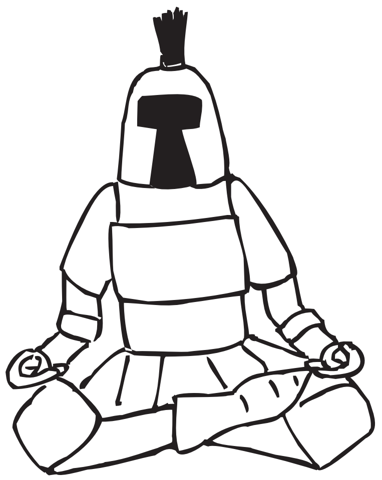
# 5 Developing a Higher Security Mind



## THE ART OF HIGHER SECURITY

Earlier I discussed the Virtue of Higher Focus as a fundamental security concept. With hundreds of thousands of hackers using hundreds of thousands of tools to exploit hundreds of thousands of vulnerabilities, there is little hope of addressing every possible security issue directly. Thus, it is important to approach security from a "higher" view.

If a new worm broke out, putting the Internet on high alert for contamination, it would, of course, be necessary to take a specific action, apply a specific patch, close a specific port on the firewall, or add a specific signature to all IDS devices. This is not contradicting the Virtue of Higher Focus. However, if we took specific actions that prevented only this worm and not the 100 similar worms soon to be developed, we would be in violation of this virtue.

The Virtue of Higher Focus represents the way in which we must think about security in our everyday lives. Addressing security in a higher manner helps us deal with two common security problems:

- It is impossible to secure ourselves by applying unique security measures for every vulnerability in existence.
- By thinking in terms of specific vulnerabilities and exploits, we are only able to react to security issues rather than deal with them proactively.

The question is, then, how do we deal with higher security? How do we work to keep ourselves safe when hundreds of new exploits are developed every month? The answer to these questions comes with some time-honored security practices, best practices that have been used for thousands of years. In this chapter, I will review several of the key security tools that will keep an organization safe, despite the highly dynamic nature of information warfare. All of the following practices help to generalize security practices and further develop security minds. These practices include:

- Thinking in zones
- Creating chokepoints
- Layering security
- Understanding relational security
- Understanding secretless security
- Dividing responsibilities
- Failing securely

## THINKING IN ZONES

Zoning is a process that is essential for making any security decision. Briefly, zoning is the process by which we define and isolate different subjects and objects based on their unique security requirements. Again, I use the standard terms "subjects" and "objects" because we could really be talking about anything. Zoning is most commonly thought of as a network-based solution, but truly, the concept of zoning is fundamental to all security decisions. A store pharmacy could, for instance, be classified into three zones, or three separate places where security is treated in a different manner. There is the front counter, where the customer requests the drugs and provides payment; there is the technician, who relays the request to the lab in back and returns with the drugs; and then there is the actual pharmacist, who fills prescriptions. Each of these areas has its own unique risks, vulnerabilities, and security needs that define its zones. Imagine if we simply let the customers directly into the back room to fill their own prescriptions!

In this section, we will discuss several different zoning scenarios that are possible and the advantages each has to offer. We will then work to apply the zoning process by defining subjects and objects and determining which zoning scenario fits best. Almost every security decision, technical or non-technical, involves zones, so while going through each zoning scenario, be sure to keep an open mind for how these concepts can be applied. Remember that zoning is not a network-specific concept, and that zones should be created for applications, physical areas, and even for employee interactions as a defense against social engineering.

### Defining a Zone

The term "security zone" is thrown around a lot in the security world; it is used for everything from application design to security camera placement. So, how do we define a zone?

A zone is a logical grouping of resources that have a similar security profile. That is to say, it is a grouping of objects that have similar risks, trust levels, exposures, policies, or security needs. A client's computers connecting from across the Internet, for example, have a different level of security and trust than an internal DB server. Similarly, a local mail server that accepts mail directly from the Internet has a different level of exposure than an internal mail server. Thus, the two would be considered to be in two different zones.

Though there can be numerous zones within any situation, the most common scenarios involve the three zones shown in Table 5.1 the trusted (or internal) zone, the untrusted (or external) zone, and the semi-trusted zone (or DMZ). These three zones can apply to almost anything, including networking and application programming, as well as designing physical security layouts.

**Table 5.1** *Zone Definitions*

| **Zone** | **Description** | **Examples** |
|---|---|---|
| Trusted | The trusted zone is where the organization's most valuable and sensitive resources exist. This zone is under our control and governed by our policies. | *Network:* Internal servers and workstations<br>*Application:* Trusted pieces of application code<br>*Physical*: Server rooms |
| Semi-Trusted | The semi-trusted zone (or DMZ) is for resources that have some degree of direct exposure. This zone is still under our control and governed by our policies, but is somehow exposed or more vulnerable than a trusted zone. | *Network:* Externally accessible Web, mail, and DNS servers<br>*Application*: Front-end code or untrusted third-party code<br>*Physical*: Lobbies and waiting rooms |
| Untrusted | The untrusted zone is the area where we have no direct control and which is not governed by our policies. | *Network*: The Internet and dial-up phone lines<br>*Application*: End-users or external devices<br>*Physical*: Everything outside the building |

## Separating Zones

Having a security vulnerability or exposure is similar to having the common cold. If one object has it, all other objects near it are likely to be exposed. To protect valuable resources, we must be able to maintain high levels of security by protecting resources from zones of lesser security control. "Zoning" is the process by which we group similar objects into proper zones and separate them from other zones for added protection. The separation mechanism could be as simple as a firewall, a security control applet, or a locked door. The goal is to have some degree of control over what happens between the different zones.

## Communication Between Zones

While separating zones is all well and good for security, it would not be practical to completely isolate all zones from each other and never allow them to communicate. Just because the Internet is untrusted does not mean we should simply cut off all internal access to it. However, allowing communication between zones can be extremely dangerous if the proper security measures are not taken. Fortunately, there are several conventions to safely allow access to take place between different security zones. Each convention has its own advantages and level of exposure to consider, but almost every situation can find a security solution in one of these zoning conventions.

In the following section, we will be looking at these zoning conventions. We will look at the different zoning possibilities and the levels of exposure associated with each, as shown in Figure 5.1. We will start with the least secure and least desirable scenario and progress to the most secure and desirable scenario. Each added level of security has its potential drawbacks in flexibility and functionality, so it is important to adopt the practice that provides the least exposure without making any harmful sacrifice in usefulness.

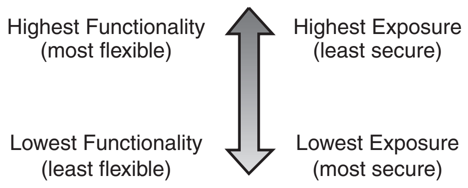

Highest Functionality (most flexible)
Highest Exposure (least secure)
Lowest Functionality (least flexible)
Lowest Exposure (most secure)

**Figure 5.1** *As zoning functionality increases, exposure increases.*

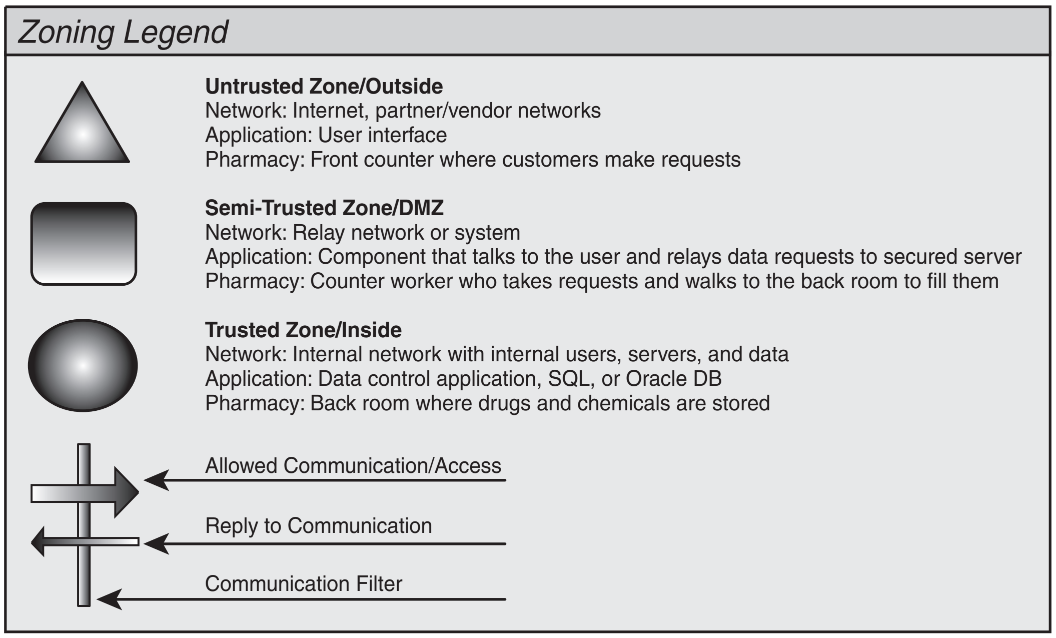

Zoning Legend
Untrusted Zone/Outside
Network: Internet, partner/vendor networks
Application: User interface
Pharmacy: Front counter where customers make requests
Semi-Trusted Zone/DMZ
Network: Relay network or system
Application: Component that talks to the user and relays data requests to secured server
Pharmacy: Counter worker who takes requests and walks to the back room to fill them
Trusted Zone/Inside
Network: Internal network with internal users, servers, and data
Application: Data control application, SQL, or Oracle DB
Pharmacy: Back room where drugs and chemicals are stored
Allowed Communication/Access
Reply to Communication
Communication Filter

**Figure 5.2** *This section includes several diagrams; this legend will help us talk about them.*

## Inbound Communications/Access

When we think about access security and zones, we normally think about inbound communications. These are the communications where something in an untrusted zone needs to talk to something in a trusted zone. A common example of this would be customers on the Internet accessing a page on a corporate Web server.

The inbound access zoning scenarios we will discuss include:

- *High exposure*—Direct inbound access/communication
- *Medium exposure*—Relayed inbound access/communication
- *Medium–low exposure*—Indirect inbound access/communication
- *Low exposure*—Inbound traffic to an isolated DMZ

### High Exposure: Direct Inbound Access/Communication

The situation depicted in Figure 5.3 is the most exposed and the most unsecure of the zoning scenarios. Here, an untrusted subject is allowed to make direct contact with a trusted object. In a casino, this would be similar to allowing the customers direct access to the vault, with perhaps one or two guards restricting this access. The trusted object is exposed to attacks from the untrusted zone. If an attack is successful, critical information and services on the trusted object will be compromised.

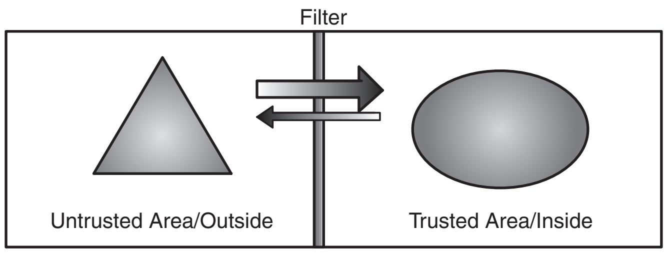


**Figure 5.3** *High exposure: direct inbound access/communication.*

### Medium Exposure: Relayed Inbound Access/Communication

In the scenario in Figure 5.4, we introduce a middle relay into the mix. A relay is responsible for negotiating communications with an external untrusted party, making

*Here it is important to point out that, despite the fact that there is a filtering mechanism in place, communication is still taking place "directly." While a firewall or filtering code may exist between the two zones, it is only acting as a security bridge between the communicating parties. The actual channel of communication is directly connecting the two parties, exposing the trusted party to attack. A successful attack could compromise any system hosting sensitive data.*

sure they are acceptable, and then passing them on to the receiver in the trusted area. The filters in this situation limit the access from the untrusted area into the relay, which allows the relay to remain dynamic, and communicate with many parties with some level of protection. This would be similar to having casino customers request their money from the vault and using trusted employees to actually collect the cash.

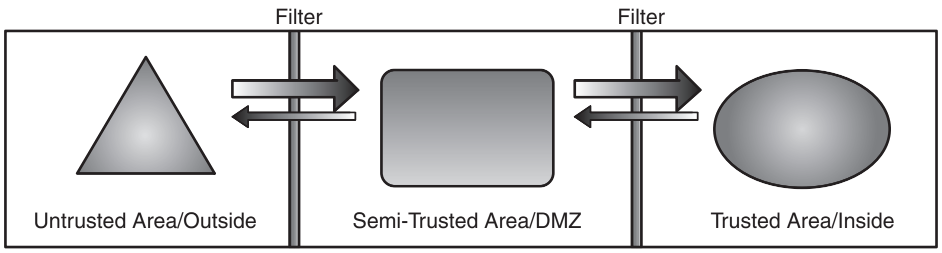


**Figure 5.4** *Medium exposure: relayed inbound access/communication.*

Here, the relay is exposed to attack from the untrusted area. The relay is, however, dedicated to its task and is not running extra code, applications, services, or other things that would expose it to many attacks. The communication between the relay and the internal object is also extremely restricted. Unlike the channel to the untrusted zone, the channel between the relay and the trusted zone is highly restrictive and communications must conform to a strict set of expectations. Thus, if the relay is compromised, the chances of being able to use the relay to attack the system in the internal zone are greatly reduced.

This scenario is one of the most commonly adopted for securing an email or DNS server, authentication and data access application models, and other services or functions that require real-time access to systems, applications, or services in a trusted area.

## Medium–Low Exposure: Indirect Inbound Access/Communication

We make another major improvement in our relay scenario when we can treat the relay as a mechanism that listens to everyone and talks to no one (see Figure 5.5). More accurately, the relay receives requests and updates from parties on the trusted and untrusted sides, but initiates no requests of its own. Thus, the relay has no direct access to any resources in the trusted zone. This would be similar to having security personnel supply the casino cashiers with funding from the vault, rather than letting the cashiers get funds directly.

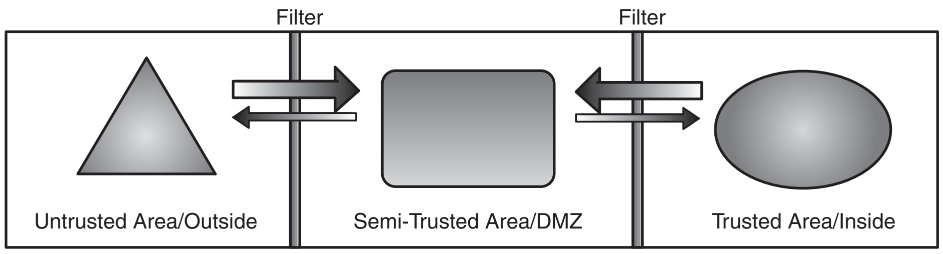


**Figure 5.5** *Medium–low exposure: indirect inbound access/communication.*

A common example for this scenario is a Web server or application that replicates data from an internal source on a regular basis. The Web server may be designed to request information from customers over the Internet. When a customer enters the information, it is not directly relayed to the internal system, but rather, it is stored in a temporary location until the internal system performs a scheduled polling. This allows for customers on the Internet to update their information without exposing the internal data system to the untrusted Internet, or to the semi-trusted Web server.

## Low Exposure: Inbound Traffic to an Isolated DMZ

Figure 5.6 shows the most secure approach that can be taken when hosting a service accessible from an untrusted zone. In this scenario, the semi-trusted application or server functions independently of the trusted zone. It services requests from the untrusted area without risk of compromising trusted zones. This would be similar to having the casino cashier only give out chips, with no access to the vault at all.

As might be expected, our most secure solution is also the least functional. While it provides the most secure approach to hosting services and applications, it can only be

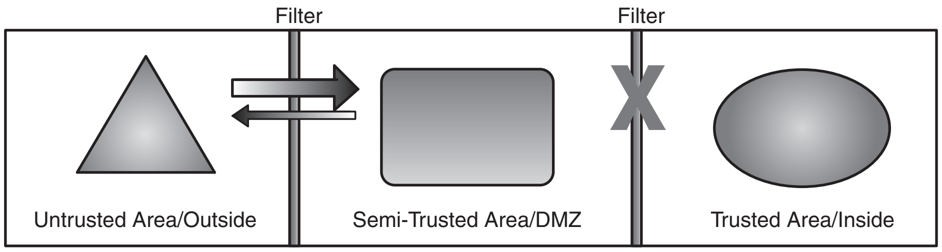


**Figure 5.6** *Low exposure: inbound traffic to an isolated DMZ.*

used in limited situations where data updates and requests are not required to or from a trusted area. The classic use for this scenario is in an external Web server that requires no data to be polled or pushed from an inside network. Updates for Web pages are copied to a CD-ROM and installed manually on the server. If there is any statistical information on the Web server, it is copied manually via diskette and not through the network. This scenario is highly recommended for systems and applications that do not require heavy maintenance or dynamic data.

## Outbound Communications/Access

Outbound communications and access are often thought of as being safe and secure. In networking, for example, many organizations put up firewalls with a simple ruleset that states: "Everything is allowed outbound; nothing is allowed inbound." In fact, there are even professional firewall devices from prominent network companies that use this as the default configuration, encouraging their customers to worry only about inbound communications.

It is true that outbound communications are less often exploited than inbound communications. This is only true, however, because hacking into a system through its outbound access does not present quite as many easy opportunities to strike. Hacking a system making outbound calls is still however quite common in the security world and is just as deadly as hacking an object accepting inbound calls.

In most forms of access, there must be data transferred in both directions. To browse a Web server, for instance, that server must send pictures, text, files, and even scripts. Sometimes this communication is obvious; sometimes it is transparent and hidden by other tools. Most modern firewalls, for example, automatically open a port for an untrusted party to send data back to a requesting client. This data can be loaded with malicious scripts, files, and miniature applications that can be used to hack a system or network.

The NIMDA worm, for example, infects Web servers and then infects the PCs of those clients browsing the server's pages. Those PCs would then spread the worm to internal systems and servers via numerous other methods. A normal network firewall would be unable to detect or prevent such activities, even if the firewall strictly forbade access from untrusted to trusted systems. Since the client is the one requesting the access and opening the channel, NIMDA simply rides back on the existing communication.

The outbound access zoning scenarios we will discuss include:

- *Medium exposure*—Direct outbound access/communications
- *Low exposure*—Relayed outbound access/communications

## Medium Exposure: Direct Outbound Access/Communications

In the scenario shown in Figure 5.7, an internal system is communicating with an external system in an untrusted area. For instance, in a networking example, internal users would communicate (through an open firewall) to systems on the Internet for Web browsing or chatting.

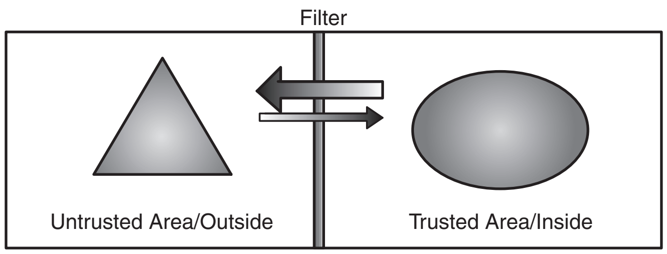


**Figure 5.7** Medium exposure: direct outbound access/communications.

There are two primary concerns when considering this form of direct outbound access, be it via networks, applications, or physical means:

- *Possible exposure in the return path*—In almost all cases where outbound access is required, a small gap must be made in security to allow for the untrusted party to reply. When we open our front door to talk to a person outside, there is always the chance that the individual will attempt to force his or her way into our protected area. If we make a network communication with a partner, there is always the chance that the second half of the communication will contain an exploit or virus.

- *Increased chance of exposure from internal parties*—When direct outbound access is allowed with external parties, there is a much greater chance for an exposure to occur from the inside. If, for example, a bit of malicious code is launched within an internal application, it can open a communication channel to untrusted areas, allowing for remote control. For example, a back door on a computer can initiate a tunneled communication to an external party to allow it to hack into the trusted network.

## Low Exposure: Relayed Outbound Access/ Communication

To allow access from a trusted area into an untrusted area without being openly exposed requires a relay between the subject and object. Similar to the inbound relay scenario, the relaying system or application component must be a dedicated device or code that is responsible for communicating with the external server on our behalf. A classic example of this process would be a network proxy server (application firewall) that relays all Web requests to the Internet for us.

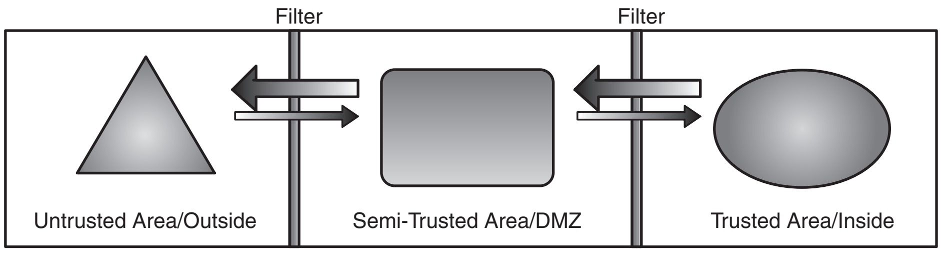


**Figure 5.8** Low exposure: relayed outbound access/communications.

The system or application performing the relay services accepts numerous requests and replies from both trusted and untrusted networks. The relay limits its own communications to the internal systems in such a way that it will only pass on normal data and will strip any unexpected, extraneous, or unauthorized data. This process helps to protect all systems, and focuses more security attention on the relay instead of the internal object.

## Applying the Zoning Concepts

To apply these zoning concepts in our decision-making process, we must learn to quickly recognize where different levels of security are needed. The following six-step procedure can help in this process:

**Six-Step Zoning Process**

1. Identify any instance where an untrusted or less trusted object comes into contact with a trusted, valuable, or more sensitive object.
2. Determine the direction of communication that is required. Ask yourself, "Is it possible to use an outbound model, or does the communication need to be initiated by an untrusted object?"
3. Determine where it would be possible to separate the trusted object into two components: one that handles the sensitive data and the other that acts as a relay or middle entity in the transaction.
4. Decide what forms of communication need to take place between the outside, middle, and inside, and which zoning model to apply. Work through them in this order:
   a. Can you effectively and efficiently use the low-exposure models?
   b. If not, can you use the middle–low- or low-exposure models?
   c. If neither model can be used, does the value of the communication and its potential risk make the high-exposure model an acceptable solution?
5. In the model that is chosen, place as many security controls between each of the components as is reasonably possible.
6. Document the reasoning, supporting data, and conclusion in this decision-making process. Keep this document for reference and to simplify the decision-making process for similar situations in the future.

## Example of the Zoning Process

A baked goods engineering division has an existing server within its internal infrastructure that keeps the recipes for various types of bread. This information is kept in a secure DB on an internal network. A new initiative has been made to allow for partner bread-baking companies to access specific entries in the ever-changing DB. The board of directors has requested access to designated recipes for partner companies with a valid login and password through a secure Web browser.

**Step 1** In this situation, the trusted/sensitive and untrusted objects are obvious:

| | |
|---|---|
| Trusted/Sensitive object: | Recipe data |
| Untrusted object: | Bread-baking partners |

**Step 2** Since the bread-baking partners are going to be accessing the data as they need it and will be using a standard Web browser, we must allow them to initiate requests for information. The communications will be inbound (their Web browsers will be coming into the network to access the recipes).

**Step 3** The recipes are stored in an Oracle DB; a standard bread-bakers application is being used to access the information. For external partners, we will need to create a Web server to act as the front-end. The Web server will be in a semi-secure part of the network and will get its data from the Oracle server, which will remain safely in the internal network.

**Step 4** In considering the different exposure models, the low-exposure model would make it nearly impossible to handle real-time requests. By the nature of the project, partners will be accessing the data in real-time, and thus we must be able to relay the request at the same time that it is made. As such, the medium-exposure model, where the Web server will authenticate a user and then pass the request for information back to the Oracle DB, is a better choice. The Web server is the middle entity, so if it comes under attack, it does not directly contain any sensitive data.

**Step 5** The Web server is on an isolated network protected by a firewall and IDS devices. The firewall allows only Web (Hypertext Transfer Protocol [HTTP] and Secure Hypertext Transfer Protocol [HTTPS]) traffic to flow from the partners into this Web server. The Web server must then pass through the firewall again to access the internal Oracle server. This communication is limited to a very strict and defined set of DB calls that allow access only to required types of information. Thus, if someone manages to attack the Web server, there should be no effect on the critical DB server.

**Step 6** Document the decisions made with each step and place them in our "Security Decisions" folder for future reference.

## CREATING CHOKEPOINTS

*Chokepoints* have been the key to security practices since the dawn of warfare. A chokepoint is a tight area wherein all inbound and outbound access is forced to traverse. Kings of old have understood that funneling enemies through a tight doorway makes it much easier to rain down fiery oils on them. Likewise, it is much easier to keep a thief out of a network when the network has only one gate leading in and out. In information security, chokepoints offer many advantages, including:

- *Security focus*—A chokepoint focuses our attention and resources on one area of control. This greatly enhances security while reducing the ultimate taxation on our resources.
- *Ease of monitoring*—Chokepoints greatly enhance our ability to monitor access and watch for intrusions. It is much easier to see enemies entering the castle when there is only one place to look.
- *Ease of control*—Chokepoints allow for a stronger breed of security control. It is much easier to implement good security mechanisms when only dealing with a limited space.
- *Cost reduction*—By filtering all access though one point, we will only need to implement one control device as opposed to implementing a separate control for every object. This reduces the time and materials required for the implementation and maintenance of security measures.
- *Exposure reduction*—By focusing on one or two areas of access, we introduce fewer opportunities for error and exposure than if we enforce security controls in multiple areas.

Chokepoints are a key element in maintaining a higher security practice. Creating chokepoints greatly reduces the infinite number of possible attacks that can take place, and thus are some of the best tools to use in information security.

## Network Chokepoints

Network security uses chokepoints all the time. Rather than having all desktops dial into the Internet, it is common to consolidate traffic through a single controlled access point. Such chokepoints enable a high level of control on transactions between internal trusted networks and the outside world. Without such a chokepoint, higher levels of security would be needed at all entry points, making security much more difficult and expensive.

Every entry point from an external network into an internal network should be consolidated through one or more protected areas. Policies and practices in this respect should be focused on two things:

1. Securing chokepoints via filtering and monitoring
2. Ensuring that all traffic flows though chokepoints and that no new entry points are introduced

Common forms of traffic to force through a chokepoint include:

- Internet connections, including inbound and outbound access
- Vendor, partner, and customer WAN connections
- Virtual private network (VPN) and dial-in access points
- Wireless networking access points

## Application Chokepoints

Chokepoints are important in both applications and services. User access into an application should be controlled by a module that filters and monitors activities. Rather than allowing a user to jump from service to service and having to enforce security on each, we should place the majority of the security focus on a single point.

If we take, for example, the Microsoft Windows 2000 operating system (or indeed, many other operating systems), we see that each workstation and server belongs to a larger domain that controls authorization, monitoring, and other aspects of security. Each user or group of users belongs to a specific domain. This scenario allows us to make the domain controller an application chokepoint. We do not need to rely on every workstation to authenticate users and log activities; we can simply forward authentication and logs on to the chokepoint controller. This allows us to focus security efforts in a central area rather than in each and every workstation.

Many other applications allow for access chokepoints, including some single sign-on and portal applications. Large organizations oftentimes develop a front-end application that secures access for many back-end applications. Applications that create an access chokepoint are very helpful in securing large organizations.

## Social Chokepoints

Just as our networks and applications can be directly exposed to attacks from external entities, so can our employees, executives, partners, and customers. It is important to understand that, in the average organization, employees are given a great deal of information that is useful to a hacker. If every employee is in direct contact with everyone else in the world, then there is a great potential for a social engineering attack to perform. No doubt, one employee will prove to be the weak link within the social chain and disclose sensitive information to an attacker.

Creating a social chokepoint can be accomplished in many ways. In extreme cases, employees are not allowed to directly contact the outside world during business hours. This approach, however, is not applicable to many organizations. A good solution for many organizations has been to create virtual chokepoints for specified types of information. For example, employees can be trained that passwords and other sensitive information can only be discussed with a very specific group or department in a very specific context. Some form of predefined verification process must occur before such information is disclosed. Meanwhile, employees will be informed about confidential information, and that such information will never be solicited via email, outside phone calls, Web browsing prompts, or other unsafe contexts. Such techniques can be used to guard other information, such as employee names, phone numbers, physical security measures, access points, as well as passwords and other technical security measures.

## Consolidating Chokepoints

At first glance, we may conclude that all access should be consolidated though a single chokepoint. Indeed, for some organizations, this is the best choice, but certainly not for all. When allowing access though a chokepoint, we are essentially opening a hole and potentially a vulnerability in our defenses. If, for example, we make both the Internet and trusted partners filter though the same chokepoints, we may experience some undesirable results. An untrusted entity, for example, could try to gain access by masquerading as a trusted partner, or by simply attacking the partner and attacking our organization from there. A chokepoint that controls 100 access points will be very difficult to properly secure.

For small organizations with few network connections, consolidating all access though a single chokepoint is probably the best option. However, for larger organizations, it may be desirable to separate access between multiple chokepoints, keeping higher risk enforcement policies away from more trusted ones.

## A Note on Singe Points of Failure

One inherent problem with chokepoints is the tendency to introduce a single point of failure into the environment. If a component running a chokepoint service were to fail, the effects would be far more dramatic than if a component controlling a single access point failed. As such, it is important to increase the availability measures taken in relation to the number of access points consolidated. If, for example, we forced the Internet, our partners, and all dial-up traffic though a single chokepoint, we would most likely desire to add a level of redundancy by introducing a redundant chokepoint.

## Applying the Chokepoint Concept

Here are some simple steps to take when contemplating chokepoints:

1. Identify all access points to a particular resource or related set of resources.
2. Consolidate all such access points though a single security object.
3. Enforce tight controls, monitoring, and redundancy on that security object.
4. Establish a policy for future access points, stating that they must be filtered through an approved chokepoint.
5. Continue to test and scan for new access points that do not filter through a chokepoint.

## LAYERING SECURITY

When looking at security architecture, it is important to recognize that no single device is without flaws. *Every significant application, server, router, and firewall on the market harbors some vulnerabilities.* Additionally, all devices have a good chance of being misconfigured, unmonitored, and improperly maintained. On their own, each object will eventually become the weak link that allows a hacker into the network. This understanding is what leads to the expression: "Nothing can be 100% secure."

If nothing can be 100% secure, then it would certainly not be wise to trust any one device with all security. The firewall, for example, should not be the only thing guarding a perimeter network. Always consider the fact that a security device will have some flaw in it that will ultimately be an exposure to attack.

### Basic Security Layering

Rather than focusing on any specific device or application as an all-in-one security solution, we must look at security in layers. If one device succumbs to an attack, another device should be there to save the organization from exposure. In most situations, we should consider at least three layers of security (see Table 5.2):

- *Internal layer*—The internal layer consists of controls that are applied directly to protected objects. If an attacker penetrates the outer layers, he or she should still not have complete access to internal resources. examples of this type of control include a host-based IDS on a server and a locked cabinet inside a computer room.
- *Middle layer*—The middle layer consists of the primary security devices, such as firewalls and the front door to the server room. This is the main line of defense against outside intruders.

- *External layer*—This consists of controls that protect the middle layer and help to protect us in the event that the middle layer fails. Example external layer devices include a screening router that protects the firewall and a gate outside a building.

**Table 5.2** Examples of The Three Layers of Security

| Scenario | External Layer | Middle Layer | Internal Layer |
|---|---|---|---|
| Perimeter network | Screening router | Firewall/IDS | Server-based controls |
| Physical security | External gate | Front door | Internal locked cabinet |

## Layering Network Security

It is common to practice layered security within networks. Perhaps a hacker cannot gain access to internal systems because he or she is stopped by a firewall (the middle layer), but the firewall itself can be vulnerable to attack. Thus, we also implement measures to protect the firewall from attack, and to maintain security even if the firewall is compromised. Rather than just placing a lock on the castle gate, it is best to build a moat to protect the castle and the lock itself. A thief is far less likely to pick the lock if he or she must first swim a moat filled with hungry alligators. Similarly, we can force the thief to pass through multiple locks and multiple moats before access is granted. Once he or she passes the front gate, the thief will find that the treasure room also has a lock on it.

We will look at this concept more in the section titled, *Perimeter Defenses* in Chapter 11, *The Rules in Practice*. For now, consider the following tips for layering network security:

- Start by having a firewall separating internal and external networks (middle layer). Enhance this by placing other security mechanisms, such as an IDS on the network between the firewall and Internet.
- Program an external router to limit access to the firewall and internal systems (external layer). The router should perform some minor sanity checking to catch any obvious attacks.
- Terminate all remote access services outside the firewall. Many dial-up and VPN vendors place security directly on remote access devices themselves, and recommend placing them in direct contact with the internal network. Doing so removes the essential middle layer and is a bad security practice. Make sure that such devices have security enabled, but are placed outside the firewall.

- Enforce strong internal security controls (internal layer). Make sure that if someone breaks through the perimeter and attacks the network, there are additional defenses. Controls should be placed on individual systems, services, and devices.

## Layering Systems Security

When possible, systems and devices should have some form of layered security. There are many resource control options within most operating systems and applications, making layered security possible. An intruder may be able to attack an operating system, for example, but critical resources are still controlled by the application. Even better, if we follow our practice of zoning, we can separate critical data from externally accessible services. By doing so, we can apply security on the both front-end and back-end systems:

- Apply front-end controls, preferably ones that are centralized for multiple applications (like chokepoints). These controls should be the primary line of defense, protecting background data and services.
- Apply security that protects the front-end. For example, provide a network filter to limit Internet Protocol (IP) ranges and ports to help protect the front-end from being attacked.
- Apply security directly to back-end data and services. This includes direct controls for protecting the DB, filesystems, and operating system.

## Layering Physical Security

Physical security should always be constructed in layers; the more controls that can be layered, the better the ability to control and monitor access. This can include simple layers, such as a locked front door, a locked server room door, or a locked cabinet within the server room; or, this can be complex, with stationed security guards, cameras, and other forms of access control in each area. The following guidelines will help in establishing physical security through layering:

- Protect your sensitive equipment and resources via strong centralized controls. Try to consolidate such controls by limiting the physical areas where equipment is stored. Locking doors, alarm systems, and cameras should be considered in this main area.
- Use physical protection that prevents unauthorized users from even trying to gain access to critical equipment areas. Hallways outside of sensitive rooms should be restricted to authorized personnel. If possible,

cameras and other security measures should be placed in areas that lead to entranceways. At a minimum, staff should be trained to question any unknown person who is near an entry point to a sensitive area.

- Controls should be placed on sensitive objects within protected rooms as well. Critical servers and devices should be locked in cabinets. Someone gaining unauthorized access to a room, or even someone with authorized access, should not necessarily have access to all objects within the room.

### Applying the Concept of Layered Security

The following steps will help when contemplating layered security within your own environment:

1. Take any object and apply as much security directly on the object as is reasonably possible.
2. Consider the access points to the object and apply as much security between the subjects and the object as is reasonably possible.
3. Consider all the object's dependencies, including operating systems, third-party services, etc., and apply security to each. This should be performed for both the object itself and any security mechanisms protecting the object.
4. Make sure the object itself and anything guarding the object are monitored and generate access logs. If one object is compromised, secured logs should exist elsewhere on a secured device.
5. NEVER consider an object safe simply because another object is protecting it. NEVER forgo directly applying security on the object assuming no one will ever be able to attack it.

## WORKING IN STILLNESS

In information security, we must always be "listening" for our enemies. When someone steps across a security boundary, a log should be generated and an alarm should go off, spurring us into action. Such logs and alerts are vital elements to our overall security practices. Just like in the physical world, however, *it is impossible to "hear" the enemy if there is excessive surrounding noise to confuse us*. We will not notice the hacker tripping over a security checkpoint when our logs are full of millions of unimportant activities.

*I have seen many organizations make a considerable investment in devices that alert them about anything and everything going on within their environment. Sadly, such security measures create excessive "noise," resulting in thousands of logs and alerts that need to be searched. Meanwhile, the hacker quietly slips in and out of the treasure room, his or her footsteps hidden in the chaos.*

In security we listen; but for there to be noise, there must first be stillness. Security strategists from all ages have used silence as a key tool of defense. So much of security relies on the ability to detect enemies as they attempt to enter the castle. Thus, before we learn to combat our foes, we must first learn how to create silence within the environment.

## Creating Stillness

Creating stillness requires a level of silence at every location where a security check takes place. This involves the art of alert filtering, making sure we only hear that which is suspicious. When a new security device is put in place, a new operating system is installed, or a new application is created, it must go through a "noise-tuning process." The tuning process should include some post-installation tweaks and adjustments, one of which should be log reduction to provide filtering. Here are some simple guidelines for tuning an object:

**Five Steps for Tuning Silence**

1. Study the logging and alerting features of an object. Every device and application works a little differently and weighs security events in its

own unique way. It is important to understand the classifications and severities of different events, as well as the actions that trigger them.

2. Initially set the logging and alerting mechanisms to be sensitive, thus generating a substantial number of logs and alerts. Let this run for a few days and note the types of alarms and logs that are naturally generated within the environment by "normal activities." You will most likely see many types of traffic that the security devices will consider "suspicious," even though they are harmless.
3. Adjust the settings and apply filters to remove common acceptable events. It is important to be as specific as possible and to not filter out so much that you miss a hacker performing malicious events with similar qualities.
4. Once "normal" events are operating silently, other abnormal events should begin to show up. These events may not be malicious, but they may not be authorized. Commonly logged activities include devices with misconfigured network settings, pointing them outside the local network, and broadcast services like print servers. In cases where traffic is unauthorized but not malicious, it is important to stop the event from happening at the source, rather than filtering the logs and allowing the action to continue. When possible, go to each device and make changes to stop unauthorized activities.
5. Once all "normal" activities are filtered and abnormal activities are stopped, we are ready to start monitoring. It is important to document any filters that were put in place to ignore "normal activities." Others should know what is not being monitored in case there is ever a security issue related to such an activity.

Once all the common events are removed from the environment, we will be much more likely to notice strange events that may indicate an attack. Having finished the tuning process, it will become a part of our daily effort, applying proper filters as new log-generating activities occur.

## Tiered Silence

While it is important to maintain silence at security checkpoints, it is also important to maintain a thorough and accurate record of the events taking place. Many of the events filtered out of the logs could potentially play an important role in a future investigation; do we really want to erase them? When filtering events from logs, it is best to not simply discard them, but rather to store them in an unfiltered archive, separate from normal viewable logs. In this case, logging is set to record

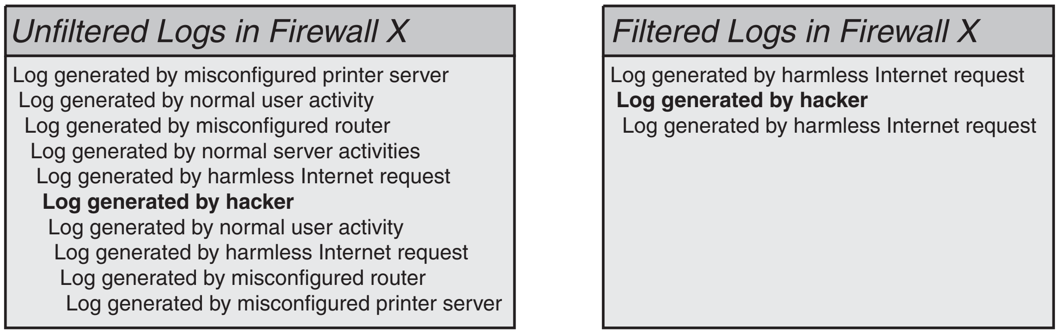

Unfiltered Logs in Firewall X
Log generated by misconfigured printer server
Log generated by normal user activity
Log generated by misconfigured router
Log generated by normal server activities
Log generated by harmless Internet request
Log generated by hacker
Log generated by normal user activity
Log generated by harmless Internet request
Log generated by misconfigured router
Log generated by misconfigured printer server
Filtered Logs in Firewall X
Log generated by harmless Internet request
Log generated by hacker
Log generated by harmless Internet request

**Figure 5.9** Basic log filtering.

many types of events that could potentially be of importance in the future. Only the logs that are the most significant, however, are held in the monitoring system; the rest are sent to an archive server. This way, we can filter out all the extra noise, but still have events on record in case we ever need to go back and do a detailed investigation. At a minimum, this should be performed with critical applications and servers, IDSs, and firewalls.

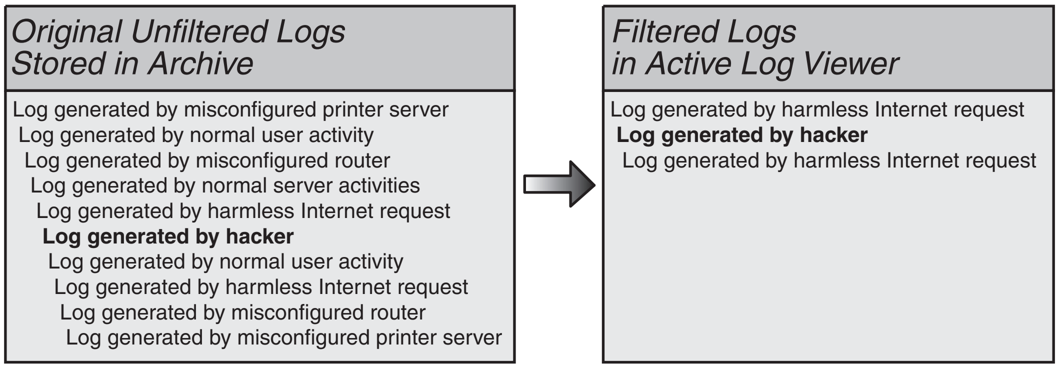

Original Unfiltered Logs Stored in Archive
Log generated by misconfigured printer server
Log generated by normal user activity
Log generated by misconfigured router
Log generated by normal server activities
Log generated by harmless Internet request
Log generated by hacker
Log generated by normal user activity
Log generated by harmless Internet request
Log generated by misconfigured router
Log generated by misconfigured printer server
Filtered Logs in Active Log Viewer
Log generated by harmless Internet request
Log generated by hacker
Log generated by harmless Internet request

**Figure 5.10** Filtering with log preservation.

## Striking a Balance

There is a delicate balance that must be struck when creating silence within an environment. If enough of the noise is not filtered out, an attacker's steps could be lost in a sea of extraneous logs and false alarms. If too much is filtered, the alerts will never show a hacker's presence. Here are some good higher practices to follow:

1. Watch for common events in logging and alerting systems and determine the cause of them. Who is doing what, where, when, and why?
2. Design a filter for such events, being as specific as is practical. Often, including extra conditions like the time of day for "normal activities" is important in your filters.
3. Before applying them, filters should be presented to management or others within the security group. Follow the *Rule of Change* (see Chapter 4) since configuring devices to intentionally ignore security events is a very important action. No one should perform such actions alone.

## UNDERSTANDING RELATIONAL SECURITY

Information security involves numerous chains and relationships. It is rare to have a security situation handed to us in a nice little box, isolated from the rest of the world. Any given object will almost always have a series of relationships with other networks, applications, events, etc., which will prove to be of great significance to our security considerations. The security of any object is dependent on the security of its related objects, and if we fail to see these relationships, we will be unable to properly address security. I call this *relational security*.

A server, for example, may be considered safe because it is not connected to the Internet. It is, however, accessible by the administrator's home computer through a dial-up session. The administrator's system itself is connected to the Internet through a DSL connection. Thus, by following this chain of relationships, the server is actually connected to the Internet (see Figure 5.11).

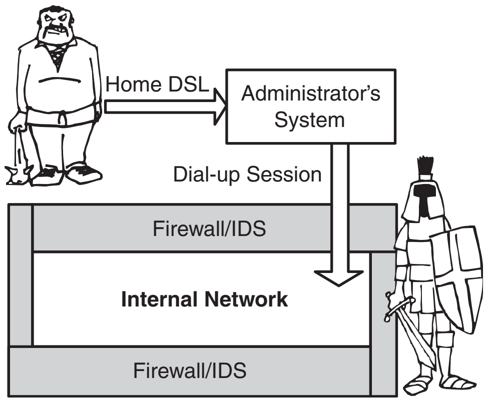


**Figure 5.11** *Security Relationships.*

Following such chains can point out where systems and networks that are considered to be safe are, in reality, vulnerable. Such relationships are commonly exploited by hackers and eventually lead to the compromise of many organizations. Here we will take a look at some of the most frequent problematic chains, including:

- Vulnerability inheritance
- Chained values and risks
- Chained trusts

## Vulnerability Inheritance

Probably the most vital and yet most neglected security relationship is that of vulnerability inheritance. The level of vulnerability within any object should be considered in relation to the vulnerability of its related objects. A file share between a secure system and a vulnerable system greatly diminishes the security of the secure system. If the secure system is accessible in any way from the vulnerable system, then, to some degree, it will inherit those vulnerabilities. This is especially true when considering chained paths of entry and chained trusts.

### Chained Paths of Entry

Everything in this world is connected in some way or another. We could, for example, draw a physical path between any two locations and any two objects. A castle treasury exists on the same physical ground as the thief desiring access to it. The two are connected by the air, by the earth, by the empty space between them, and by their common relation with the guards and night janitors. In security, we work to recognize and remove or defend such relationships.

In the Information Age, most computers and devices are connected to each other. If any computer is connected to a network or phone line, then we could literally follow a path of wires leading from the inside of my laptop to the inside of that computer. Even if some systems are not connected, we could draw a logical map connecting my laptop to another computer, and then from there to an unconnected computer via exchanged CDs and floppy disks.

Good hackers are continually thinking in terms of such chains. They take an object and think of everything that it is connected to: the Internet, the local network, the phone system, employees, a floppy disk, a printer port, anything. Given so many possible connections, a hacker will simply evaluate the easiest and quickest method of gaining access. Most often, a hacker does not even have to look for such relationships since they simply stumble onto them while exploring an environment.

*Modern worms are great at exploiting simple paths of entry. A Web server on the Internet is infected through an infected home user's client. Another client accessing that Web server is infected through the Web browser application when accessing a Web page. A partner is infected though an infected email sent from that client. The partner's boss's PC is infected via an open file share to the partner. The partner's boss's home system is infected by copying some files onto a floppy and taking them home. Then, the boss logs on to the Internet and it starts all over again.*

The basic point is that a distinct relationship exists between the majority of computers in the world, including systems that are not even connected to the Internet. Consider all theses common paths of entry that chain systems together:

- Direct or dial-up Internet connection
- Email, instant messaging, and other forms of networked communications
- Partner and vendor network connections
- Modems, remote access, and VPN connections
- Removable media like floppy disks, CDs, DVDs, and zip disks
- Employees that work with multiple systems
- New computers and equipment configured outside the local environment

Any object that has one of these possible paths (which just about every computer system does) is accessible indirectly by all of them. And such chains are not limited to systems or networks. Applications commonly consist of modules that are chained together, some directly accessible and some hidden in code. Many hackers spend a great deal of time exploiting a common application module and then search for all the applications that have integrated that module.

With this concept in mind, it is important to expand our scope of entry points beyond those that are obvious. A firewalled Internet connection does little if home VPN users do not have adequate protection on their PCs. When considering the security of an environment, or any situation, it is a good idea to draw a map of all significant and likely chained access points. Never consider any system safe simply because it does not connect directly to the Internet.

### Chained Privileges

When a subject is granted access to an object, that object now shares some degree of risk with the subject. The risk, however, goes far beyond the trust extended to that individual or system. To determine what level of risk we are really dealing with, we must look at the chain of access behind the scenes. Consideration should not only be given to the subject to which we are granting access, but also to the other subjects that have access to that subject. If X grants access to Y, and Y is accessible by Z, then X is somewhat accessible by Z as well.

We could, for example, trust John with access to our treasure chamber and give him the only key. John, however, is accessible by a great many people. His wife will have access to the key if she so desires it, as well as his children and his best friend. John could get mugged on his way into work and thus a mugger would have access. When we give John access, we are giving some degree of access to all of these people.

Similarly, an application that creates and runs an administrator-level process on a server has complete access to the system. This might be considered acceptable because the process is running for a benefit. However, by granting such a high level of privilege to the process, we have also granted privilege to everything with access to that process. When a hacker determines how to access the process through an exploit, he or she may be able to operate with administrative privileges.

Chained access is a big problem within modern organizations, especially with more and more ways to conveniently access systems. Home-based VPN connections are giant issues for corporations because the home system is also accessible by the employee's children, family, friends, and is most often connected to the Internet. Such relationships must be considered whenever making a security decision.

## Chained Values and Risks

Relationships are also important to consider when thinking in terms of values and risks. I will not spend a great deal of time discussing this here; we have a whole section on risk coming up. However, when determining where risks exist and what degree of risk is within each object, we must think in terms of relationships. A simple example would be a server room. When determining the degree in which we need to protect the server room and what risk this server room's loss would pose to us, we obviously need to think beyond the value of the room's construction. The room itself may hold minor value to us; whereas the room has many related objects that could suffer should the roof fall in. The room may have servers, routers, WAN links, and people within it; all of these come into consideration when we think to secure the room. This is true for all of objects, as we will see later.

## Chaining Trusts

"Trust" is one of those words that should instantly make one think of vulnerabilities and exposure. Of course, we must extend some level of trust to the world around us or nothing will ever get accomplished. It is also true that many day-to-day activities can be handled much easier when we are able to trust someone or something. With every degree of trust, however, there is a degree of exposure that must be considered. Unfortunately, such exposures are often hidden in what I like to call a chain of trust.

When receiving a file through an email from an old friend, there is little concern that he or she would desire to infect our system with a virus, or send us malicious code of any kind. The immediate response is to run the executable he/she sent without question because we trust that individual to not have any motivation to try and attack our system. It is not, however, the trust of that friend with which we must concern ourselves. The fact is that when we trust our friend with this file, we also trust a great many other people with whom we have no formal relationship. We trust, for example, the person who sent him/her that file in the first place to not have put a virus on it. We are also trusting all the applications that our friend has executed since he or she installed the system, because any one of them could have infected this file with a virus. Thus, we are trusting a great many people, applications, and events that could potentially lead us to installing a virus on our system.

### Trust in Networking

Most organizations have a lack of concern when attaching their own networks to the networks of trusted partners and vendors. The security concerns of the Internet seem far too obvious, while the concerns of hooking up the network of a local supplier are minimal. Similar to the file sent by the old friend, however, the trust extends far beyond our partner. Any given partner will probably have other connections into other networks as well. They may, for example, have a connection to the Internet, in addition to connections to other partners, customers, and vendors. Are we willing to trust all of these connections as well? Before any foreign networks are connected, it is important to think about the potential chains involved. As a general rule, no external entities should share connections with a local organization unless filtered through a security device.

*Many years ago, I was working with a client in the research industry who connected a network to a trusted vendor. This connection had no firewall, only very basic router filtering. After performing an audit, it was discovered that this vendor had similar connections to every major competitor of my client. On top of that, this vendor had an Internet connection without firewall protection! Thus, my client was unknowingly connected to the Internet and all their major competitors without even using a firewall.*

## UNDERSTANDING SECRETLESS SECURITY

In everyday life, basic security often relies on some form of secrecy. If you have $1,000 in cash under your bed, you are reasonably safe as long as no one knows it is there. Likewise, if you have all your money in a safe and the combination is 35-21-02-31, you can be pretty sure that it will still be there when you wake up tomorrow morning as long as the number remains secret. These security solutions are all based on secrecy, a concept that makes up the most basic form of security possible. If no one knows about it, or no one knows how to gain access to it, then it is reasonably secure. The only problem is that secrets are difficult to keep.

Relying on secrets for security has several weaknesses. For example, secrets have a tendency to leak out. If you talk in your sleep or if you unlock your safe in front of a window, your secret can be easily compromised. Secrets can also be guessed. A thief may look under your bed while in your house, or he/she may steal your safe and spend hours guessing the correct combination. The basic point is that *secrets are very hard to keep secret*. If you magnify this problem by a few thousand end-users and several administrators, then you will probably be spending more time securing your secrets than securing your valuables.

### Secretless Security

*The best security solutions are those that rely as little as possible on secrecy for protection.* Our strongest forms of protection come from devices and applications that the entire world could know about, and yet would still remain secure. We should never assume that systems will remain secure simply because no one knows where they sit or what their addresses are. Likewise, we should never base security on the idea that a hacker would have to know a great many things about an environment to be able to break in. It should always be assumed that all secrets are going to be discovered. Let's take a look at some classic examples where secretless security is commonly applied.

## Open Encryption Algorithms

Many older forms of encryption relied on secrecy for protecting information. For example, the writer of an encrypted message would scramble up the words, and the reader would reverse the process to decode the message. If, however, someone knew how the words were scrambled, they could also unscramble them and thus, would have access to the information.

Most modern encryption is based on secretless algorithms. The computation that is used to scramble and unscramble a message can be known to everyone in the world and yet the encrypted information remains safe. The algorithm is published for the world to see, giving everyone an opportunity to find a flaw and break it. Bad algorithms are broken in a manner of weeks, while good algorithms remain secure.

The weakness of any encryption algorithm exists in its key. The key is the small component that is only known by people with permission to decrypt the information. Therefore, the focus of encryption is mostly concerned with protecting the key.

## Open Security Applications

A security application is at its best when the code for the application can be seen by everyone, and yet the application still remains secure. Applications that base their security on a secret provide a very weak level of protection in comparison. As has been proven time and again, the secret will eventually be discovered and the security will be rendered useless. Good security applications never base their security on secrets. I will discuss this more in the section titled *Open Source vs. Closed Source Security* in Chapter 10, *Modern Considerations*.

## Secretless Authorization

Secretless authorization has been an emerging trend in information security over the past several years. With the dismal failure of secret-based solutions like passwords in protecting large enterprises, many organizations have implemented alternate approaches. Advanced authentication no longer bases its decision on something you know, but also on something you have or something you are. It is much easier, for example, to fake someone's password at an authentication prompt than it is to fake their eye pattern during a retinal scan. I will discuss this further in the section titled, *Handling Authentication* in Chapter 11.

### The Necessary Evil of Passwords

Passwords are the most common form of security based on secrecy. Unfortunately, passwords are everywhere, including at the very start and end of an encryption process. Keeping a large number of passwords secret is extremely difficult if not impossible for large organizations. A high percentage of hacker attacks begin with the use of a stolen or guessed password, making these things one of the biggest problems in modern security. Chapter 11 will discuss the password problem in detail; for now, we will address it as yet another area where security based on secrecy fails. It is important to recognize the incredible weakness inherent in secrecy-based security mechanisms such as passwords. Organizations that must rely on passwords for security should go to great lengths to secure them from unauthorized access.

ZDNet published a good article on the secret password epidemic called "Psst… I Know Your Password"; you can find it at: http://zdnet.com.com/2100-1105-920092.html

## DIVIDING RESPONSIBILITIES

A primary rule for doing business is to never put all your eggs in one basket. Never have all your investments in one industry; never rely on a single individual for performing critical processes; and never, *never* assign all security responsibilities to one employee, one system, or one process. And, if you are a security professional, make sure you are never the one with all the responsibilities and power.

Two things about us humans: We are very curious and we are very good at deception. Earlier I talked about "The Rule of Trust," which ties very closely to this concept of division. Since we should never fully trust anyone with our security, we must make sure that everyone is subjected to the same degree of security checking. This includes our security professionals, our managers, and our consultants. These individuals should have to request access, be required to authenticate, and have their actions restricted and logged just like everyone else. Anyone who is not restricted by such measures is a security threat to our environment, even if "he's/she's the nicest person in the world." A common mistake organizations make is to put complete power into the hands of their best employees. If Jane is the smartest woman in the world and can fix any problem in a matter of seconds, we have a tendency to give her access to everything and monitor nothing that she does. This, however, presents an all too fatal hole in our security model and will often lead to major security problems.

*During one consulting engagement, I was brought in because the head UNIX administrator was starting to cause problems. Apparently, he did not receive the raise he desired and began to act strangely and make threats. Unfortunately, this individual had access to everything UNIX-based and was also the employee enforcing and checking security. Before anything could be done, the individual erased the entire DB storing client data gathered over thousands of hours. Much of the information was time-sensitive and could not be recovered.*

Separating responsibilities does not stop with personnel, however. This concept applies just as strongly to placing all our faith in one security application, or one security device. If Server X is the only thing protecting our entire company, performing filtering, content management, intrusion detection, and authentication, and running our VPN and logging, we have a security issue. No system is perfect, and no security device is unbreakable. At a minimum, we should always have something monitoring and protecting the security of our main security devices.

## *Practicing Division of Responsibilities*

Dividing responsibilities requires that we follow some standard management practices:

- **Maintain redundant staff—**Even if you are not budgeted for two people trained in a specific area, make sure that there is a designated backup employee who can take over another security employee's primary responsibilities.
- **Monitor everyone equally—**Ensure that any security measure applied to the organization is either universally enforced or has some equivalent security measure applied to the administrators and security staff. If the security staff monitors everyone's access, make sure you have someone else in charge of monitoring them. This does not mean you have to have a duplicate guru in every area of practice, just someone who is capable of keeping an eye on the other employees. Always follow the Rule of Trust, especially with more powerful staff members.
- **Enforce security rules on everyone equally—**Everyone should be made aware of the rules and the fact that no one is an exception. In many organizations where Administrator X was found knee-deep in sensitive information, it is all too easy to give the excuse that administrator privileges

were needed for various reasons. If, however, there are standard security measures to get through, then he/she is unlikely to bypass the required clearance. If such clearance is bypassed, then at least there is an indication and evidence that something is going on.

- **Always follow layered security practices (discussed earlier)—**If the firewall fails, what then? Will everyone be allowed into the network? Will we even know? Every security device should have some small external component assisting in security. Firewalls, for instance, should be behind screening routers. Even if the firewall was compromised, the router would still only allow specific traffic through. The firewall, as with all security components, should be reporting logs back to a central logging server that is separate from the firewall itself. This logging server should be able to tell when a system has been compromised, and should be capable of monitoring the important security devices.

## FAILING SECURELY

You may have noticed that most of the examples of security issues I provided in this chapter involved some devices or applications that failed in one way or another. Hackers commonly use exploits that cause services to fail due to unexpected events. Most exploits are simple scripts that cause services to crash and open security holes. The worst examples are services that run as administrator and, when successfully attacked, give up control and allow the attacker to become the administrator.

Many times, the failure of an application, networking service, or operating system can be performed gracefully. When dealing with critical DB servers, for example, failures usually trigger events that attempt to leave the data in a usable state. Similarly, a firewall that detects a failure will oftentimes shut down access services to avoid allowing unauthorized access from outside entities. This is what is called *failing securely.*

Everything is subject to failure no matter how robust or expensive it is. Such failures often lead to lost productivity and potential security issues. As such, potential failure scenarios should be considered before any new implementation. When programming an application, failures should be made to lock down security. When a network architecture is designed, failures should not result in bypassing security as is commonly done. If a power outage occurs, services, applications, and devices should apply security during the reboot process. Consider failures in all devices and services, walk through the contingency plan, and consider the security implications therein. This is especially essential for major failure plans like disaster recovery policies.

I have had many clients implement an expensive firewall and intrusion detection architecture to protect their Internet connections and remote vendors. Knowing that a router or firewall may fail, however, they implement an inexpensive ISDN or DSL connection to act as a backup. This backup hooks directly into the network, bypassing the firewall and security. Many attacks are designed to disable a firewall or network devices, wait for an unsecure failover to take place, and then take advantage of the unlimited access.