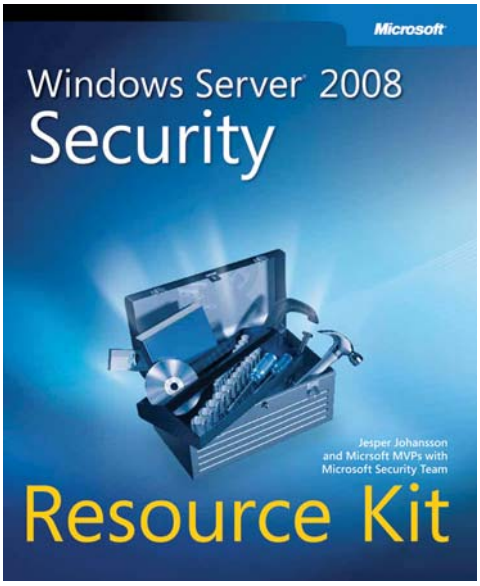

Windows Server® 2008 Security Resource Kit

Jesper M. Johansson and MVPs with the Microsoft Security Team

To learn more about this book, visit Microsoft Learning at
http://www.microsoft.com/MSPress/books/11841.aspx

Microsoft® Press

978-0-7356-2504-4

Table of Contents

Dedications

Introduction

Part III: Common Security Scenarios

Chapter 14 Securing the Network

Introduction

Truth, at Last

Introduction to Security Dependencies

Acceptable Dependencies

Unacceptable Dependencies

Dependency Analysis of an Attack

Types of Dependencies

Usage Dependencies

Access-Based Dependencies

Administrative Dependencies

Service Account Dependencies

Operational Dependencies

Categorizing Dependencies

Mitigating Dependencies

Step 1: Build a Classification Scheme

Steps 2 and 3: Networking Threat Modeling

Step 4: Analyze, Rinse, Repeat as Needed

Step 5: Design and Isolation Strategy

Step 6: Derive Operational Strategy

Step 7: Implement Restrictions

Minimize Account Scope

Define Organizational Policies

Separate Service Accounts

Manage Privileges

Restrict Communications

Restrict Access to Resources

Summary

Additional Resources

Part III

Common Security Scenarios

Chapter 14

Managing Security Dependencies to Secure Your Network

Jesper M. Johansson

I am often asked how to protect workstations on a network. More specifically, the question is framed against the latest attack-du-jour that was demonstrated at some conference. For example, many people are extremely concerned about USB Flash Memory—those incredibly handy little finger-sized, solid-state memory devices that are now available in capacities bordering the ludicrous. People are worried about an attack that starts with the attacker inserting a USB Flash Drive into a computer, or causing the user to do so. The USB Flash Drive is laden with malware that either automatically—or with minimal user interaction—executes malware on the computer.

The problem with this preoccupation with USB Flash Drives is that it is an extremely narrow view of a much larger removable-device problem that also includes CDs, DVDs, FireWire drives, parallel port devices (does anyone still have these?), and just about any other orifice on the computer that can be used to access external content. Too often, people are only worried about workstations and not the rest of the network. I believe that the question is not how you keep workstations from getting hacked, but how you keep the rest of the network from falling like dominos once they do. Let's look at the math. If you have 10,000 end users in a network, what are the chances that you can keep all the workstations secure? Let's assume that each of those workstations is up to date with security updates, fully managed, and operated by users who are savvy enough about security to not run malicious content 99.99 percent of the time. Ignore the complete unrealism of these numbers for a moment and focus on the math. With 10,000 workstations, these numbers mean you have a 37 percent chance of having a secure network at any given time. With 20,000 workstations, your chances are about 13 percent. Add in a more realistic probability of each of your workstations being secure, and you will find that the probability of keeping all of them simultaneously secure asymptotically approaches zero as your network grows in size.

Clearly, it is absolutely critical to protect the network as a whole from the compromise of a single workstation. In fact, as an IT manager, I argue that no single thing you can to do improve the security of your operational environment is more crucial than managing dependencies in your network to isolate exposures. One part of this is to restrict communications within your network; Microsoft calls this Server and Domain Isolation. However, that puzzle has some more pieces.

Direct from the Source: Server and Doman Isolation

Server and Domain Isolation is one of the hidden security gems in Windows Server 2008 that's worth taking a closer look at. Although you had the ability to create virtual networks through end-point authentication in previous releases, the work we've done in Windows Vista and Windows Server 2008 makes this even easier to deploy. By

combining IPsec connection security rules with Windows Firewall filters we have given Windows Server 2008 administrators a powerful tool to increase the security on their networks and better safeguard their important data—and all of this can be done without installing new software.

Chris Black, Program Manager, Windows Networking

Many organizations are still overconfident that their perimeter firewall solves most of the puzzle for them. However, perimeter firewalls are virtually meaningless in today's environment. To understand why, take a moment to try to enumerate the entry points into your network. If you have a medium sized or larger network I am willing to bet a very good dinner that the last audit you had found a few egress points you were never aware of. Every computer on a virtual private network (VPN) is a potential ingress point. Every system that you have not updated is a potential ingress point. Every insecure, custom-written piece of software is a potential ingress point. Every misconfigured router, firewall, VPN device, and wireless access point is a potential ingress point.

> The principle of Defense in Depth simply requires that you put significant effort into reducing the impact of a compromise on your network.

One obvious method for addressing the problem of malicious removable devices is to ban everything with the potential to be malicious. This includes more than just USB Flash Drives. We would need to ban all removable devices, including anything that plugs into any device bus in the computer. I've said before that the best way to handle that problem is to use a giant tube of epoxy to plug up every opening you find on the back, front, sides, top, and bottom of the computer.

This approach has a couple of problems. First, your users might ambush you on the way to your car and perform ritual sacrifice on you if you do. Second, they would be right to do so. Many of the aforementioned devices serve legitimate business needs. For instance, it is pretty much universally accepted that the most secure configuration of the BitLocker full hard-disk encryption technology in Windows Vista is to use an external key on a USB Flash Drive. Doing so would be rather difficult if you filled the USB ports with epoxy. I suppose you could glue the USB Flash Drive into the port, but that would sort of defeat the whole purpose of using it for encryption key storage. The same argument can be said for most ports on a computer these days.

The better option, in all but the most sensitive environments, is probably to attempt to manage the risk and contain the exposure. We need to accept a fundamental truth here. The general statement in Law 3 of the "10 Immutable Laws of Security" (see http://www.microsoft.com/technet/archive/community/columns/security/essays/10imlaws.mspx?mfr=true) still holds:

> If a bad guy has unrestricted physical access to your computer, it's not your computer anymore.

If an attacker has—or has ever had—access to your computer, that computer must be considered compromised. This kind of attack can even be perpetrated remotely, if the attacker can get you to run malicious code on your computer. Law 1 from the Immutable Laws states that:

> If a bad guy can persuade you to run his program on your computer, it's not your computer anymore.

If we take it as a fact that the immutable laws still hold—and we probably can because they have proven to be remarkably resilient, and it is unlikely they will be proven invalid in any significant way until we fundamentally change how computers work—we cannot rest with a few registry tweaks to reduce the threat posed by removable drives. Clearly, we must use additional layers of protection. In fact, if we simply make the quite reasonable assumption that many of our client computers are either already compromised, or operated by people who do not always have our best security interests front most in mind (or both), we arrive at the conclusion that we need to mitigate their effects on the remainder of the network. This leads us naturally to understand, analyze, and mitigate security dependencies.

Security Alert: On the Efficacy of Security Guides

For the past 15 years or so an unbelievable amount of effort has been devoted to building security guides. I have taken part in building about half a dozen of these over the years. These guides invariably a list of various security tweaks that— according to the authors—you must make to a standard installation of some software to meet some security requirement. The requirement itself is far too often unstated, and many of the guides are merely listings of every possible tweak that the authors thought might have even the most marginal impact on security; most of the time without considering the functionality your computers need to provide or threat environment your computers face. Often the settings recommended by the guides don't actually work on the software the guide is intended for.

The best of these guides make it very clear what the settings do, what application compatibility impact you can expect from them, and what specific threats they mitigate. Yet even the best of these guides spend scant space on the problem of network security at large. The guides are invariably focused on hardening a single computer against attacks, not fully accounting for the environment that computer is deployed in. The fact of the matter is that once the attacker has a foothold in the network, not a single setting in the security guides matters. The fact that account lockout is set to infinite lockout after three bad guesses—aggravating every user in the process—makes no difference to the attacker that has administrative privileges.

> Rather than focus your efforts on which tweaks you need to make to your computers, you will get a lot more mileage out of simply accepting that some portion of your network is, and always will be, untrustworthy. The reality is that Enterprise networks today are semi-hostile at best. Let's accept that sad state as fact and move on. We deal with that problem by protecting the network as a whole from the few bad elements. You cannot secure a society by setting down rules and a sturdy wall around the society. You also need police officers. Police officers are basically a function of society's acceptance that some portion of its members refuse to live within the boundaries that have been set for them. Your network is no different.

Introduction to Security Dependencies

A **security dependency** occurs when the security of one computer is dependent on the security of another. This is quite common, and in many cases desirable. For instance, you might have heard that "if your domain controller (DC) has been hacked, your entire network has been hacked." This is a simplistic way of stating that all domain members are dependent on the DCs for their security. If the domain controller is not kept secure, the member computers cannot possibly be kept secure. An attacker who can change the security configuration of the domain can take over any computer in the domain—for example, by adding new accounts to the Administrators group on a member computer. This explains why any so-called vulnerability that allows a system or network to be compromised by an administrator is not really a legitimate security vulnerability. That's because an administrator, by definition, is supposed to have complete access to the system or network he or she is administering.

Dependencies in computer systems are clearly unavoidable. However, that does not mean that they are all acceptable: Some are acceptable and even desirable, while others are unacceptable. Before we analyze the different types of dependencies and how to mitigate them, we need to understand which types of dependencies are acceptable and which are not.

Acceptable Dependencies

Acceptable dependencies can be summed up by the following statement, from *Protect Your Windows Network*:

A less sensitive system may depend on a more sensitive system for its security.

Computers—and systems in general—can be divided into classes based on their security sensitivity. A system that is more sensitive has higher security requirements. while one that is less sensitive needs less security. The specific set of classes in any particular environment is irrelevant to the general discussion; only the fact that there are inherent classifications is important. For the sake of argument, let us assume that we have two classes of systems: workstations and DCs. The DCs, obviously, are far more sensitive than the workstations. If you control a workstation, theoretically you should have access to only the data used on that workstation. However, if you control a DC, you have the keys to the kingdom—you have complete access to everything in the forest. In that case, it is acceptable for the workstations to depend on the DCs for their security. The DCs class is far more sensitive

than the workstations, and must be correspondingly better protected. This is a form of an acceptable dependency.

The same argument can be made for user accounts. It is acceptable for an administrator to compromise data owned by a user. This is what it means to be an administrator in the first place. Administrators have unfettered (although not always direct and obvious) access to the computer and everything on it. If we understand that and manage the computers appropriately, this is not a problem.

Software can be analyzed the same way. A less sensitive piece of software, such as a Web browser, may use and depend on a more sensitive piece of software for its security, such as the operating system itself. That is acceptable. If the operating system has a bug, the fact that the Web browser is now vulnerable to some new problem is really not surprising and is probably rather low on the list of worries. This also helps us understand where bug fixes go. The bug should be fixed as close to the problem as possible, to have the maximum protective impact. Rather than work around the problem in the Web browser, fix it in the operating system. Alternatively, rewrite the Web browser to reduce its dependencies on functionality in the operating system. This latter approach is appropriate if the functionality in the operating system was never intended to be used in the way it is being used, or if the functionality is not designed to protect against the particular attack the Web browser is suffering from.

Unacceptable Dependencies

Unacceptable dependencies should by now be obvious. Again, quoting from (Johansson, 2005):

A more sensitive system must never depend on a less sensitive system for its security.

If we again think in terms of classes of sensitivity, this statement is easily understood. If a compromise of a workstation means that the domain controller's security has been breached, we have a serious security problem on our hands. As mentioned earlier, it is impossible to protect a network if its aggregate security is dependent on the security of every single computer in that network. The likelihood that the network is secure is inversely exponentially related to the size of the network. A network of any reasonable size is, for all practical purposes, never entirely secure. This makes it paramount that more sensitive systems are protected from less sensitive ones.

This argument can easily be extended to user accounts and software. For example, the new Terminal Services client for Windows permits storage of user names and passwords for virtually transparent Terminal Services logon. Those credentials are stored using the Credential Manager API, protected by the credentials used for the primary logon session.

To see how this can create a security dependency, let us analyze the case of a network administrator logging on to his personal workstation. He uses this workstation for e-mail, Web browsing, and other typical information worker tasks. Naturally, he uses a low-privileged domain account for this purpose. At some point during the day he connects to one of the domain controllers to perform some form of management. He uses the Terminal Services client to do this, and elects to store his password to make future connections easier. This results in at least one, possibly two, unacceptable security

dependencies. The first is that his domain administrative account credentials are now protected by his low-privileged information worker credentials. If his low-privileged user account is compromised, his domain administrative user account is also compromised, and thus the entire domain is compromised.

The second dependency results from the fact that he typed a domain administrative credential on a non-domain controller. Unless his personal workstation is protected at least as well as the domain controllers—and that it is hard to believe—we have a dependency situation in which the security of the domain controllers depends on the security of this user's personal workstation. If, for example, a disgruntled employee in the same office has installed a hardware keystroke logger on the network administrator's workstation, the domain administrative credentials are now stored on that keystroke logger. Any time you type a domain administrative credential on a non-domain controller you have exposed to entire domain to any security flaws on the non-domain controller. For instance, if an attacker inserts a removable drive into a computer where a Domain Administrator is currently logged on, or has ever logged on, or will ever log on, that Domain Administrator is compromised, and by extension the entire domain is compromised. It is absolutely imperative that you understand how these dependencies work so that you can avoid letting them compromise your network. It means, for example, that you should be very careful which computers you use to administer sensitive computers in the network.

The foregoing analysis leads us to two very concrete pieces of advice. First, never use a computer to enter, retrieve, process, or store data that is more sensitive than the computer itself. Remember, everything piece of data handled by a computer should be considered accessible to everyone who has ever used that computer, or who will ever use that computer. Saving credentials on a computer whose every user you trust is safe. Saving them on a computer that may be used by untrusted users, or that may have malware installed at some point, is not, for example.

Second, never administer a sensitive computer from a computer that is less sensitive. Practically speaking, this means that you should have dedicated management stations used to administer ultra-sensitive computers, such as domain controllers. Simply using runas, or User Account Control (UAC) does not introduce a sufficient security boundary.

Obviously the same situation can happen with software. For instance, let us say we want to write a very secure Web browser. We want this browser to be far more secure than the built-in browser. In this case we cannot rely on any functionality provided by the built-in browser. In the case of Windows, where the browser implements much of the client-side, Internet-related functionality in the operating system, we cannot use any built-in Uniform Resource Locator (URL) validation functions or any Hypertext Markup Language (HTML) display functionality provided by the operating system, because those are really components of Internet Explorer. If we rely on functionality provided by the built-in browser, we have a security dependency on the built-in browser. Based on our stated objective of being more secure than the built-in browser, this dependency is unacceptable.

Dependency Analysis of an Attack

At this stage, it might be useful to take a quick detour and analyze an attack from a dependency perspective. Earlier we saw what can happen if a malicious removable drive is inserted into a computer. However, it may not be obvious what would happen to the network where that computer lives. Let's assume that the computer in question is domain-joined, as shown in Figure 14-1.

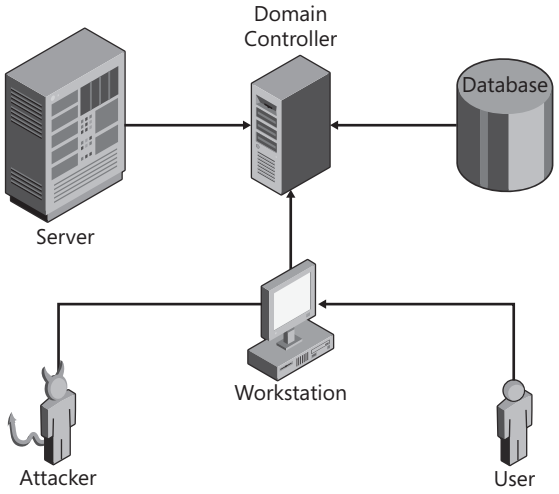

Figure 14-1 Domain dependency graph.

Figure 14-1 shows an ideal dependency graph. The arrows are directional and point in the dependency order: The security of the workstation is dependent on the security of the DC, and the security of the user is dependent on the security of the workstation. The attacker might be able to compromise the workstation, which would compromise any information the user has placed on that workstation, but the compromise would be isolated there.

Let us change the picture a little. Suppose the user logging on to the workstation is a member of the local administrators group on the Server. And suppose that a domain administrator frequently logs on to the server. We now have the dependencies shown with bold arrows in Figure 14-2.

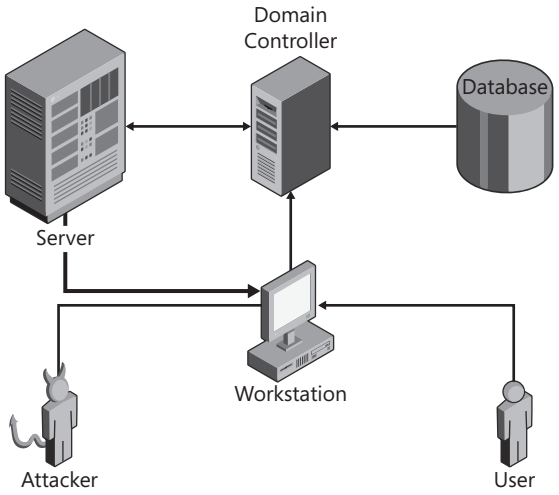

Figure 14-2 Domain dependency graph showing unacceptable dependencies.

As you can tell from Figure 14-2, we can completely violate the security of the entire network by simply changing the assumption of who logs on to which computer. Because a domain administrator logs on to the server, the security of the DC—and hence the domain—is dependent upon the security of the server. This would be acceptable if the server were managed as securely as the DC. However, a user that logs on to the workstation is a member of the Administrators group on the server, making the server dependent on the workstation for its security. Dependencies are *transitive*, which means that the security of the entire domain is now dependent on the security of the workstation, where the user, unfortunately, just ran the attacker's malicious tools. This is why it is so important to manage your dependencies appropriately.

Types of Dependencies

You need to manage many different kinds of dependencies. Some are beyond the scope of this book, such as dependencies inherent in software development. For instance, the security of code on a Web site is dependent upon proper isolation being enforced in the Web browsers that all the visitors use.

However, several different kinds of dependencies are relevant to a network, and in this section I will introduce them and discuss how to mitigate them using standard analysis techniques and actual implementation of these techniques in Windows Server 2008.

Usage Dependencies

The first and simplest kind of dependency is a **usage dependency**. A usage dependency results from usage of computing resources and data in a manner inconsistent with the trust levels of those resources. The first scenario in this chapter—the removable device—is an example of a usage dependency. A user that uses a removable device creates a usage dependency on that device. Whenever a user at one trust level uses a resource at a different trust level there are potential usage dependencies.

There are other kinds of usage dependencies as well. One great example is usage of a single credential in multiple places. For instance, suppose your network is divided into a datacenter forest and a corporate forest. All the users in the datacenter forest also have accounts in the corporate forest. The likelihood that at least one user will have the same user name and password on both of these accounts is extremely high. Yet this violates the entire purpose of having the two forests, which is to ensure that a compromise in one forest does not result in a compromise of another. By using the same password in both places, this particular user has opened a potential pathway between the two. An attacker that breaches a computer in one forest that this user is using can extract the password hash and use it to authenticate to resources in the other forest.

How It Works: Password Hashes Are Plaintext Equivalent

Virtually every computer system in existence today accepts passwords authenticators in at least some situations. On Windows Server 2008—as well as previous server versions of Windows—you can configure a domain to require smart cards for authentication from one or more users. However, as you saw in Chapter 4, "Authenticators and

Authentication Protocols," even when you do so, there will still be a password hash for the user. This hash is transmitted to the client each time the user authenticates to enable automatic access to NTLM-protected resources. This means that an attacker that has access to this hash can access network resources as this user. For more information on this, see Chapter 4.

Access-Based Dependencies

An **access-based dependency** occurs when a user at one trust level accesses a resource in a way that makes the user dependent on the security of that resource. Access-based dependencies result from the access itself, not from usage of a resource or computing construct. Many times they rely on one user or entity trusting another entity that has a security problem.

For example, suppose user Alice accesses a network resource. The network resource is on the server LOKE, which, unknown to Alice, was hacked by Bob earlier that same day. Bob has installed a rootkit on the server that causes authentication to be downgraded to an insecure form of authentication. Alice's computer is running Windows XP, which by default is configured to negotiate authentication to whatever the server and client can agree upon. In doing so, Alice sends a challenge-response sequence that the attacker can replay against Alice's computer, thereby gaining access to her computer with the same privileges she has. To understand this flow, look at Figure 14-3, which shows a normal authentication flow from a client to a server.

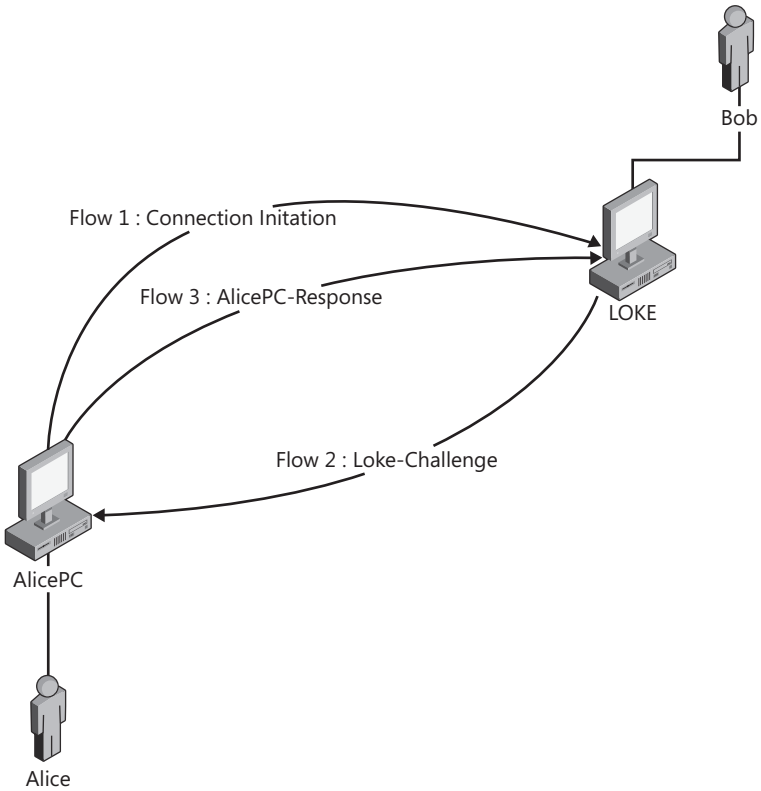

Figure 14-3 A normal challenge response flow from a client to a server.

In the normal flow a client initiates a connection to the server. The server responds with a challenge. The client creates a response to the challenge by performing a cryptographic operation with the authenticator (typically a password hash) and the challenge and returns this as the response. The server performs the same computation and compares the results. If they match, the authentication succeeds.

Now consider Figure 14-4. In this case the client does not respond as it should.

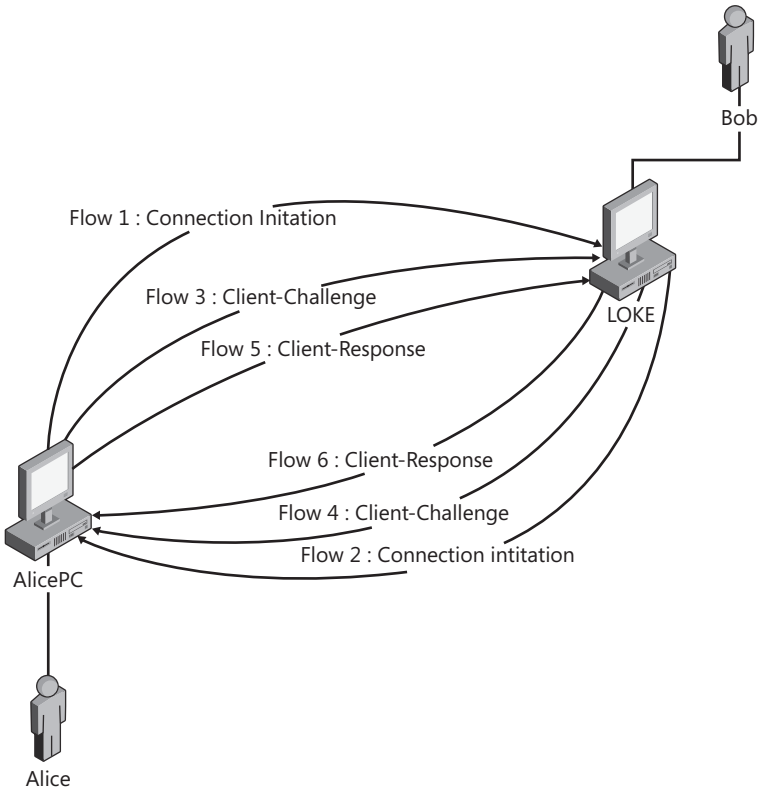

Figure 14-4 Using a reflection attack, the client can "reflect" the server's challenge back to the client to get a valid response.

In Figure 14-4 the client attempts to connect as before. At this point, the server is supposed to send a challenge back. However, the server instead responds with its own connection attempt in flow 2. The client responds to this connection attempt with a challenge (flow 3), which the server subsequently reflects back to the client as the challenge for the connection the client initiated (flow 4). The client now has the same challenge it originally sent back. Unaware that something is amiss, the client computes a valid response to this challenge, which it originally sent, and returns it to the server for the connection the client initiated (flow 5). The server takes this response and returns it as a response to the challenge the client issued for the inbound connection (flow 6). The net result is that we now have two successful connections—one from the client to the server and one from the server to the client. This is known as the **reflection attack**. In Windows Vista and Windows Server 2008 this attack is broken using stateful challenge management. The computer will no longer accept an inbound challenge that matches an outstanding challenge that it sent. In earlier versions the attack can be broken using various security settings.

This attack works because of an access-based dependency. There are other forms of such dependencies. A user might use a public kiosk to access e-mail by using Microsoft Office Outlook Web Access. Public kiosks are among the most malware-infested, untrustworthy, dangerous computers in the world today. Any resource you use on a public kiosk should be considered accessible to any user that has ever accessed that computer in the past or will ever access it the future. There is an access-based dependency between the security of the public kiosk—which you do not control—and any resource you have access to with credentials you use on a public kiosk.

Administrative Dependencies

One of the most common types of dependencies is an **administrative dependency**, which occurs when the same account is used to administer two different resources. For example, when you use a domain administrative account to administer member servers, as shown previously in Figure 14-2, you create an administrative dependency. This may sound a lot like a usage dependency, and it is. However, there is one important difference: administrative dependencies need not be usage-based. Let's say that the Administrators group on Server A includes Teddy, Maggie, and Alex, and the Administrators group on Server B includes Maggie, Jesper, and Jennifer. Maggie might never have logged on to Server A. However, Server B is compromised. When Maggie logs on to Server B, the attacker that compromised Server B has access to Maggie's credentials and can now use them to access Server A.

Service Account Dependencies

Service account dependencies occur when the same identity is used to run a service in multiple places. Suppose you use a network-wide Enterprise Management Solution (EMS). The EMS package includes an agent that runs on all computers to enable remote deployment of software, remote management, and all kinds of other goodness. The agent runs as the _DomainTools account. The _DomainTools account obviously needs to have elevated privileges on all the members to enable this type of remote management. This creates a service account dependency between all the computers where the _DomainTools account has high privileges. If any one of those computers is compromised, all of them are potentially compromised because the attacker now has access to a highly privileged account.

Operational Dependencies

Finally, we have **operational dependencies**. Operational dependencies result from the way a network is operated. For instance, Active Directory creates an ipso facto operational dependency. Any asset within a forest is dependent on the forest for its security. If the forest is compromised, so are all assets within the forest. The forest, in turn, is dependent on all domains in the forest. If a domain is compromised, so is the forest.

> **Security Alert**
>
> The forest, not the domain, is and has always been the security boundary in Active Directory.

Another very common dependency in a network is based on the software distribution system. Very often a single server or a set of Distributed File System (DFS) shares are used to distribute software to computers within the network. If an attacker compromises the server(s) that host the software, all computers that receive software from it are potentially compromised. The operational dependency has created an access-based dependency on the software distribution servers.

Categorizing Dependencies

As you may have noticed by now, the boundaries between the different kinds of dependencies are not always clear, and a single dependency can belong to several categories. For example, a service account used on multiple computers is both a service account dependency and an administrative dependency. The idea behind classifying the dependencies is simply to facilitate thinking about them. One type of dependency is not inherently far worse than another. Individual instances of dependencies might be more or less severe, but that is because of the facts of that instance, not the category of dependency it belongs to. Use the categories as a guideline to help you think about your network, not as a forced framework to plug things into. If you find a different taxonomy to be more useful, use that instead.

The only rule here is that everything you do should improve the security of your network.

Mitigating Dependencies

Finally, many pages into the chapter, we get to the part about how to solve the problem. It has taken this long because the concepts we have discussed so far are barely touched on in the vast majority of security literature, which often does not even mention these issues.

One of the most important techniques for mitigating security dependencies today involves isolating computers that do not need to communicate so that they cannot do so. Microsoft calls this Server and Domain Isolation. To build a strategy to do so is best done in a step-wise process:

1. Define a classification scheme.
2. Model your network.
3. Analyze your network model relative to the classification scheme.
4. Revise the classification scheme as needed and re-analyze.
5. Define an isolation strategy consistent with your risk management strategy.
6. Derive an operational strategy from your isolation strategy.
7. Build a server implementation based on your isolation strategy.

These seven steps are quite a bit more complicated than they might seem. The key is to realize that this is not a single-afternoon project. You really need a far better handle on the structure and usage patterns in your network than what most organizations have. In fact, if you get no further than simply understanding your network better, you have

created significant value. The remainder of this chapter discusses how to use these concepts to design and implement a Server Isolation strategy.

> Before we go any further, it is important to better define the term Server Isolation. When Microsoft first coined the term, it was in conjunction with the term Domain Isolation. **Domain Isolation** simply meant that to communicate with any domain member (with some exceptions) you had to be a domain member. This type of isolation is quite simple and, while valuable, leaves rather large holes by assuming that all domain members are good and nice.
>
> **Server Isolation** is the next step. In Server Isolation each server has its inbound traffic restricted, usually using IPsec, so that only the traffic necessary for the server to fulfill its business purpose is permitted. This provides very good isolation indeed.
>
> When Microsoft and other customers started implementing these isolation mechanisms they discovered that while Domain Isolation was simple in concept, implementing Server Isolation was far easier because IPsec was very difficult to work with in a large network. Therefore, they generally started with Server Isolation.
>
> However, what most observers fail to recognize about Server Isolation is that every Windows-based computer is a server. Every workstation also runs the erver service by default, and if you do not restrict inbound traffic—or even if you use Domain Isolation—you will have a network where every client can attack—I mean communicate—with every other client. Therefore, do not forget to include clients in your Server Isolation strategy.

Step 1: Create a Classification Scheme

The first step in building a server isolation strategy is to classify systems. You can think of network protection mechanisms as residing on a spectrum. Take, for instance, administrative accounts. One extreme of the spectrum is using one account for all purposes, on all computers, by all administrators. On the other extreme you have one account per administrator per task, with the least possible privileges necessary to complete that task per computer. While the former example might be practically possible, it would violate more security principles than we can list. The latter example, while highly secure, is intractable to manage and so cumbersome to use that it will likely be ignored by everyone involved. A similar spectrum exists for all other techniques. For instance, in terms of restricting communications, you can certainly analyze every single computer and restrict access to each one based on exactly what you need to use it for. However, in a network with many thousand computers, this is virtually impossible. You would be hacked long before you completed the analysis.

A far better option is to create a classification scheme. This scheme can be as simple or as complex as you need it to be. The idea is to divide your computers into categories that make sense to your business. Classifications can take many forms. In the military establishment it is common to have a two-dimensional classification scheme, such as that shown in Table 14-1.

Table 14-1 Military Classification Scheme

	Class		
	Unclassified	**Secret**	**Top Secret**
Compartment			Compartment 1
			Compartment 2
			Compartment 3

The military-style classification can be converted to classifying computers quite easily. One variant is shown in Table 14-2.

Table 14-2 System Classification by Role

	Class		
	Public	**Workstations**	**Server**
Compartment	Kiosks	Information Worker Workstations	Domain Controllers
	Infrastructure Servers	Developer Workstations	File Servers
		Admin workstations	Web Servers
			Database Servers
			…

Table 14-2 shows a subset of a computer classification based on the role the systems are fulfilling. No matter how you create your classifications, you almost certainly want to base them on the *role* the computer is fulfilling. The more granular you make the classification scheme—that is, the closer to a single role you can get—the more secure the resulting implementation will be. However, don't go overboard with this classification. First, you will probably need to revise it once you start analyzing your network and realize that you missed something and that some roles that do not cleanly make sense. Second, treat this as a risk management effort. If you are designing a classification scheme for an extreme risk environment, you want more granularity. If you are in a low-risk environment, you may be fine with a coarser system.

You may have noticed that one potential problem with using the two-dimensional classification system based on the military scheme is that you cannot neatly take into account the data that a particular computer of a given type is processing as well as the server type. For instance, not all database servers are alike. Some process highly sensitive personal information such as national ID numbers. Others hold public information, such as Web pages, that can be read by all users but written only by a few. Yet others servers may be entirely public and used simply as centralized temp folders. You can add rows to the classification for each computer type, but because many of the parameters you need to apply to computers are similar within a major type, this is not the cleanest method.

One way to accommodate sub-typing of computers a bit more neatly is to use a different modeling method. I like the organizational chart metaphor. It is infinitely extensible and permits easy sub-typing. You can, of course, use a more complicated modeling scheme, but because I find parsimony in your metaphor to be far more valuable than having hundreds of modeling constructs available, I tend to use simple modeling schemes. Using an org chart metaphor, we might come up with a picture such as the one in Figure 14-5.

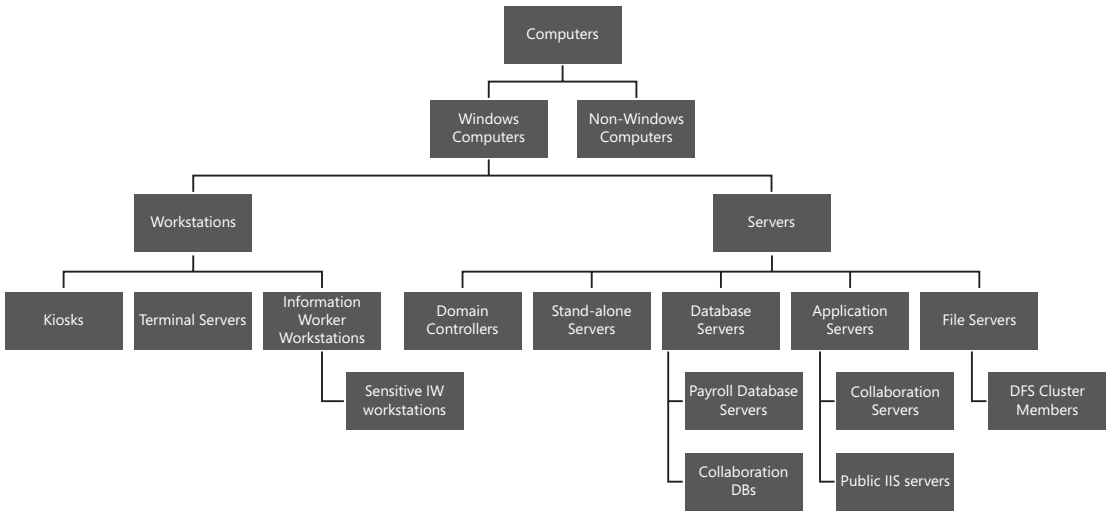

Figure 14-5 An org chart–style classification system is useful for complex environments where a lot of sub-types are needed to adequately express the security needs of the computers.

As Figure 14-5 shows, an org chart–style classification model can get rather large very quickly. Therefore, it might not be right for all environments. Note also that many of the categories are unlikely to have any computers in them. For instance, while Servers is a useful abstract super class, no computers should be assigned to it. All of them should be part of some specialization. However, when discussing server roles, as we did in Chapter 12, "Securing Server Roles," this type of hierarchical designation can be extremely valuable.

Once you have a preliminary classification model to start evaluating for fitness, you can begin your analysis. A useful technique for the analysis portion of the task is Network Threat Modeling, first described in (Johansson, 2005).

Steps 2 and 3: Network Threat Modeling

The next step is to see how well your classification model maps to the actual computers in your network. If you do not already have a map of your network, build one. It should detail everything important on your network, although you may group identical things together. The objective is to have something that lets you understand what your network looks like. Figure 14-6 provides an example.

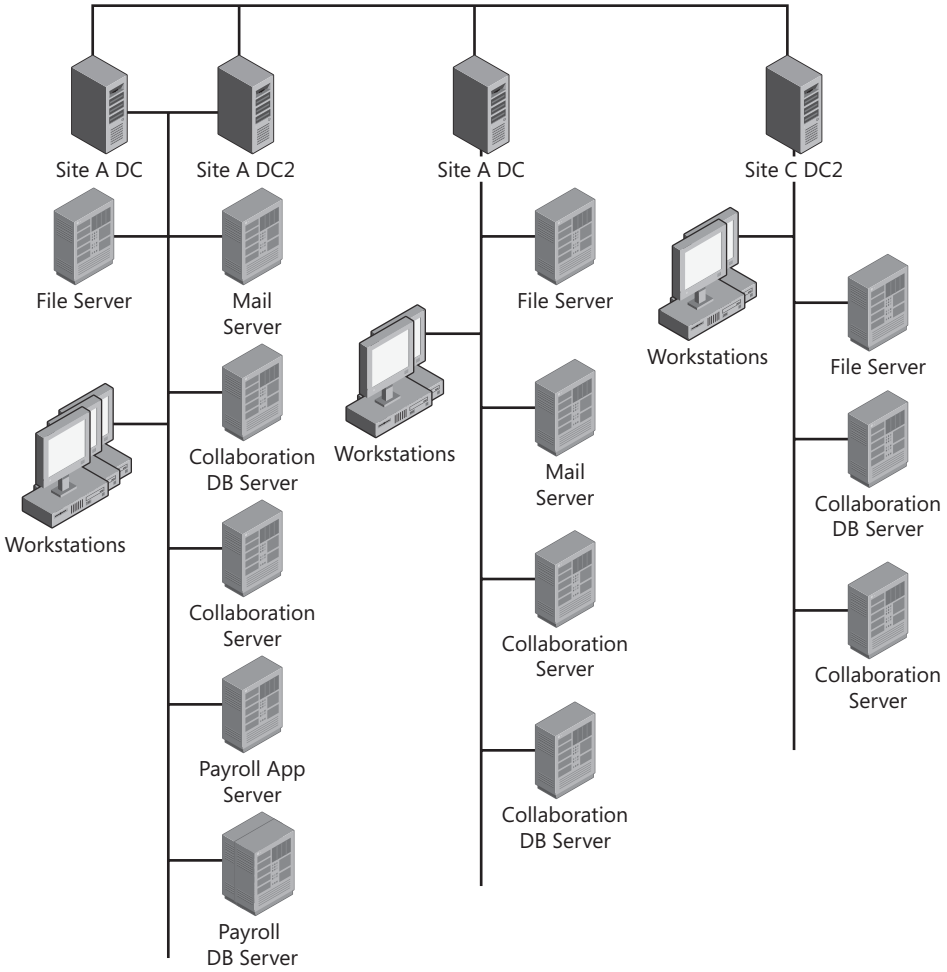

Figure 14-6 Locate or build a map of your network.

The next step is to start applying the classification scheme to the network map. As you have already noticed, Figure 14-6 is based on the physical design of the network, with each site shown separately, and with the same type of server in multiple sites. In Network Threat Modeling we are really not interested in the individual servers. Our objective is to understand the *types* of computers, not the individual computers. To that end, we take our classification scheme and overlay it on our network map. This will probably cause us to lose the distinction between sites. However, if the security needs of similar computer types are the same across sites, we have achieved exactly what we want to achieve. At this stage in the process we are trying to create a higher level of abstraction in our understanding of the network. This should result in a picture similar to Figure 14-7.

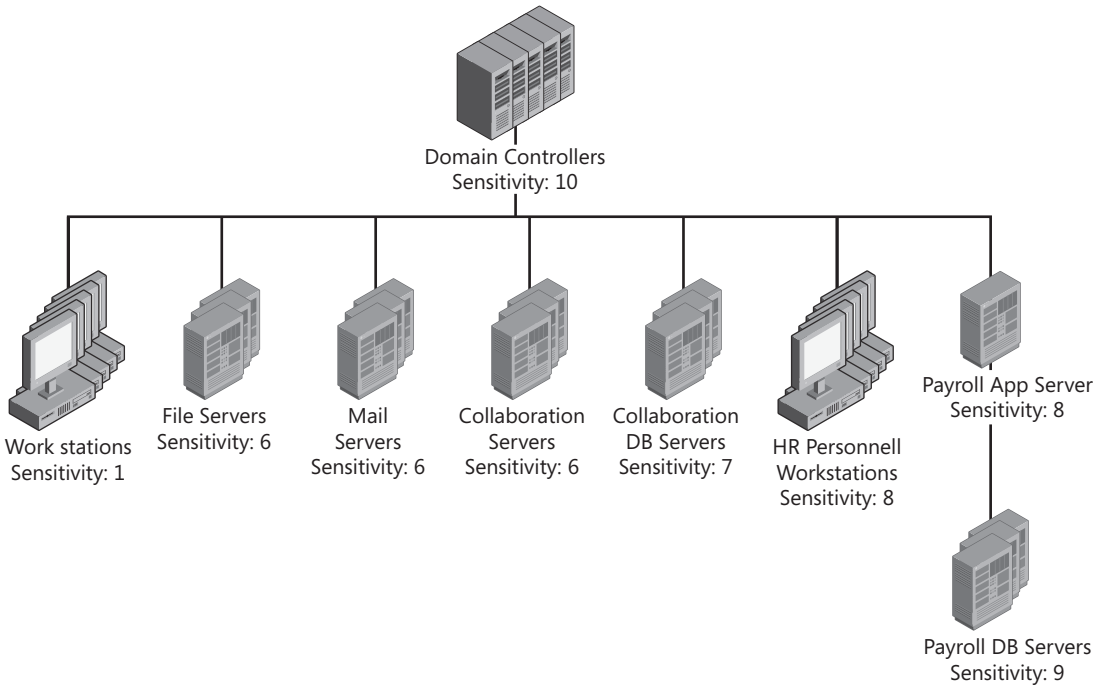

Figure 14-7 Start threat modeling by flattening the network and grouping computers into the classification scheme.

Figure 14-7 classifies computers into types based on our classification. Note that we have a new type of computer that did not appear before: the Human Resources (HR) Personnel Workstation. In this enterprise, we decided that because HR personnel have access to sensitive data on every employee, we needed to apply special security to their computers. Only some members of the client operations team that administers clients will have access to these computers. This prevents all client operations employees from having indirect access to personnel Personally Identifiable Information (PII).

When you have a classification scheme you have achieved a large portion of the objective of Network Threat Modeling. You should now be able to assign sensitivity labels to the various computer types. These labels are based on the types of data stored on that computer and the type of access to other computers you have if you successfully attack that computer. I have used numeric labels here, although you can use whatever makes sense.

DCs, obviously, are the most sensitive computers of all. Therefore, they have a sensitivity label of 10. By itself the number means nothing. It is just a way to relate one computer type to another. Workstations, because they are used by the largest proportion of users and at the highest risk, *should* be the least sensitive computers in the network. That does not mean that they are the least likely to be attacked. On the contrary, they are probably the *most* likely to be attacked. Therefore, they should be the least sensitive—in other words, the ones that give you access to the least amount of information in the network.

After you assign labels to all the computers, you should have a good idea of the patterns of operation in the network. This will drive your isolation strategy later. For now, we need to proceed to analyzing the communication patterns in the environment. To do that, we construct a picture similar to Figure 14-8.

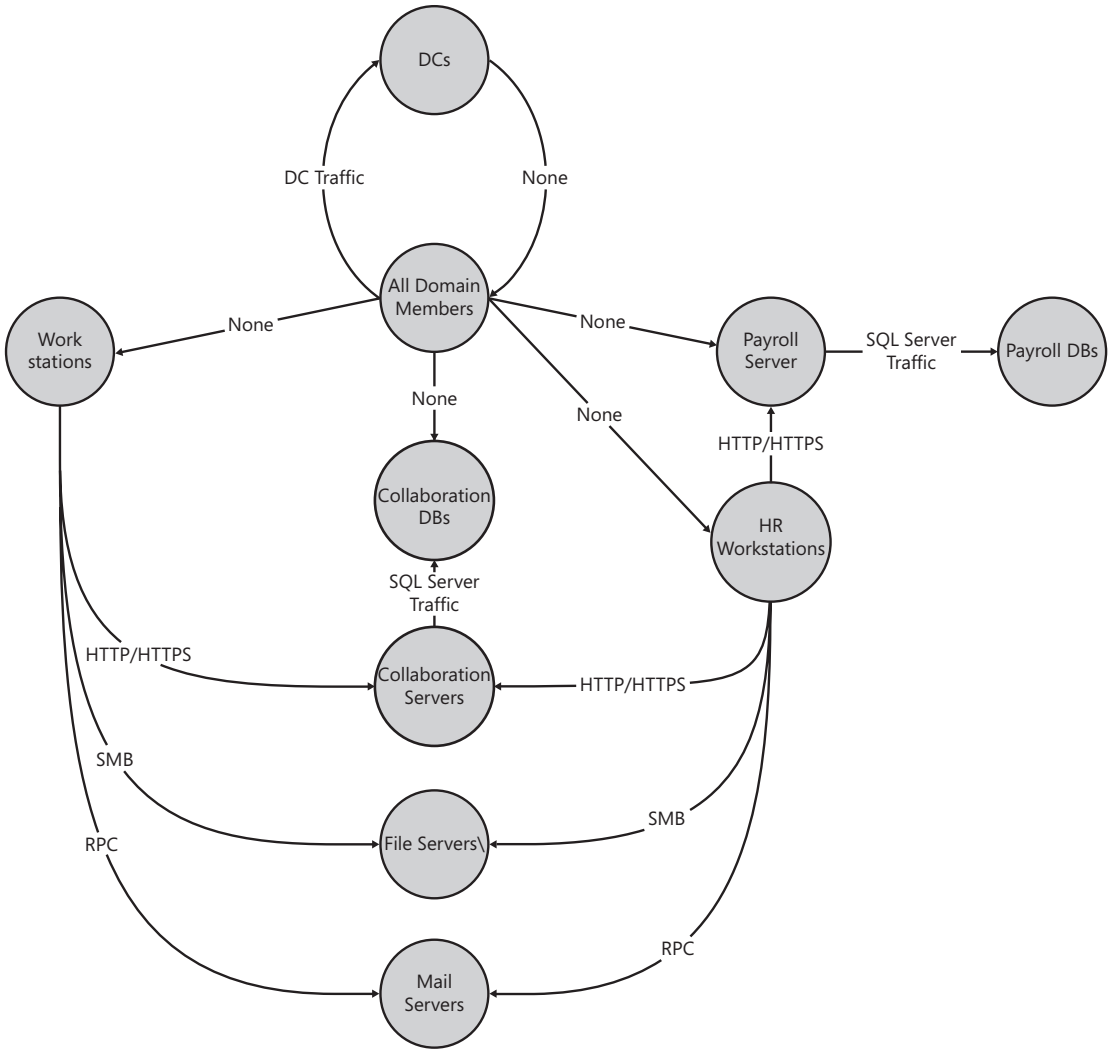

Figure 14-8 After you have grouped the systems, analyze their communication patterns.

Figure 14-8 is a basic Data Flow Diagram (DFD) of the network. The graph shown in Figure 14-7 does not easily lend itself to documenting communication patterns. However, a DFD is tailor-made for that purpose.

We start converting a network diagram to a DFD by simply turning the computer types into processes (the circles you see in Figure 14-8). Even databases are processes because the database server is actually what performs the processing on all database requests. Figure 14-8 also shows a little trick to make the picture far easier to read. Note the process named All Domain Members. It is marked as a duplicate entity with a slash through the corner. It represents all the non-DC computers in the domain. It serves as a very simple placeholder to clean up the diagram, letting us capture any communication pattern that is common to every computer in the entire domain. For example, all computers in the domain need to access the DCs. Instead of drawing separate lines from each computer type to the DCs, we draw just one from All Domain Members. In addition, rather than enumerating all the different types of traffic that domain members need to send to DCs, we use just one vector labeled DC Traffic. With this shortcut technique, what could easily have been 30 separate lines becomes just one.

> To learn more about the types of traffic used to access each type of server, see Knowledge Base article KB 832017. "Service overview and network port requirements for the Windows Server system" found *at http://support.microsoft.com/kb/832017*.

Note also that all the communication vectors are directed. The fact that domain members need to access DCs does not mean that DCs need to access domain members. In fact, they rarely do. If you are diligent about not using your DCs as workstations or management stations, you might not have to access any other computer from them.

Step 4: Analyze, Rinse, Repeat as Needed

While going through the Network Threat Modeling exercise, you might realize that your classification scheme is deficient. Youwill probably have computer types that are not used, and you will almost certainly realize as you go through the exercise that you are missing some types. If you are not, you probably have not adequately considered the security needs of your systems. Keep in mind that two things drive the classification scheme: First, you need to consider communication patterns. A computer that does not need to communicate in a particular way with another computer should not be permitted to do so. Second, computers that have different sensitivities should be managed differently to ensure that if one is compromised, the others do not fall.

One common mistake is failing to consider database servers separately from application servers. With properly written database middleware, which only calls exposed store procedures on the databases—and uses least privilege to do so—application servers are typically less sensitive than database servers. Unrestricted access to a database server means that you have complete access to all the data on it. Unrestricted access to an application server means that you should have access only to what the database will give you.

> You might have noticed that we have made an unstated assumption that there is no difference in the level of access to a particular computer, or rather, that this difference is not relevant to the Network Threat Modeling process. Windows does, in fact, have a reasonable level of isolation mechanisms to prevent someone with mere user access to a computer from taking complete control of that computer. However, Network Threat Modeling is complicated enough as it is. Mixing that in makes the model that much more complex. Instead, we are taking a worst-case scenario approach; we are basing our isolation techniques on what an attacker with complete control over a computer can do with that computer. For example, if an attacker has complete control over a SharePoint server, what access would that give her on the Office SharePoint Server 2007 database servers? The answer depends on how we manage the network (which users log on to the SharePoint servers) and on what traffic is allowed between the two.

If the classification scheme seems inadequate to the task of adequately capturing what your network looks like—or ought to look like—the solution is simple: either modify the classification scheme or change your assumptions. The classification scheme may just not be correct for your risk management strategy. In this case, you may change the classification scheme to better match the risk management strategy. Or, you might decide that although your risk management strategy is sound on paper, it is impossible or undesirable in practice. Many organizations have developed a risk management strategy

that looks great on a Microsoft Office PowerPoint slide in the boardroom, but is impossible to implement in the real world. This is your opportunity to verify how well your strategy really can be implemented. If you do not have a risk management strategy, you probably ought to take this opportunity to think up one.

Step 5: Design the Isolation Strategy

Once you have a network threat model that makes sense, you can start deriving the isolation strategy. The isolation strategy is largely based on the communication patterns identified in Figure 14-8. It should be as restrictive as possible, within reason. You can document the outcome in a table that outlines the server types and the communications patterns. The table includes the source and destination hosts and ports, the protocols, whether the traffic must be authenticated and/or encrypted, and whether the connection can also happen in reverse (mirrored). This is where you really need to get specific. Rather than simply saying "DC traffic," you need to enumerate the ports and protocols. An extraordinarily useful reference at this stage is Microsoft Knowledge Base Article 832017, "Service Overview and Network Port Requirements for the Windows Server System."

The end result of this step in the process should be a table that lists all necessary communications patterns in your network. Your table might look similar to Table 14-3, but will likely be far longer.

Table 14-3 Network Communications Patterns

Description	Source	Destination	Source Port	Destination Port	Protocol	Require Authentication	Require Encryption	Mirrored
DC SMB traffic	All Domain Members	All DCs	Any	445	TCP	No	No	No
DC RPC EP Mapper	All Domain Members	All DCs	Any	135	TCP	No	No	No
…								
Appserver DB Access	SharePoint Servers	SharePoint DB Servers	Any	1433	TCP	Yes	No	No
File Server Access	Workstations	File Servers	Any	(139), 445	TCP	Yes	No	No
HR Payroll Access	HR Workstations	Payroll Servers	Any	80	TCP	Yes	Yes	No
…								

As you can tell, the data captured in Table 14-3 can get quite extensive. However, if you have done the job of segmenting the network appropriately, the data should be mostly just tedious to gather. Once you have done so, you have almost completed the IPsec implementation of Server Isolation in your network. Notice that the headings in Table 14-3 actually capture the exact information you need for your IPsec rules. If you want to be really enterprising, you can enter Table 14-3 in a spreadsheet and then use a

macro to convert it into a series of IPsec commands to generate the required IPsec policy. You can configure IPsec on the command line using the **netsh advfirewall consec add rule** command. For more information on IPsec see the Microsoft IPsec site at *http://technet.microsoft.com/en-us/network/bb531150.aspx*.

Note that this kind of analysis will take some time. It is not unusual for a computer to have 50 or so ports open. The more standardized your OS images are, the easier it will be to track down the information you need. Furthermore, once you start doing this kind of analysis you will most definitely realize how valuable it is when the software vendor documents in a conspicuous manner what ports are used for what features.

Step 6: Derive Operational Strategy

The operational strategy is designed around how you are going to manage the various computers in your network. The strategy needs to capture the administrative needs of the computers as well as any services and other steps you take. For example, you probably want some backup strategy for your network. However, if you use a single, centralized backup system for all computers, you have probably defeated a large part of the isolation because you now have a backup server that has access to everything in the network, and it is potentially subject to attacks by every computer in the network. Therefore, you may want to analyze the risk involved in doing so. That analysis might lead you to conclude that the correct way to perform backups is to group computers by sensitivity and then handle backups uniformly within each sensitivity level. You might, for example, decide to use a single backup solution for all computers of sensitivity level 6.

You need to do the same analysis for administration. It would defeat the entire purpose of the exercise if you were to use a single, domain administrative account to access every computer in the domain. By doing that you expose the single administrative account to attacks on every computer. The appropriate decision is to use different administrator accounts for different purposes. You might decide that you have one account per sensitivity level. Or you might assign your sysadmins to different sensitivity levels. Doing so permits you to assign different sysadmins to different computers, rather than allowing all of them to administer every computer. You might even have separate administration stations for each sensitivity level. You have to decide how much pain you are willing to go through to manage your network. That decision will guide the rest of your decision making. With regard to security there is, as always, a tradeoff between how much security you wish to have, and how much inconvenience and work you are willing to put iup with to get it, all while taking into account the resultant functionality you want. The key point of this part of the process is to ensure that you implement the isolation in such a way that you do not expose computers of one sensitivity level to unnecessary attacks by computers at a different sensitivity level.

Step 7: Implement Restrictions

Finally, it is time to implement your strategy. By this stage in the process you should have a complete design for how you want to manage your network as well as for what communication patterns you want to permit within it. The implementation should be relatively straightforward at this point. However, you do want to ensure that several things get done.

Before we go further, if you are like most network managers, you get cold chills thinking about rolling out changes that could disrupt communication. After all, as the old saying goes, nobody ever calls the helpdesk to inform them that everything is working today (and if they do, you have a whole different set of problems to address).

Fortunately, there is a trick. Obviously, you want to eventually require authentication of all or most network connections. However, you may want to start out by *requesting* authentication instead. That way you can test the policies, monitor where IPsec negotiation fails, and adjust as necessary, all while maintaining full connectivity. This permits you to do a safer roll-out that is far less likely to result in events that have an adverse impact on your opportunities from promotion out of network management.

That being said, there are a number of other restrictions that you need to include in your plan.

Minimize Account Scope

First, reduce the scope of your accounts, particularly the highly privileged accounts. Everyone that accesses computers at different sensitivities should, at least if they have high-level permissions on those computers, have different accounts. For example, a highly trusted server administrator might need a domain administrative account for managing the DCs, a level 7 administrative account for managing servers at sensitivity level 7, and an information worker account for e-mail and surfing the Web. An HR employee might need one account for performing HR-related tasks and a different account for reading e-mail and working on presentations. Alternatively, you might decide that based on your risk management philosophy and the fact that both uses are at very low privilege levels, the same account might suffice. However, you should never permit an account that has administrative privileges at one level to access resources at a different level. Administrative accounts at any level must only be used to administer computers at that level.

Organizational Security Policy Changes

Much of the isolation must be done by organizational security policies, not necessarily technical policies. You simply cannot technically enforce many of the isolation decisions. For instance, your domain administrators are omnipotent within the scope of your network. You cannot restrict them from seeing or doing anything within the network. However, you can set rules and guidelines for them to follow, and track those guidelines. You must also have penalties for violating those guidelines. An administrator who refuses to take necessary steps to keep your network protected should be turned into an ex-employee.

Separate Service Accounts

Service accounts are a common problem in Windows. It is a well-known fact that any administrator on any computer has access to the clear-text password of all services—and of all interactive users—on that computer. There is no standard log file where these nuggets are stored, but with commonly available hacking tools it is a simple matter to get them.

For that reason, managing service accounts is crucial. It is still quite common to see services running on many computers in a network under a domain admin account. This

exposes a domain admin account on every computer in the network. For this reason, the scope of service accounts must be limited. A logical way to do this is to only use service accounts within a sensitivity level. For example, as mentioned earlier, the backup service might run in one service account on computers at level 7 and a different account on computers at level 9.

Do You Want To Back Up Workstations?

Do you really want to back up workstations? Many organizations are struggling with that question these days. Users, obviously, are storing data on their workstations, but is that what you really want? Ideally, very little data that does not exist elsewhere in the network should be on workstations. Using techniques such as roaming profiles and folder redirection, the default storage locations for users can be moved to the network. With the Offline Files feature, these files are automatically backed up to the network, and also available offline for roaming users. By using a combination of these strategies you can ensure that the only data available only on workstations is that data which users create while roaming, and that data which they choose to store locally—in possible violation of standard operating procedures. Combine that with a solid imaging strategy using (for example), the Windows Deployment Services, and you might achieve a state where you do not need to back up data on workstations. You might not even need to troubleshoot them. If anything ever goes wrong with a workstation, you could troubleshoot it by using same approach you use for servers, following this simple process:

1. Restart the service, if applicable.
2. Restart the computer.
3. Reimage the computer.
4. Send the computer back to the manufacturer and deploy new hardware.

If you do not need to worry about data being stored on workstations, you can make them disposable.

Note that getting to this stage will take discipline, along with hardware that permits you to implement a strategy like this. However, this is a business decision you need to make. Do you want to greatly simplify your desktop operating procedures, buy hardware and software that lends itself to that, and buy some really big storage servers, or do you want to have complicated and costly desktop operating procedures and spend less up front?

Manage Privileges

You must not forget to manage your privileges properly when you are implementing your isolation strategy. Users with certain privileges can be just as powerful as administrators. For example, a user that has the privilege Impersonate A Client After Authentication is as powerful as any user that connects remotely to the computer. A user that has the Restore Files And Directories privilege can replace any file on the computer. Because this permits that user to control code that is executed by administrators, such a user implicitly has all the rights that administrators do. This is why it makes great sense to separate backup

operators from restore operators. They are two different tasks, at very different sensitivity levels. Privileges are discussed in Chapter 2, "Objects: The Stuff You Want."

Restrict Communications

It should be obvious after our discussion on Network Threat Modeling that we want to restrict communications. In this step we use IPsec and Windows Firewall to restrict inbound traffic to a computer. This will greatly reduce the risk to systems from other systems. Take, for example, a database server accessed by a middleware server. Say there is a SQL Injection flaw in the middleware that permits an attacker to run arbitrary code on the database server. Once the database server has been compromised, what access does the attacker have to the middleware server? If you have set up Windows Firewall on the middleware server to reject all unsolicited inbound traffic (or, rather, to only accept exactly what it must accept) the answer is "none." You can contain the attack right there. Use the table you created listing your communication patterns and design a set of IPsec policies based on it. Deploy these policies using Group Policy or any other means that makes sense in your environment. To learn more about Windows Firewall and IPsec and how to deploy the policies, see Chapter 5, "Windows Firewall(s)."

Restrict Access to Resources

Finally, use the detailed knowledge you have gained, and the isolation strategy you have designed up to this point, to build a data and resource access strategy that enforces the principle of least privilege. You should, at this point, have a fairly detailed user account strategy. You can take advantage of that to prevent access, as well as to enforce the isolation strategy. For example, before you embarked on this project, your HR personnel might have had access to the Exchange servers, all the file servers, the internal SharePoint servers, and the payroll applications—all using the one account you gave them. After you define the isolation strategy, you have the ability to restrict their access to payroll applications to when they are using their HR_Personnel accounts, and possibly even when they are working on a specific HR workstation.

Summary

Few steps that you can take today will have as great an impact on the security of your network and its data as a proper network segmentation and Server Isolation strategy. By going through the process defined in this chapter, you can create a network that does exactly what you want it to do, and nothing else. If you do it right, the network will still be flexible enough to support new applications, with a minimal amount of modifications.

Will this strategy result in additional considerations for your end users and administrators alike? Certainly it will. However, in the world we operate in today, that might be the only way to secure your network. The conventional approach of a completely flat network, where everyone has one account that has access to everything they need, and much more, is simply unsafe in virtually all environments today. How far away from that model you are able to move depends on your risk tolerance, and the security needs you have in your environment.

Additional Resources

- Johansson, J. M. *Protect Your Windows Network*. (Addison-Wesley, 2005).
- Knowledge Base article 832017, "Service overview and network port requirements for the Windows Server system," at *http://support.microsoft.com/kb/832017*.
- "The Immutable Laws of Security," at http://www.microsoft.com/technet/archive/community/columns/security/essays/10imlaws.mspx?mfr=true.