilege; that is, a person is granted the minimum amount of access to resources needed to accomplish his or her job. Even when administrators have implemented appropriate access controls and authentication mechanisms, it is prudent to monitor activity within a system.

Auditing

Auditing is the process of recording information about events on a system and reviewing that information. For example, a systems administrator may want to know of repeated login failures on a single account—information that might indicate someone trying to break in to the system. In other cases, application managers might want to know anytime a particular type of transaction record is deleted or whether—and when—a transaction over a specified amount is entered. The goal of auditing is not to know the detail of every event, that would be overwhelming, but to know when unusual or suspicious events occur.

Audit information is also essential to forensic investigations. If a system is breached, security professionals can use audit trails to trace the steps used by the intruder to infiltrate the system and perform functions on the compromised system. Of course, the integrity of the audit information must be protected from tampering if it is to be of any use in forensic investigations.

Access controls are a key technology for preserving the integrity of information and systems. Again, like cryptographic systems used to preserve the confidentiality of information, access controls depend on secure and reliable networks and servers, which, in turn, depend upon infrastructure protection techniques.

Maintaining Availability and Integrity by Protecting the Infrastructure

IT infrastructure includes several different technologies, each with its own security requirements. The key elements of IT infrastructure addressed here include:

- Network security
- Systems security
- Application security

These three areas are like pillars upon which IT operations depend, and the failure of any one of these can result in disruptions for a range of IT operations.

Maintaining a Secure Network

Networks are the pathways for sharing information within and outside of organizations; they are also the pathways for malicious software, intruders, and other threats. With the increasing demands for regulatory compliance and the emergence of case law governing hostile workplace environments, the ability to control what content comes into an organization is of growing importance. Simply controlling traffic into and out of a network with a firewall is no longer sufficient to secure network services; at least four technologies are required:

- Firewalls
- Intrusion prevention systems (IPSs)
- Content-filtering systems
- Antivirus/spyware/spam systems

Together, these technologies address the major threats that directly impact network operations or use networks to carry out malicious operations on other systems.

Perimeter Defenses: Firewalls

There are several types of firewalls, but they all serve the same basic purpose: separating traffic on different segments of a network and limiting traffic across the segments. A common use of firewalls is to set up three network segments: an internal trusted zone, a semi-trusted zone—commonly referred to as a demilitarized zone (DMZ), and the external network, typically the Internet (see Figure 3.6).

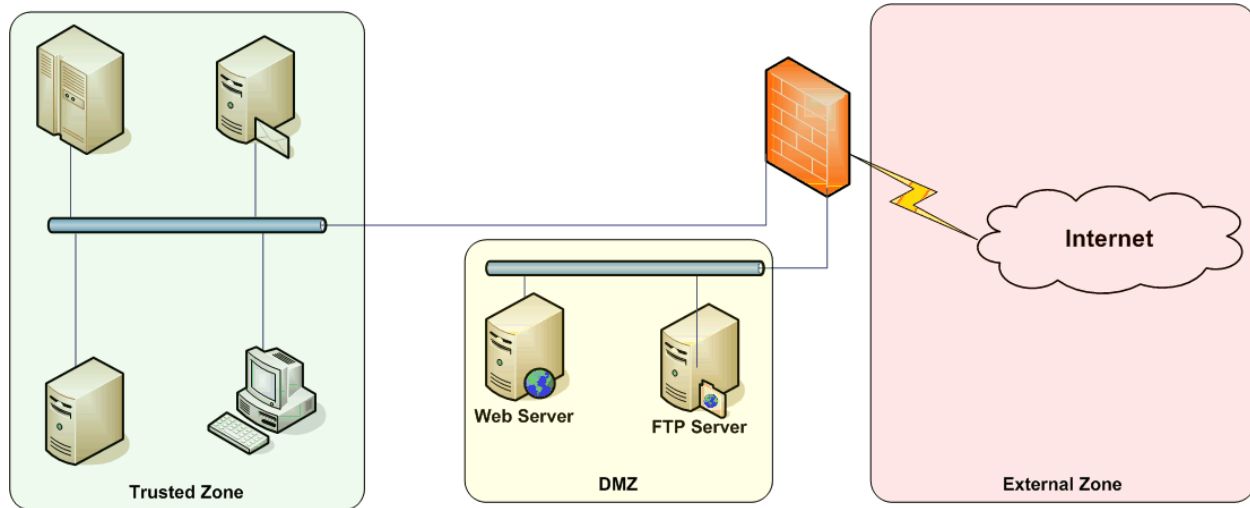


Figure 3.6: Firewalls allow for segmented networks with controlled access between segments.

By partitioning areas of the network, administrators can balance the need to exchange information with outside entities with the requirement to protect internal assets. For example, critical resources, such as mainframes and file servers, could be located in the trusted zone, inaccessible to users in the external zone. Services for external users, such as Web servers and file transfer servers, could be located in the DMZ. Only limited types of traffic are allowed between the trusted zone and the DMZ; similarly, traffic between the DMZ and the external zone would be limited to necessary protocols such as HTTP for Web servers, FTP for file transfer, and SMTP for email services. Other protocols—such as the network management protocols like SNMP—might not be allowed into the DMZ from the external network. How firewalls control access depends on the type of firewall.

The main types of firewalls are:

- Packet filtering firewalls use rules or ACLs to limit traffic; for example, denying traffic from some IP addresses or blocking all ping requests
- Stateful inspection firewalls keep information about traffic that has gone over a connection and can make decisions based on past transmissions and not just by looking at network packets as they come in
- Proxy firewalls act as a middleman, hiding internal resources from external services, and act as an agent for internal users and applications

Even with firewalls in place, it is possible that the network will be breached. When that occurs, IPSs can provide a second line of defense.

IPSs

IPSs—and their predecessors, intrusion detection systems (IDSs)—monitor networks or individual systems looking for signs of an attack or a breach. In addition to monitoring, IPSs can respond to attacks. IPSs are either network or host based.

Network-Based IPSs

A network-based IPS monitors traffic on a network looking for particular patterns that indicate an attack. For example, a simple Denial of Service (DoS) attack floods a server with SYN packets used in the TCP handshake to establish a communications session. The SYN packets use fake sender IP addresses so that the attacked hosts respond and wait for an acknowledgment that never comes. Eventually, the host will stop waiting, but in the meantime, resources are tied up. When enough of these bogus SYN packets are sent to a host, so many resources are tied up that legitimate communications cannot be established.

Network-based IPSs deploy sensors throughout the network to monitor traffic. Some sensors may be placed just inside the firewall in the DMZ and others inside the trusted zone to monitor the distinct traffic patterns in different segments. Networks are one target of attacks; individual workstations and servers are other prime targets.

Host-Based IPSs

A host-based IPS is a monitoring program that runs on individual servers, workstations, or desktops looking for unexpected changes to the system. These programs can analyze:

- System logs
- File hash value
- Changes to registries
- Changes to file access controls

The objective of a host-based IPS is to detect unauthorized changes. For example, if an intruder broke into a system and tampered with a data file, the file size and modification data would normally change, but an intruder could easily hide those changes. A better option is to calculate a hash value (described earlier); it would then be virtually impossible to change a file and keep the same hash value. The host-based IPS would calculate hash values for critical files, such as those in a WINNT directory of a Windows system or a /bin directory on UNIX and related OSs.

Both host-based and network-based IPSs have means of notifying administrators of significant events, generating reports about events, and analyzing data gathered in the monitoring process. Detection is obviously a fundamental function in IPSs and there are two general approaches to this: signature-based detection and behavior-based detection.

Signature-Based Detection

Signature-based detection uses patterns to describe and detect known attacks. For example, a large number of SYN packets can indicate a DoS attack. This technique works well with known attacks that are easily distinguished from legitimate traffic. It does not work well with new attacks unless vendors can develop new patterns quickly and deploy them to production sites. Unless signatures are carefully crafted, they can match patterns of traffic that are not attacks but legitimate traffic. This problem, known as false positives, is a significant challenge for IPS applications. An alternative to signature-based detection is behavior-based detection.

Behavior-Based Detection

Behavior-based detection, also known as statistical detection, uses patterns of activity in a specific network to determine whether unusual activity might constitute an intrusion attempt. The first step in behavior-based detection is building a profile of “normal” network activity. This profile will obviously change throughout a single day, week, and month. For example, there may be moderately heavy network traffic early in the day when users first log in, check and send email, open files, start applications, and so on. There may also be peak periods in the middle of the night when batch processes are transmitting files, for example, to a data warehouse. If a centralized database replicates to satellite databases once a week, there will be weekly spikes that the IPS must take into account. Once profiles are built, the IPS can identify any significant variations from the expected patterns.

As with signature-based detection, behavior-based IPSs can generate false positives if the pattern of usage changes and profiles are not updated. They can also occur if profiles are not developed with enough data to get an accurate assessment of network traffic. Of course, the concept of profiling assumes that past behavior is a good indicator of future behavior; that is true in some situations more than others. Although IPSs can detect and stop some attacks, there are times when the content that is transmitted is the problem.

Content Filtering and Anti-Malware Systems

Firewalls and network-based IPSs operate at the network level monitoring types of traffic and checking for patterns indicative of an attack. Threats are not limited to the transmission medium of the network, some threats are carried by the network into servers, desktops, mobile clients, and other devices connected to the network. Before going any further, it is worth noting what constitutes a “threat” today and how the definition has changed.

Evolving Nature of Threats

Broadly speaking, a threat is anything that could compromise the confidentiality, integrity, or availability of information systems. These are general terms and many IT professionals might think of viruses, worms, Trojan horses, DoS attacks, and related problems facing security and systems administrators. These are certainly major threats, but the advent of security and privacy regulations and employment case law, particularly in the area of harassment, has expanded the concept of an IT threat.

Today, an employee who downloads offensive material or emails crude, inappropriate jokes to coworkers could be found to be creating or contributing to a hostile work environment. This is a threat to the integrity of an organization as a whole and, although not strictly part of information security, technology can play a significant role in these cases and therefore computers and networks should be managed to reduce the potential of creating a hostile workplace.

 For an overview of hostile workplace issues see Toni Bowers “Hostile Work Environment: A Manager’s Legal Liability” at <http://builder.com.com/5100-6404-5035282.html>.

Regulations are also defining new threats to an organization. Take the Health Insurance Portability and Accountability Act (HIPAA), for example, which defines protected health information and defines limits on how that information is used, stored, and shared. Today, if protected health information is inadvertently disclosed—for example, a patient’s clinical record file is attached to an email and sent outside the organization—the offending company could be liable for sizeable penalties.

IT threats, broadly understood, now include malicious content sent from outside the organization as well as inappropriate content brought in and confidential information that is sent out of the organization. Anti-malware addresses the first area, content filtering the second.

Anti-Malware: Protecting Against Malicious Software

Malware is a general term used to describe malicious software such as viruses and spyware. Ten years ago, systems administrators and network managers were primarily concerned about viruses and worms. (Viruses are malicious programs that depend on another program to propagate; worms can spread on their own by exploiting vulnerabilities in applications). The breadth of malware has expanded and now includes:

- A wide variety of viruses, including viruses that use encryption of pattern-changing techniques (polymorphic viruses) to avoid detection
- Macro-viruses that exploit weaknesses in applications such as desktop productivity applications and email clients
- Worms that target unpatched vulnerabilities in widely used applications, such as SQL Slammer
- Keyloggers, which capture keystrokes, filter for confidential data,—such as usernames and passwords, and forwards data to the perpetrator’s server
- Frame-grabbers that capture the content of the display every few minutes or more and send the content to a server controlled by the perpetrator
- Bot clients that listen for instructions from a perpetrator to perform some unauthorized function, such as send spam from the compromised machine

The methods to detect malware are similar to those used in IPSs: signature-based and behavior-based detection.

Scanning for Patterns: Signature-Based Malware Detection

Antivirus programs were initially designed to use signatures, or patterns, to detect viruses. This design worked well initially because viruses are programs that follow a specific sequence of instructions, and those instructions translate into fixed bit-patterns that are transmitted to the target system.

Virus writers tried to avoid detection by encrypting their programs, but antivirus developers simply switched to looking for decryption modules (which is required to decode the virus before it can execute). In the seemingly continuous cat-and-mouse pursuit between virus writers and antivirus developers, the virus writers made a major leap by developing what are now known as polymorphic viruses. Polymorphic viruses contain the usual code for propagating the malicious payload (which takes actions such as deleting files or corrupting data) as well as a specialized module for changing the code of the virus without actually changing the program's behavior (see Figure 3.7).

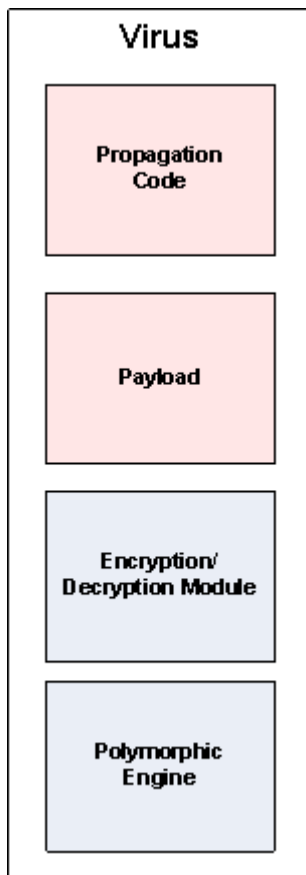


Figure 3.7: All viruses contain propagation code and a payload; newer viruses use encryption and self-modifying code to avoid detection by signature-based methods.

The polymorphic engine changes the sequence of instructions by introducing junk instructions, such as $\text{Variable1} = \text{Variable1} + 0$, which do not change the final effect of the program. Every time the virus replicates it uses different junk instructions, so no single signature will detect the virus. Another method is needed.

Behavior-Based Malware Detection

With behavior-based malware detection, suspicious code is executed within a virtual machine created by an antivirus program. In this way, the antivirus program can look for telltale signs of a virus, such as low-level system calls, without actually exposing the computer to the virus.

This new detection technique is a significant advance over signature-based techniques, but it is not foolproof. The virtual machine can only simulate a virus execution for a limited period of time before having to make a decision as to whether the program is a virus.

Malware detection systems are available for both single hosts and for networks. Following the defense-in-depth strategy of information security, it is a best practice to use malware detection at both the desktop and on the network.

Beyond Malicious Programs: Filtering Inappropriate Content

The content that flows in and out of an organization can create a number of problems, including:

- Hostile work environments, if employees are disseminating offensive material
- Wasted time, if network users are spending time on shopping, gambling, and other sites unrelated to work
- Potential avenues for introducing malware when popular sites, such as music download sites, are used to spread malware, especially Trojan horses
- Lost intellectual property in the case of employees intentionally or unintentionally transmitting confidential information outside of established practices

These problems are addressed, in part, with content-filtering systems. The most basic form of content filtering is to identify and block the Web addresses, URLs, of problematic sites. As sites are easily changed, the lists of offensive sites must be updated continuously. Most content-filtering vendors provide regular updates to these lists, commonly known as *black lists* (*white lists* are sets of URLs that are always allowed to transmit content into an organization).

 Many realtime black lists are maintained for public use. See <http://directory.google.com/Top/Computers/Internet/Abuse/Spam/Blacklists/> for more information.

Content filtering is also used for email to block spam and phishing messages. In these systems, patterns within the content—such as telltale phrases, formatting tricks, and bogus URLs that look like legitimate Web site addresses—are used to classify bulk and suspicious emails.

All the security countermeasures described so far, including firewalls, IPSs, content-filtering and anti-malware systems, depend upon secure reliable systems to operate. Securing those systems is another fundamental challenge facing systems administrators.

Maintaining Secure Servers, Desktops, and Other Devices

Secure servers, desktops, and other devices depend on secure OSs and applications. These, however, are complex systems in themselves and are likely to contain bugs and security vulnerabilities that can be exploited to breach a system. One of the worst malware attacks, the SQL Slammer Worm, exploited a vulnerability in Microsoft SQL Server; although Microsoft had corrected the problem months before the attack, many database and systems administrators had not installed the corrected code. To minimize the chance of falling victim to attacks leveraging OS or application vulnerabilities, adopt three practices:

- Patch management
- Vulnerability testing
- OS hardening

These practices are part of well-developed defense-in-depth postures.



In addition to vendors, vulnerability information is available from third parties. The National Vulnerability Database at <http://nvd.nist.gov/> and Open Source Vulnerability Database at <http://www.osvdb.org/> are two examples of publicly available databases of system and application vulnerabilities.

The Role of Patch Management in Systems Security

Patches are updates to OSs and applications provided by vendors outside of the normal product release life cycle. Unlike upgrades, patches do not typically introduce new functionality but correct problems with current implementations of the systems. In some cases, patch management is a relatively straightforward process. For example, home users can easily update their Windows desktops using a Microsoft-provided utility. At the enterprise level, though, things become more complex.

Any organization with an IT infrastructure will have dependencies between systems. Some applications will depend on a particular version of the OS, other packages will need to run with a certain database version, and some custom scripts and utilities may be designed for legacy systems that continue to work and no one wants to change for fear of breaking them.

Prior to installing patches in production systems, patches should be installed in test environments that mirror the configuration of the production environment. In addition, systems administrators should have rollback plans to ensure that if something does go wrong, the original state of the system can be restored.

One question faced by systems administrators with regard to patches is, which patches should be applied? That depends in part on what, if any, vulnerabilities exist in the current configuration of the system. Vulnerability testing can help answer this question.

Vulnerability Testing

Vulnerability testing applications can probe OSs and applications looking for known vulnerabilities. One of the earliest vulnerability scanners, known as SATAN, generated a good deal of controversy because, it was argued, hackers could use the tool to find vulnerabilities and compromise systems. This is true but it is also true that hackers have a variety of tools at their disposal anyway and one more would make only marginal difference in a determined hacker's ability to breach a system. Ultimately, the school of thought that argued that system managers should at least have equivalent tools as the hacker to protect themselves prevailed. Vulnerability testing is now a well-accepted practice.

Like patch management, vulnerability scanning can be relatively simple for home users and small businesses with tool such as the Microsoft Baseline Security Analyzer (see Figure 3.8). When moving to the enterprise scale, vulnerability scanning requires planning about resource demands on the network and servers, how to prioritize findings, developing remediation plans, and other tasks associated with updating OSs and applications.

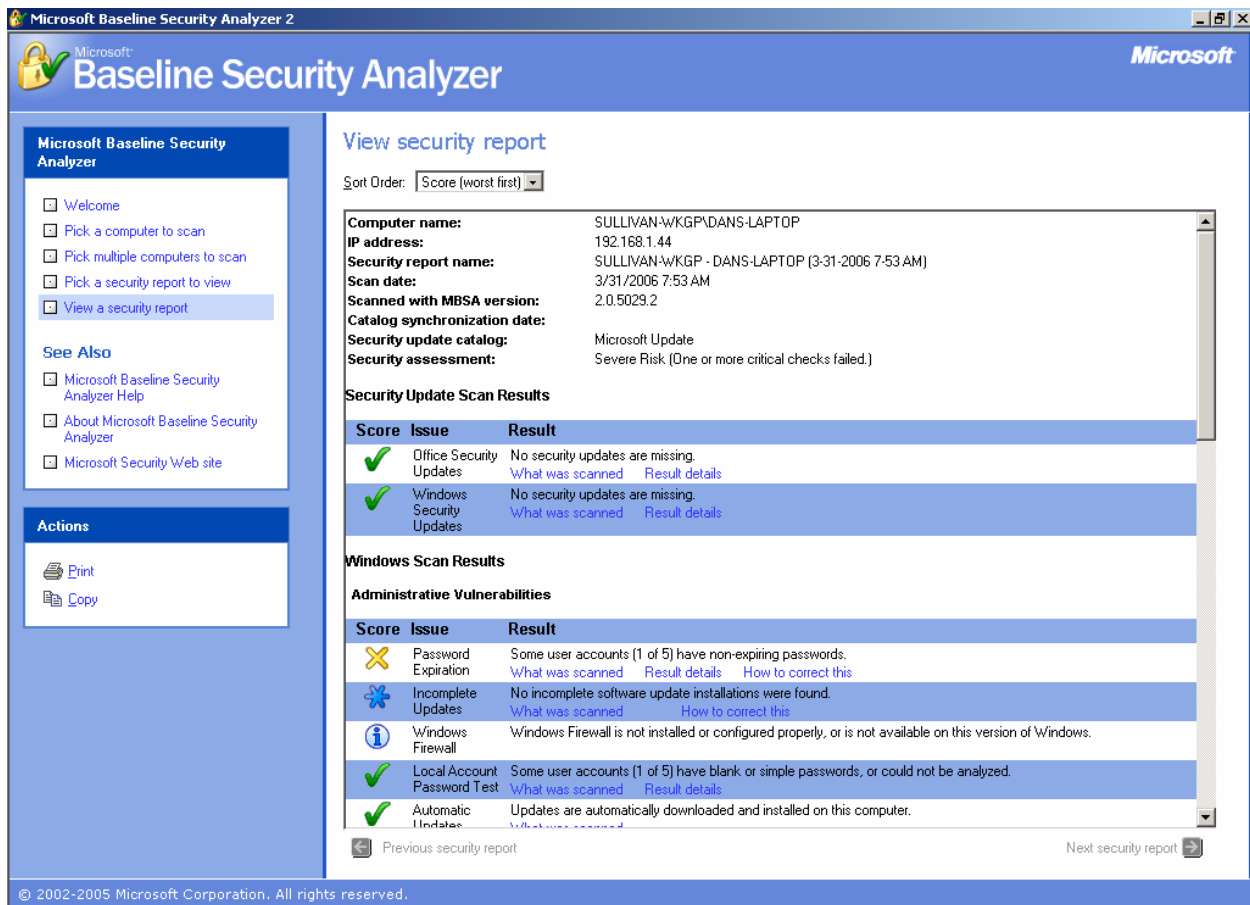



Figure 3.8: The Microsoft Baseline Security Analyzer scans for both OS and application vulnerabilities.

The Microsoft Baseline Security Analyzer is freely available at <http://www.microsoft.com/technet/security/tools/mbsa2/default.aspx>.

Hardening OSs

Default installations of OSs are designed to get systems up and running as quickly and as efficiently as possible. Since needs vary so widely, these default installations often start services that may not be needed (for example, an FTP server). In the best case, these extra applications just occupy space on a disk and consume an insignificant amount of CPU time; in the worst case, they introduce vulnerabilities without providing any benefit. Analyzing installed applications and shutting down any unnecessary services is known as hardening the OS. This should be done for all servers, but it is especially critical for servers that reside in a DMZ and are subject to vulnerability scanning from outside the organization.

 The Bastille Project has developed tools to help harden Linux, HP-UX, and Mac OS X. See <http://www.bastille-linux.org/> for more information.

Securing Applications

Another area of system security that is easily overlooked is application security. When large packages such as ERP systems are purchased, you have to depend upon vendors to ensure the systems are developed using sound software engineering practices. In the case of custom applications, software developers should follow sound security practices including code reviews and database security reviews. Projects should also be given the time and resources to write secure code.

 For more information about securing applications, see Matthew Schwartz's "Building Better Applications: Beyond Secure Coding" <http://www.adtmag.com/article.aspx?id=18205>.

Securing network perimeters, blocking malware, filtering inappropriate content, and locking down servers are demanding enough tasks to keep even the most seasoned systems administrators challenged; unfortunately, those may be some of the easier challenges.

Managing Client Security

Maintaining secure clients has always been difficult. Users often want administrative privileges to install applications and browser plug-ins or make other configuration changes. That is heresy to systems administrators and Help desk support staff. When users have administrative privileges, systems will vary from standard configurations, Help desk staff will be expected to assist with unsupported applications, and troubleshooting becomes more complex. In many organizations, clients are simply locked down and administered only by IT staff.

With centralized control, IT departments can implement common client security measures:

- Antivirus
- Anti-spyware
- Patch management
- Local vulnerability scanning

These can be deployed in standard ways across the organization and managed effectively.

The widespread adoption of Web-based applications and the blurring of network perimeters has introduced a new security challenge: unmanaged devices. Essentially, an unmanaged device is a client that is used to access an application or service but not managed by the IT group that supports that application or service. This situation creates a host of problems from a security perspective:

- Lack of administrator rights to install software, change the registry, and so on
- Potentially inadequate or improperly implemented security measures (for example, outdated antivirus signatures) on the unmanaged device
- No ability to fully scan for vulnerabilities or malware

In spite of these serious concerns, application developers and network administrators are expected to allow these unmanaged devices access to Web-based applications that may depend on core applications, such as order processing systems, billing systems, and other operational databases.

Traditional security measures for securing clients are no longer adequate. This reality has prompted the development of techniques that center around securing communications sessions rather than entire devices. Session-specific security is more appropriate for unmanaged devices for several reasons:

- Security mechanisms can be downloaded on demand—there is no permanent changes to device
- Security is tightly coupled to information, not the devices containing the information
- Those that control the information can now control the security of that information; this shifts responsibility away from users

The advent of this type of security on-demand is an important addition to the more traditional security measures discussed in this chapter and will be the focus of much of the remainder of this guide.

Controlling security on unmanaged devices is the third of three distinct areas in which particular security techniques are required. The first area, within a managed network, firewalls, IPSs, content filters, and related measures are used to preserve the integrity of infrastructure and data within a controlled network.

The second area, when data is in transmission over public networks, requires the application of encryption technologies to ensure confidentiality and integrity of information as well as to provide non-repudiation of messages. Protecting information in transmission is the subject of Chapter 4.

Once information is delivered to its destination, the need for securing that information does not stop. Spyware, browser vulnerabilities, and other weaknesses on unmanaged client devices can leave confidential information at risk. On-demand security techniques address the needs of protecting information that is no longer either in a controlled realm, such as an internal network, or protected by encryption, such as when transmitted over a public network. Chapter 5 describes in detail techniques available today to protect information that must reside on unmanaged and potentially compromised devices.

Summary

There are three key goals of information security: confidentiality of information, integrity of information, and availability of systems. To meet these objectives requires a host of security techniques, systems, and practices, ranging from encryption and access controls to network scanners and session-oriented security measures. Maintaining a secure environment now requires a balanced mix of older well-established technologies, such as firewalls, and new security on-demand measures.

Download Additional eBooks from Realtime Nexus!

Realtime Nexus—The Digital Library provides world-class expert resources that IT professionals depend on to learn about the newest technologies. If you found this eBook to be informative, we encourage you to download more of our industry-leading technology eBooks and video guides at Realtime Nexus. Please visit <http://nexus.realtimepublishers.com>.