computer-to-computer links." Fulghum et al. [FUL07b] refer to an analyst describing spoofs of the Syrian command and control capability, done through a network attack. Fulghum [FUL07c] then described a technology likely used in this attack: "The technology allows users to invade communications networks, see what enemy sensors see and even take over as systems administrator so sensors can be manipulated into positions so that approaching aircraft can't be seen, they say. The process involves locating enemy emitters with great precision and then directing data streams into them that can include false targets and misleading messages [and] algorithms that allow a number of activities including control."

In short, not only did the Israelis presumably intercept or block signals, but they also inserted signals of their own into the air defense network. Envision an air defense screen that shows an empty sky while enemy jets are racing through the air.

THREAT: MAN IN THE MIDDLE

The threat described in this attack is called a **man-in-the-middle** attack, sometimes denoted **MitM**, which is an active wiretapping technique in which the attacker catches and replaces a communication between two endpoints without either endpoint knowing the transmission is modified.

It is easiest to explain a man-in-the-middle attack in terms of a protocol for network communication. Suppose two parties, Amy and Bill, want to communicate. As we just showed in Chapter 11, asymmetric cryptography can be a way they can exchange cryptographic keys securely. Basically, the key exchange protocol, depicted in Figure 12-1, would work like this:

1. Amy says: Bill, please send me your public key.
2. Bill replies: Here, Amy; this is my public key.
3. Amy responds: Thanks. I have generated a symmetric key for us to use for this interchange. I am sending you the symmetric key encrypted under your public key.

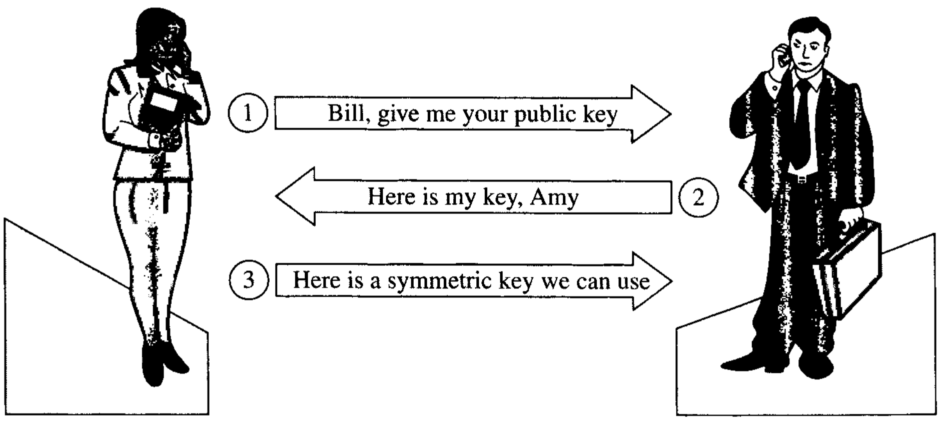
1
Bill, give me your public key
Here is my key, Amy
2
3
Here is a symmetric key we can use

FIGURE 12-1 Key Exchange Protocol

Digitised by the Department of Library Services, University of Pretoria, 2019

In a man-in-the-middle attack, as shown in Figure 12-2, we insert the attacker, Malvolio, into this communication.

1. Amy says: Bill, please send me your public key.

1a. Malvolio intercepts the message and fashions a new message to Bill, purporting to come from Amy but with Malvolio's return address.

2. Bill replies: Here, Amy; this is my public key. (Because of the return address in step 1a, this reply goes to Malvolio.)

2a. Malvolio holds Bill's public key and sends Malvolio's own public key to Amy, alleging it is from Bill.

3. Amy responds: Thanks. I have generated a symmetric key for us to use for this interchange. I am sending you the symmetric key encrypted under your public key.

3a. Malvolio intercepts this message and obtains and holds the symmetric key Amy has generated.

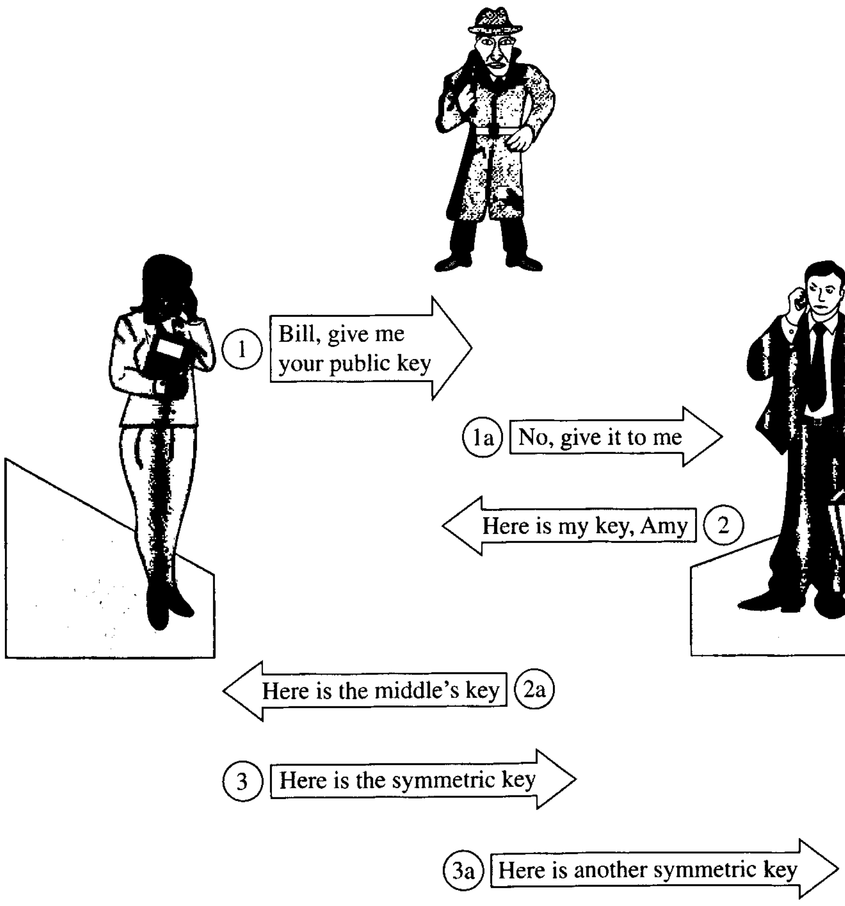
1
Bill, give me your public key
1a
No, give it to me
Here is my key, Amy
2
Here is the middle's key
2a
3
Here is the symmetric key
3a
Here is another symmetric key

FIGURE 12-2 Key Exchange Protocol with a Man in the Middle

Digitised by the Department of Library Services, University of Pretoria, 2019

Aspidistra, a WW II Man in the Middle — Sidebar 12-1

During World War II Britain used a man-in-the-middle attack to delude German pilots and civilians. Aspidistra, the name of a common houseplant also known as cast-iron plant for its seeming ability to live forever, was also the name given to a giant radio transmitter the British War Office bought from RCA in 1942. The transmitter broadcast at 500 kW of power, ten times the power allowed to any U.S. station at the time, which meant Aspidistra was able to transmit signals from Britain into Germany.

Part of the operation of Aspidistra was to delude German pilots by broadcasting spurious directions (land, go here, turn around). Although the pilots also received valid flight instructions from their own controllers, this additional chatter confused them and could result in unnecessary flight and lost time. This part of the attack was only an impersonation attack.

Certain German radio stations in target areas were turned off to prevent their being beacons by which Allied aircraft could home in on the signal; bombers would follow the signal and destroy the antenna and its nearby transmitter if the stations broadcast continually. When a station was turned off, the British immediately went on the air using Aspidistra on the same frequency as the station the Germans just shut down. They copied and rebroadcast a program from another German station, but they interspersed propaganda messages that could demoralize German citizens and weaken support for the war effort.

The Germans tried to counter the phony broadcasts by advising listeners that the enemy was transmitting and advising the audience to listen for the official German broadcast announcement—which, of course, the British duly copied and broadcast themselves. (More details and pictures are at http://www.qsl.net/g0crw/Special%20Events/Aspidistra2.htm, and http://bobrowen.com/nymas/radioproppaper.pdf.)

3b. Malvolio generates a new symmetric key and sends it to Bill, with a message purportedly from Amy: Thanks. I have generated a symmetric key for us to use for this interchange. I am sending you the symmetric key encrypted under your public key.

In summary, Malvolio now holds two symmetric encryption keys, one each shared with Amy and Bill. Not only can Malvolio stealthily obtain all their interchanges, but Amy and Bill cannot communicate securely with each other because neither shares a key with the other.

From this point on, all communications pass through Malvolio. Having both symmetric keys, Malvolio can decrypt anything received, modify it, encrypt it under the other key, and transmit the modified version to the other party. Neither Amy nor Bill is aware of the switch.

Man-in-the-middle attacks can occur in protocols (both network and cryptographic), network communications, activity between applications, and even on plain web pages. We show an example of a real-life man-in-the-middle attack in Sidebar 12-1. The basis of the attack is the same in all contexts, however: One entity intrudes in an exchange between two parties and pretends to be the other party in interactions with each of the two sides. We explore these different situations as a way to present the full breadth of this kind of attack.

Digitised by the Department of Library Services, University of Pretoria, 2019

THREAT: "IN-THE-MIDDLE" ACTIVITY

In the air defense attack, the man in the middle intercepts, modifies, and forwards images, not messages. A similar attack is possible with any interaction between two parties. It can occur between two people, two communication points, two computers, two applications, or a data device (for example, a keyboard, printer, or disk) and an application.

The keystroke logger we described in Chapter 5 is a limited form of a man-in-the-middle attack; in that case, the logger was passive, merely intercepting data without modifying it.

A classic man-in-the-middle attack involves active wiretapping: The intruder has to intercept all communication between the two legitimate parties; the intruder filters or rewrites traffic for his benefit. In the next sections we describe some other examples of man-in-the-middle attacks.

DNS Spoofing

At the heart of Internet addressing is a protocol called **DNS** or **Domain Name System** protocol. DNS is the database of translations of Internet names to addresses, and the DNS protocol resolves the name to an address. For efficiency, a DNS server builds a cache of recently used domain names; with an attack called DNS poisoning, attackers try to insert inaccurate entries into that cache so that future requests are redirected to an address the attacker has chosen. We consider DNS poisoning more in Chapter 14.

A standard DNS query and response is shown in Figure 12-3, in which the user requests a translation of the URL microsoft.com, and the name server responds with the address 207.46.197.32.

DNS service is implemented on a remote server, so a man-in-the-middle attack involves the attacker's intercepting and replying to a query before the real DNS server can respond. Such a situation, called **DNS spoofing**, is shown in Figure 12-4. In that example, the attacker quickly responds with address 7.0.1.1 (presumably an address over which the attacker has control). With that change the attacker can enter into the middle of the user's communication with www.microsoft.com, forwarding whatever the attacker wants to the real Microsoft web site. The user's browser disregards the correct response from the DNS server that arrives after the browser has already accepted the false address from the attacker.

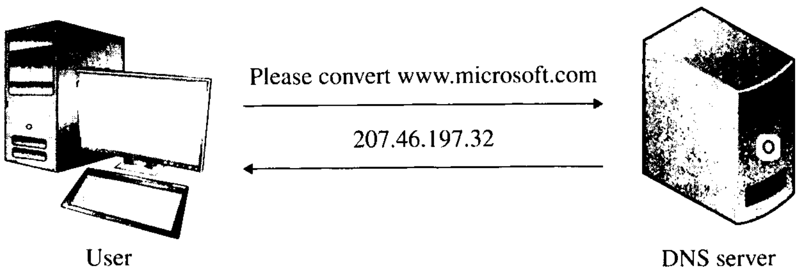

FIGURE 12-3 Resolving a Domain Name to an Address

Digitised by the Department of Library Services, University of Pretoria, 2019

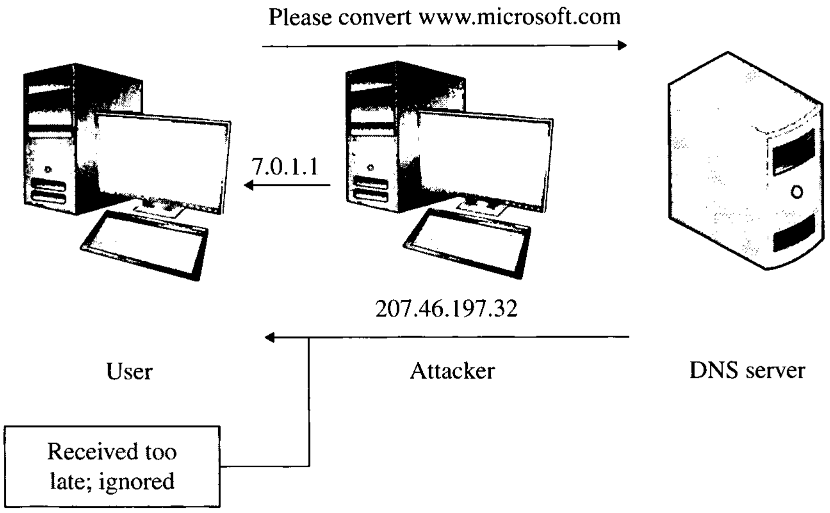

FIGURE 12-4 Address Resolution Involving DNS Spoofing

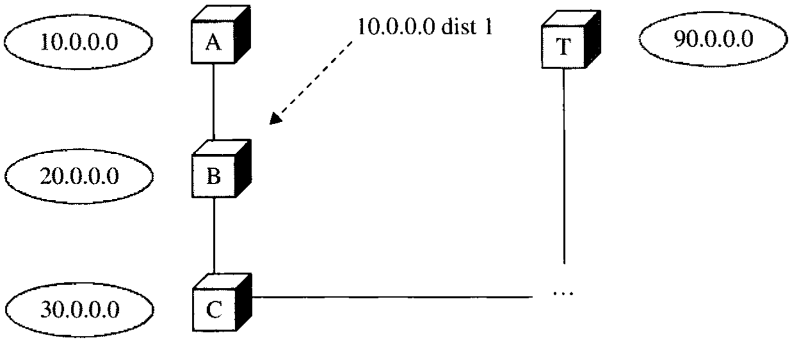

FIGURE 12-5 Router Advertises Its Subnet

Rerouting Routing

One example of a man-in-the-middle attack involves one node redirecting a network so that all traffic flows through the attacking node, leading to a potential for interception. Network routers are a loose confederation of mutually trusting components that arrange for delivery of all data through a network, including the Internet. The man-in-the-middle explanation for routers is a bit complicated, so we present a simplified version that highlights the middle role; for a more complete description of this phenomenon, consult Hepner et al. [HEP09].

Each router sends a message to other routers, listing addresses to which it has a path; the other routers then add their paths and forward the extended list to the other routers as well. In this way, all routers learn of the connections of other routers. In Figure 12-5, four routers control four subnets: A controls the 10.0.0.0 subnet; B, the 20.0.0.0, and so forth. A is adjacent to B, B is adjacent to C, and T is another router not

Digitised by the Department of Library Services, University of Pretoria, 2019

adjacent to any of the other three. A advertises to its neighbors that it is a distance of 1 from any machine in the 10.0.0.0 subnet.

Because B has just learned that router A is only distance 1 from the 10.0.0.0 subnet, B advertises to its neighbors A and C that it is distance 1 from its own subnet and distance 2 from the 10.0.0.0 subnet, as shown in Figure 12-6. Of course, A doesn't care that it could get to 10.0.0.0 addresses by going through B; that would be a senseless loop, but it does record that B is the closest path to 20.0.0.0 addresses.

Figure 12-7 shows how C takes what it has just learned from B and broadcasts it to other routers adjacent to it. In this way, the routers announce their capabilities throughout the entire network. Over time, the routers share information that details the complete network topology. Each router maintains a table of destinations and next steps, so if C had something for the 10.0.0.0 subnetwork, its table would indicate it should forward that data stream to B.

In Figure 12-8 we complicated the scene a bit by adding more routers; for simplicity we do not show their subnetworks. These routers will all advertise their connectivity,

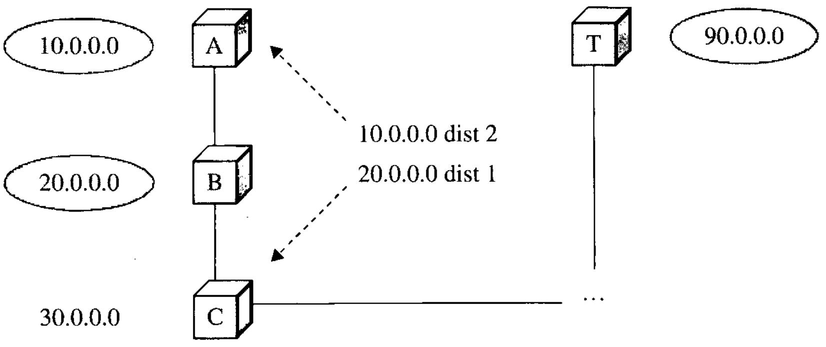

FIGURE 12-6 Router Advertises Its Own Subnet and Its Neighbor's

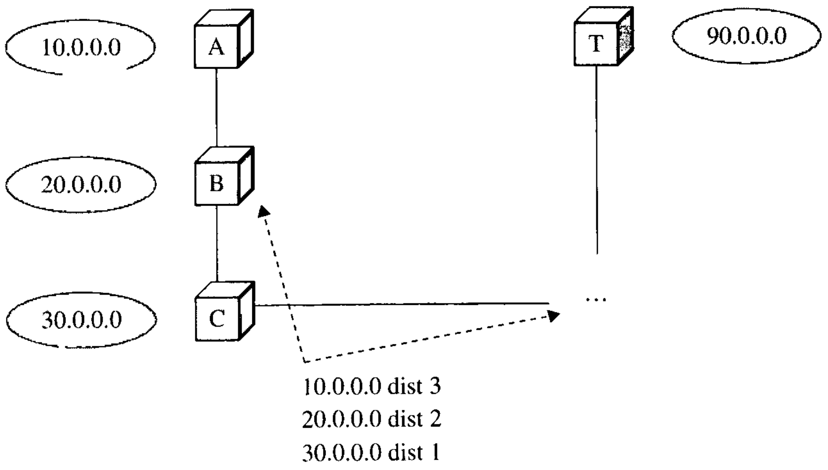

FIGURE 12-7 Router Propagates Routing Information

Digitised by the Department of Library Services, University of Pretoria, 2019

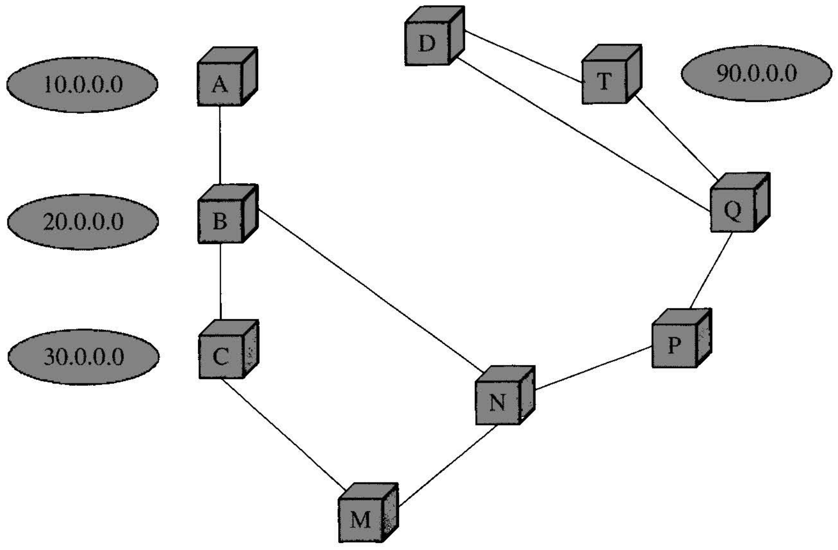

FIGURE 12-8 More Complex Router Connectivity Diagram

from which they can determine the shortest path between any pair of points. Notice that A is rather isolated from T; its shortest path is B-N-P-Q-T.

Routers operate on implicit trust; what a router reports is believed to be true. Routers do, however, sometimes malfunction or their administrators enter inaccurate data, so routing tables can become corrupted from nonmalicious (and malicious) causes. In our example, if router A advertised it was distance 1 from the 90.0.0.0 subnetwork, most traffic to that subnetwork would be routed to A, because that distance would beat any path except T itself. If A received that traffic, it could easily intercept and modify any traffic to that network, so a rogue router in a network could instigate a man-in-the-middle attack in this way.

Router Takes Over a Network

At the 2008 Defcon conference, most attendees were unaware that two researchers had rerouted the conference's wireless network through their equipment. The researchers, Pilosov and Kapela [PIL08], described and demonstrated their attack. Although the attack is more detailed than we want to present here, it extends the approach just described. Other papers (such as [HEP09, KUH07, and BEL89]) have discussed similar vulnerabilities.

Routers communicate available paths by the BGP (Border Gateway) protocol, which is complex, so attacks against it are sophisticated but certainly feasible. Details such as timing and sequence numbers must be captured and used correctly in order for a BGP update to be recognized and accepted by the rest of the network. Furthermore, attacks on the protocol depend on a device being at the "edge" of a subnetwork, that is, directly connected to two different subnetworks. Although an attacker can represent being on the edge of a local subnetwork, for example, a wireless network in a hotel or laboratory, it is harder to represent being on the edge of a larger subnetwork,

Digitised by the Department of Library Services, University of Pretoria, 2019

for example, impersonating an ISP in direct connection to the Internet. A successful attacker, however, can redirect, read, copy, modify, or delete all traffic of the network under attack.

Source Routing and Address Spoofing

Internet traffic usually travels by the best available route; that is, each router determines the best next path (called the **next hop**) to which to direct a data unit. However, a sender, using a process called **source routing**, can specify some or all of the intermediate points by which a data unit is transferred. With **strict source routing**, the complete path from source to destination is specified; with **loose source routing**, certain (some or all) required intermediate points are specified.

One use of source routing is to test or troubleshoot routers by forcing traffic to follow a specific path that an engineer can then trace. A more vicious use of source routing is to force data to flow through a malicious router or network link. Obviously, adding source routing to a data stream allows the man in the middle to force traffic to flow through his router. Because of its potential for misuse, loose source routing is blocked by many Internet routers.

Physical Man in the Middle

Network interception or wiretapping formerly involved someone with wire cutters who physically had to cut a cable and splice in a second connection. Although that kind of wiretapping is still possible, easier techniques are available today with the standard hardware interfaces of modern electronics.

Network Intrusion

Local area communications networks for small sites are often unmonitored, stand-alone collections of hardware. Because they require little attention, patch panels, cable connections, and network switches tend to be crammed into a wiring closet along with electrical circuit breakers and telephone panels. A vital piece of equipment in the closet is the router that connects the local area network to its network neighbors; one side of the router leads to the local network, and the other side to the telecommunications supplier.

An attacker can easily unplug the local network and add a second router between the network and the Internet router. Unplugging and reconnecting takes seconds and, if the new router is correctly configured, the network will quickly learn of and accommodate its new device. The attacker can connect other devices, including a wireless transmitter, to the new router, and can then perform man-in-the-middle operations on all traffic to and from the original local network. We described the essence of this attack in Chapter 11. A similar attack was performed in 2007 on the Greek cell phone network, as we describe in Sidebar 12-2.

Man in the Credit Card

Stephen Murdock and a team of researchers at the University of Cambridge [MUR10] investigated the protocols by which modern credit, debit, and ATM cards are used for

Digitised by the Department of Library Services, University of Pretoria, 2019

Greek Cell Phone Interception — Sidebar 12-2

Vodafone Greece is that country's largest cell phone provider. Sometime during August 2004 someone installed sophisticated software on the computer that routes cell phone communications; the computer physically resided on the premises of Vodafone. Mobile phone conversations were intercepted for about 100 political officials, including the prime minister, his wife, and several cabinet ministers. The software surreptitiously duplicated the communication, completing the call as normal but also directing a copy to another phone.

Vodafone uses electronic switches and software supplied by Ericsson, a Swedish manufacturer of telecommunications equipment. As reported in a detailed explanation of the incident by Vassilis Prevelakis and Diomidis Spinellis [PRE07], the switches Vodafone installed were of a general model Ericsson sold throughout the world. Some countries request an extra feature by which to implement court-ordered wiretaps, also known as lawful intercepts (as described in Chapter 11). Vodafone did not obtain that add-on, as it did not want to implement that feature, and so the delivered switches did not contain that code. In 2003, Ericsson upgraded the software in its switches and inadvertently included the intercept code in the upgrade delivered to Vodafone Greece. However, the user-level interface Vodafone employees saw did not include commands to activate or operate the intercept feature. The code was there, but Vodafone engineers did not know of its existence, nor did they have an interface to use it even if they did know of it. This hidden feature is a perfect example of a trapdoor, introduced in Chapter 3, and it demonstrates why obscurity is not an effective security control.

The Vodafone network also employs encryption so that cell phone calls are encrypted between sender and receiver. Almost. For the lawful intercept function to work, the switch must decrypt the incoming communication, forward a copy if instructed, and reencrypt the communication for delivery. (This process is an example of link encryption described in Chapter 11.)

Unknown agents installed a patch in the Ericsson switch software that activated the dormant interception code. Furthermore, the agents did so in a way that did not generate an audit log, even though interception and copying of a call usually creates an audit entry for law enforcement records. The code modification was carefully crafted to be undiscovered.

The scheme finally came to light only when Ericsson distributed a software patch; because of the rogue software, an error message in the real switch software was not delivered successfully, which triggered an alert condition. While investigating the source of the alert, Ericsson engineers found the additional software. A perhaps related fact is that Ericsson had subcontracted writing much of the original software for this model of switch to a development firm in Athens.

This attack demonstrates two points: First, an attacker who is highly motivated will use extraordinary means to accomplish an attack. Second, interception of sensitive communications is a never-ending threat. Spying existed long before computers; the presence of computers has simply expanded the number of ways for spying to occur.

financial transactions. Although the United States still uses 40-year-old magnetic stripe technology, much of the rest of the world has converted to smartcards that are capable of a certain amount of processing. To reduce the risk of card theft or loss, these smartcards use a PIN to authenticate the card's holder to the card. A customer making a purchase must enter a secret PIN to confirm legitimate ownership of the card. Verification of the PIN can be done locally by the card and terminal or remotely (over an active

Digitised by the Department of Library Services, University of Pretoria, 2019

network connection) by the bank that issued the card; a protocol negotiation between the card and terminal determines where authentication will occur. Murdock and colleagues exploit a flaw in this protocol to force local verification and to assert that the verification has succeeded, even without entry of a correct PIN.

The vulnerability is allowing the card essentially to mandate that *it* will perform verification, not the bank's server, which is not something most cards would demand. However, an in-the-middle attack could force local verification by intercepting and modifying the communication between the card and card reader. By protocol, the user has to enter a PIN at the reader, but the reader returns the PIN to the smartcard for it to verify. The middle agent intercepts the PIN, informs the card that verification is being done remotely, and informs the reader that the smartcard accepted the PIN; the rest of the transaction proceeds normally, with all parties assuming that verification had been done properly.

The attack involves a physical device in the middle, built for approximately $200 US as a prototype. The prototype device is large and bulky enough to attract attention (consisting of a laptop and a separate circuit board wired between the credit card terminal and the laptop); however, the researchers expect someone could develop such a device as a sleeve that would fit over a credit card and slip into the reader. Users are instructed not to hand cards to a merchant, but instead to insert the card into a merchant's terminal and enter the PIN discreetly. Thus, a merchant could easily miss that the customer was inserting both a card and a cover.

Murdock and colleagues note that the protocol description for smartcards is 707 pages long, with an additional 2126 pages of testing documentation. In protocols as well as software design, simplicity is a virtue.

Man-in-the-Browser Attack

A **man-in-the-browser** attack is an example of malicious code that has infected a browser. Code inserted into the browser can read, copy, and redistribute anything the user enters in a browser. The threat here is that the attacker will intercept and reuse cre dentials to access financial accounts and other sensitive things.

In January 2008, security researchers led by Liam Omurchu of Symantec detected a new Trojan horse, which they called SilentBanker. This code links to a victim's browser as an add-on or browser helper object; in some versions it lists itself as a plug-in to display video. As a helper object, it sets itself to intercept internal browser calls, including those to receive data from the keyboard, send data to a URL, generate or import a cryptographic key, read a file (including display that file on the screen), or connect to a site (pretty much everything a browser does).

SilentBanker starts with a list of over 400 URLs of popular banks throughout the world. Whenever it sees a user going to one of those sites, it redirects the user's banking activity through the Trojan horse and records customer details that it forwards to remote computers (presumably controlled by the code's creators).

Banking and other financial transactions are ordinarily protected in transit by an encrypted session, using a protocol named SSL or HTTPS (which we explain in Chapter 14), and identified by a lock icon on the browser's screen. This protocol means

Digitised by the Department of Library Services, University of Pretoria, 2019

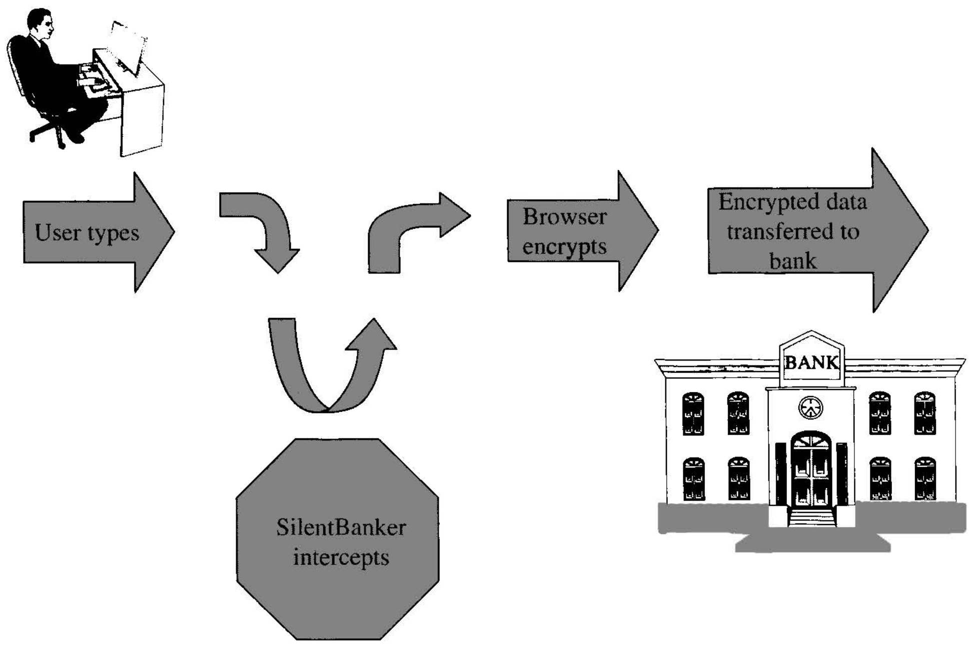

FIGURE 12-9 SilentBanker Operates in the Middle of the Browser

that the user's communications are encrypted during transit. In Chapter 11 we cautioned that cryptography, although powerful, can protect only what it can control. Because SilentBanker was embedded within the browser, it intruded into the communication process as shown in Figure 12-9. When the user typed data, the operating system passed the characters to the browser. But before the browser could encrypt its data to transmit to the bank, SilentBanker intervened, acting as part of the browser. Notice that this timing vulnerability would not have been countered by any of the other security approaches banks use, such as an image that only the customer will recognize. Furthermore, the URL in the address bar was authentic.

As if intercepting details such as name, account number, and authentication data were not enough, SilentBanker also changed screen contents. So, for example, if a customer instructed the bank to transfer money to an account at bank A, SilentBanker converted that request to make the transfer go to its own account at bank B, which the customer's bank duly accepted as if it had come from the customer. When the bank returned its confirmation, SilentBanker changed the details before displaying them on the screen. Thus, the customer found out about the switch only after the funds failed to show up at bank A as expected.

As you can see, this man-in-the-browser attack was very effective. It had little impact on users because it was discovered relatively quickly, and virus detectors were able to eradicate it promptly. Nevertheless, this piece of code demonstrates how powerful such an attack can be.

Digitised by the Department of Library Services, University of Pretoria, 2019

Continuous Authentication

We have argued the need for a continuous authentication mechanism. Although not perfect in those regards, strong encryption does go a long way toward a solution.

If two parties carry on an encrypted communication, an interloper wanting to enter into the communication must break the encryption or cause it to be reset with a new key exchange between the interceptor and one end. (This latter technique is known as a session hijack, which we study in Chapter 14.) Both of these attacks are complicated but not impossible. As we saw earlier in this chapter with browsers and Internet telephones, however, this countermeasure is foiled if the attacker can intrude in the communication preencryption or postdecryption. These problems do not detract from the general strength of encryption to maintain authentication between two parties.

These mechanisms—signatures, shared secrets, one-time passwords, and out-of-band communications—are all ways of establishing a context that includes authentic parties and excludes imposters.

COUNTERMEASURE: CRYPTOGRAPHY

Next we consider an aspect of cryptography to limit an attacker's productive use of intercepted data. You are familiar with how encrypted data are protected against unauthorized disclosure. In this section we show an interesting trick use of cryptography that squeezes out any man in the middle.

Revised Key Exchange Protocol

Remember that we began this discussion with a man-in-the-middle attack against a simple key exchange protocol. The faulty protocol was the following:

1. A says: B, please send me your public key.
2. B replies: Here, A; this is my public key.
3. A responds: Thanks. I have generated a symmetric key for us to use for this interchange. I am sending you the symmetric key encrypted under your public key.

At step 2 the intruder intercepts B's public key and passes along the intruder's. The intruder can be foiled if A and B exchange half a key at a time. Half a key is useless to the intruder because it is not enough to encrypt or decrypt anything. Knowing half the key does not materially improve the intruder's ability to break encryptions in the future.

Rivest and Shamir [RIV84] have devised a solid protocol as follows.

1. Amy sends her public key to Bill.
2. Bill sends his public key to Amy.
3. Amy creates a symmetric key, encrypts it using Bill's public key, and sends half of the result to Bill. (Note: Half of the result might be the first *n*/2 bits, all the odd numbered bits, or some other agreed-upon form.)
4. Bill responds to Amy that he received the partial result (which he cannot interpret at this point, so he is confirming only that he received some bits).

Digitised by the Department of Library Services, University of Pretoria, 2019

Bill encrypts any random number with his private key and sends half the bits to Amy.

5. Amy sends the other half of the encrypted result to Bill.
6. Bill puts together the two halves of Amy's result, decrypts it using his private key and thereby obtains the shared symmetric key. Bill sends the other half of his encrypted random number to Amy.
7. Amy puts together the two halves of Bill's random number, decrypts it using her private key, extracts Bill's random number, encrypts it using the now-shared symmetric key, and sends that to Bill.
8. Bill decrypts Amy's transmission with the symmetric key and compares it to the random number he selected in step 6. A match confirms the validity of the exchange.

To see why this protocol works, look at step 3. It is true that Malvolio, the intruder, can intercept both public keys in steps 1 and 2 and substitute his own. However, at step 3 Malvolio cannot take half the result, decrypt it using his private key, and reencrypt it under Bill's key. Bits cannot be decrypted one by one and reassembled.

At step 4 Bill picks any random number, which Amy later returns to Bill to show she has successfully received the encrypted value Bill sent. Such a random value is called a **nonce**, a value meaningless in and of itself, to show activity (liveness) and originality (not a replay). In some protocols the receiver decrypts the nonce, adds 1 to it, reencrypts the result, and returns it. Other times the nonce includes a date, time, or sequence number to show that the value is current. This concept is used in computer-to-computer exchanges that lack some of the characteristics of human interaction.

After two parties have securely established a shared cryptographic key, they can use encryption to block the impact of a man in the middle, specifically the man-in-the-browser or page-in-the-middle kinds of attacks. But as this protocol shows, even before they have exchanged keys, they can use encryption to protect data that one party will be able to decrypt only in the future.

There is still a problem with this protocol, but it cannot be solved in the protocol itself. If Amy begins the protocol interacting with Malvolio instead of Bill (thinking Malvolio is Bill, that is, if Malvolio impersonates Bill from the beginning), Amy cannot detect the falsehood within this protocol. That is a problem of identification and authentication, perhaps requiring some out-of-band or shared secret, as we just presented.

Summary

Once again, cryptography has shown its strength as a countermeasure. You can see why cryptography is so important against man-in-the-middle attacks: If the channel between the two end parties is encrypted, there is little opportunity for the man to intrude in the middle of the exchange. Identification and authentication—both for human-to-computer and computer-to-computer —is a strong countermeasure, as are techniques to enforce access control. However, as noted in the man-in-the-browser example of SilentBanker, cryptography must be employed at the right moment. The strongest cryptography cannot secure a piece of data if malicious code gets to it before encryption.

We next turn to a slightly different issue, covert channels. Instead of a man in the middle, this problem has a man on the outside, to which at least one end party does

Digitised by the Department of Library Services, University of Pretoria, 2019

the flow of information from a covert channel. The hardware-based channels cannot be closed, given the underlying hardware architecture.

Although covert channel demonstrations are highly speculative—reports of actual covert channel attacks just do not exist—the analysis is sound. The mere possibility of their existence calls for more rigorous attention to other aspects of security, such as program development analysis, system architecture analysis, and review of output.

RELATED ATTACK: STEGANOGRAPHY

We conclude this chapter with a different kind of in-the-middle attack. This attack is related to covert channels because it is a means to pass information surreptitiously. It is also a form of in-the-middle attack because it can be used by a third party to piggyback on a communication between an unsuspecting sender and receiver.

We are about to describe **steganography**, the science of hidden writing. Cryptography is different, because the goal of encryption is to conceal meaning, but the output remains in plain sight. The goal of steganography is to conceal that anything is written, which naturally also conceals the content. Steganography applies not just to writing using letters and words, but also to communication using any recorded data; music, graphics, and video files are especially good for steganographic embedding.

Information Hiding

Steganography is part of the larger topic called **information hiding**, an activity that has been practiced for centuries. Fabien Petitcolas and colleagues at Cambridge [PET99] carefully define the various aspects of information hiding. Gus Simmons [SIM84] framed information hiding in the context of prisoners wanting to communicate in a way their guards would not notice. If you were a spy, being captured with a sheet of encrypted text would be incriminating, even if your captors could not decrypt the content. However, a handwritten diary might well pass as innocuous. Your diary contains your own thoughts, and you can write using incomplete sentences, misspellings, misused words, and incoherently meandering topics. You can also implement a nonobvious pattern to represent a sensitive thought you want to write so everything can reflect elusive topics. Did you notice that the first letters of the last phrase of the previous sentence spell the word s-e-c-r-e-t? That is an example of information hiding.

Spies are not the only information hiders. Steganography can be used to mark objects to show derivation or ownership. If you copy prose, anyone can compare two samples and deduce plagiarism. It is much more difficult to demonstrate digitally that a music file is a recording of the Berlin Philharmonic. If the Berlin Philharmonic embeds a recognizable string, sometimes called a **digital watermark**, in its recording, the string can help them identify and prove that a music file is actually theirs. This topic has received attention recently because of **digital rights management**, the goal of the owner of a copyright to a work of art (painting, film, sound recording) to control use and copying of the work.

We do not intend to cover the entire topic of information hiding, which is beyond the scope of this book. However, we furnish one technical example to give you a sense of the approach.

Digitised by the Department of Library Services, University of Pretoria, 2019

Technical Example

Consider a typical computer image file, such as a bitmap or photo file. These file formats are sets of bits representing the colors of individual pixels in an image. Colors can be represented in 8-bit or 24-bit encodings; 8-bit encodings can represent a 256-color palette, and 24 bits yield 16,777,216 colors. Both encodings go from white to black, passing through pink, gold, indigo, and green along the way. A 24-bit encoding of brown, for example, is 0xc08000, meaning 192 (0xc0) parts red, 128 (0x80) parts green, and 0 parts blue. (Each of these three values ranges from 0 to 255 or 0x00 to 0xff.)

Composed of filtered light, brown is all red and green, no blue, although browns come in many shades, such as tree bark, bear's fur, soil, coffee, caramel, or sand. The human eye can hardly detect minor changes of a single bit: (using decimal notation) 191, 128, 0 looks brown, as does 193, 128, 0, or 192, 127, 0, or 192, 128, 1. Suppose you have a picture of the bark of a tree, all brown, but with variations in shading and color showing light and shadow, depth, and varied materials. Several adjacent pixels will have the same color pattern, and then there is a shift as the feature becomes darker. In the first column of Table 12-4 we show such a transition. We represent the colors in hexadecimal notation, separated to show the red–green–blue components.

With steganography we can embed a message by co-opting the low order bit of each pixel. Suppose we want to embed the binary string 000 101 011 000 101 111 011 001 110 001 000. As shown in the last column for Table 12-4, we arbitrarily change the least significant bit of each color of each pixel to the bit of the message we want to pass, effectively ORing the strings. Notice that sometimes the original rightmost bits do not change, but this does not interfere with the communication.

TABLE 12-4 Embedding a Binary String through Steganography

Original Color	Message to Embed	Result
c0 80 00	0 0 0	c0 80 00
c0 80 00	1 0 1	c1 80 01
c0 80 00	0 1 1	c0 81 01
c0 80 00	0 0 0	c0 80 00
c0 80 00	1 0 1	c1 80 01
c0 80 00	1 1 1	c1 81 01
c0 80 01	0 1 1	c0 81 01
c0 80 02	0 0 1	c0 80 03
c0 80 04	1 1 0	c1 81 04
c0 80 03	0 0 1	c0 80 03
c0 80 07	0 0 0	c0 80 07

Digitised by the Department of Library Services, University of Pretoria, 2019

We chose this embedding scheme for easy explanation. As you can see, the changes between adjacent pixels are slightly abrupt. The changes would still be invisible to the human eye, but the frequency and discontinuity of the changes might be detected by a program that measures distortion to search specifically for unusual variation. Other embedding techniques are more subtle.

This method is also a good case for a form of in-the-middle attack. How many photos of famous landmarks are there on the Internet (and how many will be added each year)? An attacker can easily collect a virtual album of photos, doctor them steganographically, and repost them in a different album. Most people finding the album would see only travel photos; colleagues of the attacker, knowing the album and hiding algorithm, can download the photos and extract the information.

Charles Kurak and John McHugh [KUR92] present an interesting analysis of covert signaling through graphic images. In their work they show two different images combined by some rather simple arithmetic on the bit patterns of digitized pictures. The second image in a printed copy is undetectable to the human eye, but it can easily be separated and reconstructed by the spy receiving the digital version of the image. Simon Byers [BYE04] explores the topic in the context of data hidden within complex Word documents.

CONCLUSION

This chapter has given a name to our attacker behind the curtain, Mr. Man in the Middle. (We apologize that the terminology makes it seem as if only men perpetrate these attacks.) We have introduced him in this chapter to describe some complex attacks that involve both interception and fabrication. You will probably notice that our attacks have gotten more complex and sophisticated, and the interception step leads to deducing information, which in turn leads to being able to create new information.

As our examples have shown, man-in-the-middle attacks can occur in many different contexts, from computer-to-computer communications such as routing data, to person-to-computer or computer-to-person interchanges, such as the Syrian air defense example. Of the vulnerabilities we discussed, trust, identification and authentication, and access control have come up several times already, which just underscores how serious they are.

In this chapter we introduced one important new countermeasure: use of public key cryptography for key exchange. We have alluded to the technique in earlier chapters, but the full solution had to wait until this chapter to explore how to block attacks from the man in the middle. Here are the key points we presented in this chapter:

- In-the-middle attacks occur in many contexts: protocols, routing, addressing, web browsing, and applications.
- There is no single vulnerability that permits in-the-middle attacks; they seem to come from design and implementation flaws, poor or incomplete identification and authentication, and misplaced trust.
- Cryptography, applied in the right places, can counter such attacks, because it prevents the intermediary from seeing or modifying critical data. But cryptography

Digitised by the Department of Library Services, University of Pretoria, 2019

TABLE 12-5 Threat–Vulnerability–Countermeasure Chart for Man-in-the-Middle Attacks

Threat	Consequence	Severity	Ease of Exploitation
Man-in-the-middle	Modification	High	Fairly easy

Vulnerability	Exploitability	Prevalence
Identification and authentication	High	High
Unauthorized access	High	High
Program flaw	High	High

Countermeasure	Issues Addressed	Mitigation Type	Mitigation Effect	Effort
Protocols	Authentication	Prevent, detect	High	Difficult
Access control	Unauthorized access	Prevent	High	Moderate
Cryptography	Authentication	Prevent	High	High

has to be applied at the right time; otherwise, it only seals the attack that has already occurred.

- Covert channels are often a type of man-in-the-middle attack. A covert channel is a method by which an inside malicious process can signal sensitive data to an outside receiver using an existing baseline communication band.
- Steganography is another form of concealed communication. Instead of trying to hide the communication, steganography presents it in clear sight, but in a form that is not likely to be noticed.

In the next chapter we consider the related topic of forgeries: how to determine that something, particularly computer code or data, is real. Two issues involved are trust, how to determine whether the source of the object is authentic, and delivery, how to determine that no malicious man in the middle has modified the object during transmission.

EXERCISES

1. Why is caching DNS query results a good strategy, in spite of the possibility that the cache may have been corrupted by a malicious outsider?
2. Does it really matter if a DNS query resolves incorrectly? Won't a user wanting to go to a site such as xyz.com readily notice being at uvw.com, instead? Explain your answer.
3. How can a router validate the veracity of information it receives from other routers?
4. The SilentBanker man-in-the-browser attack depends on malicious code that is integrated into the browser. These browser helpers are essentially unlimited in what they can do. Suggest a design by which such helpers are more rigorously controlled. Does your approach limit the usefulness of such helpers?

Digitised by the Department of Library Services, University of Pretoria, 2019