Chapter 4

A Weak Foundation Amplifies Risk

I will always remember the first house we bought. It was small and crowded with four children and a dog, but it was our palace. As with any new house, I had a huge list of projects that needed to be done, so like any guy, I picked the most fun and "manly" project, which was to build a deck off the back door. Yes, the vision of grilling, picnics, and kids playing was crystal clear in my mind.

Like any young engineer, I had a great plan. I got my list together, borrowed a pickup, and headed for the lumberyard: soon I was ready to attack the project. My first task was to dig the holes for the supports that would hold up the deck. I got the first hole to the prescribed depth with minimal problems. On the second, about a foot below the surface, I hit a piece of buried concrete. Not to be defeated, I modified my layout and moved the hole about a foot. No luck—it soon became obvious to me that the piece of concrete was roughly the size of Ohio. Damn….

Needless to say, the same scenario played out on the remaining holes. Since I had no dynamite handy, I ran the posts down to the buried obstacles and decided that the support would be adequate; after all, it would have been very difficult to fix the subterranean issues. And, talking to some of my new neighbors, I found out that my new house was built on a lot with a lot of construction debris as fill. Who knows what was down there.

All was idyllic until the following spring. I noticed the deck had a definite list to the right. Sure enough, one corner had dropped about two inches as a result of the winter freeze and thaw cycles. Not to be outdone by a small problem, I jacked it up and placed some shims to level it out. Although I didn't know it at the time, I was turning into a young Don Quixote, who instead of jousting at windmills, I was making a career out of keeping the deck level, a task that turned out to be just as futile and pointless.

Hey, I really didn't have a choice; it would have quickly gotten to a point that I could only grill flat stuff like burgers. If I tried hotdogs, they might roll off the grate and end up on the floor unless I pinned them down with toothpicks.

In the end, I really should have fixed the initial problem, which was a weak foundation. I ended up spending more time keeping the deck level than I would have spent cleaning up the legacy environment.

I'm sure all of us can relate to this lesson from my life. In technology or business, we get so busy doing new fun stuff that there is never time to fix the foundational issue. We justify it calling it everything from "legacy issues" to the one I really hate, which is "technical debt." Whatever we call it, the weak foundation and inefficiencies of the past add risk to every new project. So, we develop a new system and identify and manage the risks in the new system; however, once we place the new system on the weak foundation, any residual risk in the system is amplified due to the problems in the infrastructure.

What is "technical debt?" Missing patches and upgrades, old or non-supported applications (apps) (come on, admit that you have at least one old application that still runs on Windows XP), broken/inefficient processes, and the list can go on. There are only two ways to address it, let the bad stuff hang in until it is eventually replaced or obsolete, or fix it. The ostrich approach of burying your head in the sand and ignoring it is not a viable long-term risk management strategy.

Remember, sooner or later, your weenies will roll off the grill and it will happen at the worst possible time.

In your organization, do you have a full understanding of all the components of your foundation? You may be tempted to believe it is only built of computers and networks, but in actuality it is the full combination of people, processes, and technologies. They are all interrelated, and any one element needs the others to make the environment work robustly and smoothly. I have learned over the years that an effective foundation encompasses the three elements of People, Process, and Technology. Take away one, or have one substantially weaker than the others and you get a very unstable base. Just like a three-legged stool, if one leg is shorter or weaker than the others, there is an accident waiting to happen. Figure 4.1 looks at the three elements of People, Process, and Technology, and how they intersect with the four primary functions of any information security group. Those functions are to Prevent (bad things from happening), Detect (if they do), Respond (to limit the damage), and Recover (quickly to get the business going again).

Let's look at the components from Figure 4.1.

People: In order to adequately protect your environment, the CISO must know who is accessing the systems and why. We all have employees and customers, but many of us also must allow partners and third parties to access our systems. If you have been watching the news, you know that literally all of the major breaches have

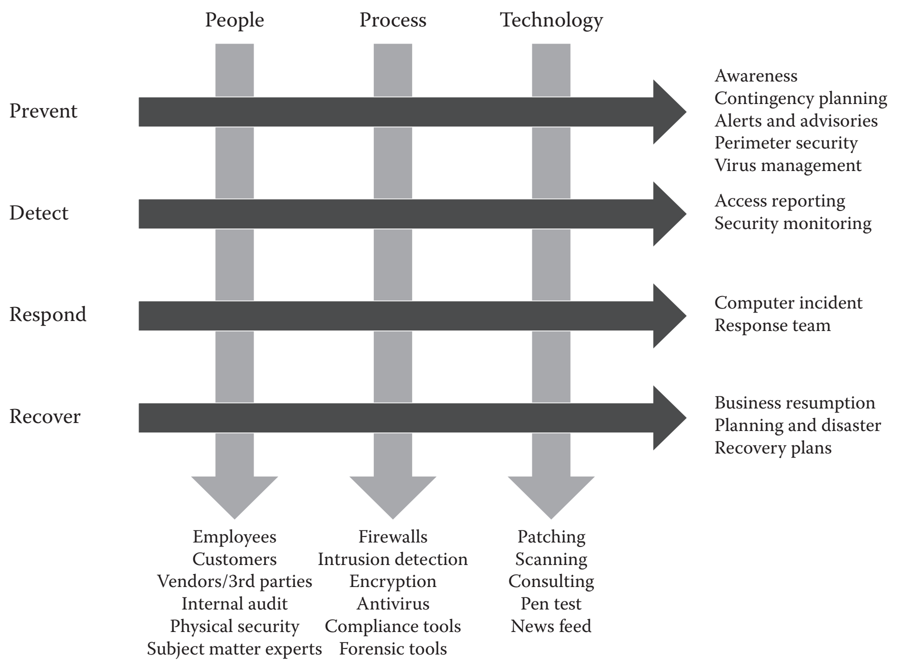

Figure 4.1 Elements versus functions.

involved a connection to a vendor or third-party partner. When you allow unfettered access from a remote location, you are extending your "risk universe" to their systems. On the positive side of the people equation, you have internal partners and peers such as internal audit, physical security, and subject matter experts. Build those relationships early and feed them through ongoing contact and team building. This will pay off in a big way when the bad stuff hits the fan and you need their help. Simply said, know your users, their functions, and your extended support resources.

Process: OK, you will hear this from me many times in this book, but a bad process is only marginally better than no process. Processes must be reliable and repeatable or they will cause more trouble than you can stand. A bad process will absolutely fail at the worst possible time. I listed a few above, patching, scanning, penetration (pen) testing.

Keep this image in mind as you read the rest of this book. Everything you do as a CISO needs to track back to these functions. This is why you were hired. Do this well and you will be successful.

So how is your foundation? Is it well maintained? Sadly, things today are not built like the Egyptian pyramids or Roman aqueducts. These have stood the test of time without the need for maintenance until recent years. The physical environment has changed much over the past 2000 years so these magnificent constructions are now in need of repairs. Unfortunately for us, the technological environment is changing at a much faster pace. So fast that most of us can't keep up.

The typical way manufacturers address technological change is through the distribution of patches, software updates, and full-scale new product releases. Considering the time and expense necessary to deploy a new product release, it is no wonder companies opt to remain on their legacy systems. Legacy tends to have different meanings to different audiences, so for our discussion purposes, legacy is any existing system in your environment.

I'm sure all of us are victims of the "Set It and Forget It" mentality. A new application goes into production and no one wants to touch it, fearing system downtime: unfortunately, this sometimes includes patching and vulnerability management. You may get away with this for the first year, but the threat clock never stops ticking. More than once in my career, I've found servers with obsolete operating systems that have literally thousands of vulnerabilities.

How do you handle the pushback from development and the business when you are facing servers with significant security issues? Start by understanding the business side of the equation.

1. What are the availability requirements for the application?
2. What is the revenue stream from the business function?
3. What information is captured or contained on the server? Are there regulatory requirements to protect the data (PCI, HIPAA, etc.)?

Now, look at the technical side and the application architecture.

4. Is there a test environment?
5. Is there a Disaster Recovery environment or hot backup?
6. Do we have a subject matter expert on the application?
7. How long would it take to rebuild the server if the worst happened?

Granted this is not a full risk assessment, but it gives you enough data to intelligently discuss the issue with management. You have an idea of the risk to the business if the application crashes as a result of patching and upgrade. From a security standpoint, you should already know the likelihood and impact of a vulnerability being exploited. Since you have asked compliance questions, you already know if there are any mandatory requirements to keep the server patched and in compliance, and the potential penalties and fines if the server is not patched.

Have a business discussion, not just a security exercise. Don't let the IT folks simply "accept the risk" of a business impact. Ensure the right people are involved in the decisions.

You can decide to stay with the current version of a system, keeping it current through patches and occasional software updates, but only for so long. Vendors don't go on providing patches forever. Eventually, they sunset their products and

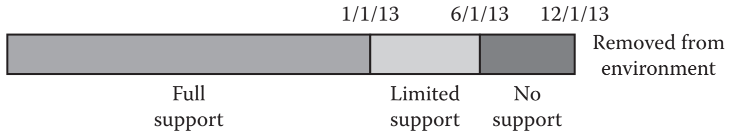

Figure 4.2 Support life cycle.

release new versions. Think of all the versions of Microsoft Internet Explorer you have used over the years or how many different Microsoft operating systems you have used. The pace at which new versions are released has accelerated over the years. Fortunately, prior versions continue to be supported, but I see that support window closing as well. Let's face it, vendors do not make a lot of money (if any) supporting older products. It ties up valuable resources.

Early in my career, I understood that just like everything, technology products have a limited time they are supported from the vendor. One of my hobbies is restoring cars; my latest is a 1957 Chevy. Chevrolet quit making parts for a 57 a long time ago. If I want parts, I have to buy them from a specialty manufacturer who builds the parts. At some point, it becomes too expensive and certainly not profitable for the vendor to continue supporting their earlier products. You need to be proactive about planning the life cycle of the critical elements in your environment. It doesn't have to be complex to start; the key is to start. Figure 4.2 shows an extremely simple method of conveying the life cycle to the rest of IT.

This lets all support functions know when they must have a component out of the environment. There will be a temptation to give exceptions and passes to the drop-dead date. However, as I've painfully learned over the years, you can dig an incredibly deep hole one shovel full at a time. Be careful and weigh any exception against the strategic direction of security and operations.

Patching: The Critical Link…

Let's face it—patching is not sexy. No one aspires to be the world's greatest patcher. Usually it is done off hours, eating into personal time. Invariably, it causes problems with applications and connectivity. Even my son, a tech manager for a college, hates the weekend patch window. Patching is much like the roll of the dice, every now and then you come up with snake eyes and all your hard work and preparations were for naught. Patching is typically "tested" by patching your development and test systems first (Figure 4.3). If they don't break, then the patch is moved into your production systems and all is well in the universe. Unfortunately, this is a very flawed strategy. As has been discovered time and time again, sometimes patches don't actually address the vulnerability they were designed for, or they introduce new vulnerabilities. You have to thoroughly test the patch against the exploit it was

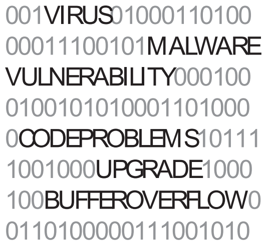

Figure 4.3 Patching.

designed for as well as other potential exploits. It would be like a car manufacturer installing air bags without ever testing them. You ride along assuming they are going to work just like you assume the patch is going to work. Fortunately, the car manufacturers do extensively test their air bag designs and installations (though recent air bag recalls would seem to indicate more testing is required). You need to do the same with your patches.

I've seen it all—or so I think—but people keep surprising me. Probably the most common mistake is installing a patch that requires a reboot, and forgetting to reboot the system. So, the organization cruises along, thinking it's protected when it is not. Without a post-patch scan, you really don't know if the patch took and did what it was supposed to.

Another common mistake comes from using an automated patching tool. The tech loads up the system and assumes the system did its job. Never assume successful completion. As the old saying goes—trust but verify.

Of course, older products tie up our resources as well. Legacy software is like a car; the older it gets, the more care and support it takes to keep it humming along. You can find support for your software, just like cars, from other vendors. As time goes on, this support tends to get expensive. Recently, I worked with a major database company to buy "extended support" for an older version. The cost was an additional 20% over my current maintenance payments coming to almost $300,000 for one year.

Eventually, though, even the vendors stop supporting their own software. I recall working for a company that had an ERP (enterprise resource planning) system so old even the vendor had no one on staff that had any experience with that version of the software. There are third parties that tend to fill the void left by vendors who have abandoned their earlier products. These third parties are likewise expensive and their actual level of expertise can vary widely. In many cases, you may find you are paying these third parties to learn your software. If your software is highly customized, support from third parties can be very limited.

Once vendor support ceases, third-party support is basically focused on keeping the software functional. There is no emphasis or concern on possible security flaws within the software. The bad guys love flaws that have been identified in newer software since there is a good chance these same flaws may exist in unsupported versions of the software that still exist in the field. They have a wide-open door to your environment with no one to close it. In most cases, you won't know that security flaws exist until it's too late, and even if you do know they exist, you have limited options to address them without vendor support.

Lesson learned? Your only true and correct course of action is to make sure your legacy software is kept up to date. This does not mean you need to be on the very latest version. However, you certainly want to be on a version that you know will be supported far enough in the future to allow you to upgrade to a newer version before support ends on your current version. This is all part of software life cycle planning that is rarely done or properly budgeted for.

It's no wonder that patching falls by the wayside. There is always something more fun or more interesting. This doesn't take away from the fact that it is one of the most critical security-related maintenance tasks that take place in any organization. That one missing patch may be just the vulnerability the hacker needs to sneak through all of your elaborately placed defenses. Hackers have plenty of targets and can afford to be very specialized. A hacker may need only focus on one or two vulnerabilities, biding their time till they run across an organization that has left them an open invitation. Once they are inside your organization, and they will get in, unpatched servers provide an easy path for them to move within your organization and escalate rights.

It's about More Than Patching

Patching is just one piece necessary for the maintenance of the foundation. Other vulnerabilities that must be corrected by means other than patching exist, much like a road that has potholes and a fallen tree branch across it. One problem is solved with patching, the other with a structural change. Applications, whether internally or externally developed, are susceptible to exploit. Poor coding techniques open as may holes as any missing patch. Poor system administration and build practices can open massive holes in your infrastructure. Vulnerabilities that require actions other than patching, such as a configuration change, tend to be far less obvious and can carry far more devastating results. All the patching in the world won't protect you if you have failed to remove a system default password somewhere in your environment. As with every technology, there are a million myths about the subject of patching. Here are four that I'd like to dispel right away.

Patching Myth One

You should always wait a month before applying a new patch.

I'll admit it. At one time, I actually thought this was a wise approach. However, that was a different time. A vendor patch addressed a vulnerability, but it didn't mean that there was an actual exploit in the wild. You had time to thoughtfully evaluate the impact of a patch. Today, it is still easy to think that the organization is better off waiting for a few weeks after vendors release a patch before deploying it internally. The idea is that if you wait a month and don't hear any screaming on security mailing lists, it will be safe to apply the patch.

Great concept, but the lack of complaints from others does not mean that you won't have problems. You need to test it yourself in your own environment. If you will have problems, waiting a month will only delay the amount of time that passes until you discover the issue. Let's also remember that in today's environment, a month of being vulnerable to a serious exploit is an eternity. Today, we must patch quickly or put in compensating controls such as Web Application Firewalls to address the risk.

Patching Myth Two

If a patch doesn't break your test systems, it will not break all of your systems.

Come on: this is just testing common sense. Make sure that your test plan is representative of the environment. The plan should take into account the risk that a bad patch could bring to the organization. High risk, in-depth testing. Low risk, limited testing.

Patching Myth Three

Push the patches and forget. They install always cleanly.

So, you deploy a set of patches to your network. A week later, the compliance folks run a scan and you're told you have a ton of missing patches. Like all good computer folks, your first approach is to try and prove they are wrong. But, finally you have to give in and admit that somehow the patches didn't take. What happened? The patch required a reboot to successfully install and half the servers didn't reboot successfully. That scenario is just one of a hundred things that can go wrong with a patch deployment. The last step of a patch process is to verify; always remember that and you will avoid a ton of pain in your future.

Patching Myth Four

One size of patch management fits all.

Look, there are some basic principles that should be kept in mind when developing a patch management plan. For instance, you could take the recommendations of one of your vendors, but you have systems from multiple vendors. You could pick a package and implement the "vanilla" process in an attempt to utilize the vendor's expertise, but their processes are not representative of yours. Ultimately, you will need to tailor industry standard methods to your organization. Once you've implemented your plan, it should not be static. Regular reviews are critical to ensure that it continues to meet your organization's needs. As the organization changes, your plan must change.

Scanning Required!

Here are a few danger signs for your environment:

1. The network guys tell you that they can't produce a network diagram because the environment is too complex. Translation: I'm not really sure what is where anymore, but hell, it still works, what's your problem?
2. There are servers and apps that are too critical to be patched or scanned. The truth? If you have apps or servers that can be crashed by a passive scan, scanning is the least of your problems. These are usually business critical servers, which are so fragile that they can't be patched or scanned. Does this sound like a Business Continuity or Disaster Recovery issue? Check for backups, you may find that there is no backup of the server.
3. The vendor didn't tell me there were new patches. Who owns the application internally? Any service necessary for the business needs an owner. The maintenance of that application needs to be part of their review and raise process.
4. Finally, and always a goody: My application is patched but the underlying components are not my problem. "I just handle the code, if JBOSS or Adobe (which the application requires) have vulnerabilities, it's not my problem." IT management must understand that all vulnerabilities are their problem; no one gets to pick and choose. Ensure that someone is RESPONSIBLE.
5. Scanning for vulnerabilities besides just missing patches is crucial. Having a methodology in place to deal with the results is equally important. Devoting resources to the problem goes without saying. The only way these resources will succeed is if they are given adequate support from the top, and their efforts are recognized for the true importance they convey. If you don't reward and recognize those who are performing your vulnerability scanning and patching, it will never get the attention it needs, nor will it attract the quality talent necessary to ensure it is done correctly.

A sound vulnerability scanning program is critical to the ongoing security of your systems. Patches alone can't resolve vulnerabilities caused by the interaction of multiple systems and applications. A combination of automated and manual scanning, accompanied by a regular penetration test, is mandatory for any diligent security program. As with any program, let's look at common misconceptions and myths about vulnerability and testing programs I've encountered over my career.

Misconception One

Vulnerability scanning can identify all vulnerabilities in an organization's environment. If you do a good job at that, you can save money by not doing penetration tests.

Been there, done that—I thought that if I could do weekly vulnerability scans of the network and systems, that was better than a penetration test since I was finding issues in real time. Besides, have you seen what the Penetration Test firms get? Wow, I can certainly use that money elsewhere.

Here was the hole in my logic. The scanners are preloaded with "signatures" to detect known vulnerabilities. The vulnerabilities are simply "doors" into the company systems. No scanners are perfect, so there will always be "doors" left open. A penetration test goes beyond finding the open doors and looks at what actions an attacker could take by exploiting a given weakness. For instance, a vulnerability scanner may detect a system using a default password. A penetration tester could use that default password to see how far into the network they could get with those rights. That is a true measure of a system's weaknesses.

Misconception Two

Professional penetration testers use really expensive tools, so it's really not a good test of what a hacker could do.

True, professionals may use some custom tools, but the majority of the tools they use are freely available to the hacker community. In fact, a large number have actually been generated by the hacking community. Many of these tools are "Wizard Based," making it easy for relatively unsophisticated bad guys to compromise networks. The good things about penetration testers who use those tools is that they can simulate an attack from a malicious hacker.

Misconception Three

A penetration test was a success if the attack team couldn't get into your network.

Contrary to what you may think, if the testers couldn't get in, you probably should hire another firm. Every network has vulnerabilities that can be exploited. All systems have users that open holes. I've used some great firms over the years. At a high level, I do three tests. The first is a look from the public Internet. The second is where I give the tester access to a network jack with no user id or password. The third is where I give them the phone number of the help desk to see if they can "social engineer" their way into my system. The longest I've survived their methods is three days. That is how long it took them to get user rights (sometimes system administrator) on the network. Understanding how they got in is always invaluable. It allows you to make real changes and upgrades to your system defenses.

Misconception Four

A system compromise is only applicable to the system that was compromised.

Any penetration tester worth his salt will use one compromised system as a launch point against other systems. Once an attacker has established a foothold, they can watch for and capture user credentials to use against additional systems.

Misconception Five

Focus your penetration testing only on production networks containing sensitive data.

Many CISOs focus on production networks containing sensitive data, excluding other networks containing nonsensitive data, such as development and test environments. These are the networks most vulnerable to malware infection due to the open and fluid nature of the work performed there. The bad guys also know this and in many cases look for holes there first. It's incredibly important that they also be scanned and penetration tested as they could be launch points for production system attacks.

Take an inventory of your applications. How many are unsupported right now by the vendor? How many will be unsupported within a year? Chances are the numbers are going to be larger than you like. More troubling will be the number of applications and systems you find where no one is the owner and no one knows what they do. And don't just focus your inventory on the big name software. There is a lot of software that runs behind the scenes that is just as critical to keep current, such as Tomcat, Apache, Adobe, Java, your databases, and so on. Any in-house–developed software is most likely composed of a number of commercial third-party products. It is typically these behind the scenes–type software that makes it so difficult to keep up to date because of all the interdependencies. Upgrading one thing might break several other things.

The easiest way to explain this is to reference the seven-layer open system interconnection (OSI) model (Figure 4.4). Every application that runs in your

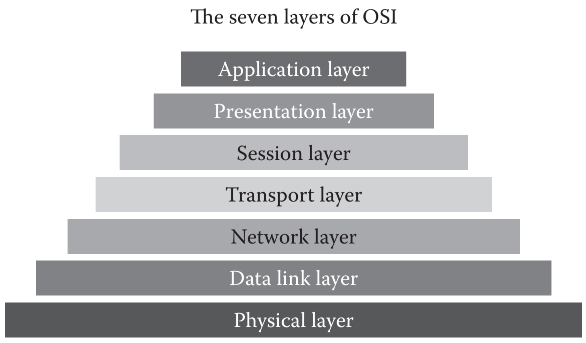

Figure 4.4 OSI layers.

environment is built on these seven layers, each having its own risks and vulnerabilities. True vulnerability management must look at the People, Process, and Technology vulnerabilities of all layers. The Application Layer may have vulnerabilities as the result of a coding error, and the Physical Layer may have vulnerabilities related to weak change control procedures that would allow a technician to pull or disconnect the wrong wire.

One necessary tool for tracking all these elements is an asset management system. If you don't have an asset management system, you seriously need to consider getting one. If you are a small shop and can't afford a big system, use a spreadsheet or Access database. The key point is to document, document, and document! The more you know about your environment, the better. Don't let the magnitude of a subject stop you from doing anything. I've been in many discussions that go like this:

"We all agree that we need an asset management system. If we don't know what we have, how do we know what to patch and maintain?" Everyone around the table shakes their head and agrees, but that is where the sanity gets very scarce. The production manager says, "You know these systems cost hundreds of thousands of dollars and take a couple of people to run. Unless I can collect every conceivable bit of information about the environment, we should not start." Next comes the development manager who says, "I need the tool to map all the applications running in the environment and how they relate to each other." Last country heard from is the Finance group who says "There is no way we can afford a few hundred thousand dollars for this project, let's just put it off until we can afford it." I sense many of you have been in the same meeting. We went from a simple spreadsheet to list what our equipment (better than what we had which was nothing) to a project we couldn't afford. Then because we couldn't afford the top shelf solution, we decided to do nothing. You don't have to boil the ocean. As CISO, you will sometimes make more progress by taking small steps that are lined up with a long-term objective. Here are some ideas to get started.

Environment Control

- Managing equipment life cycle and depreciation status
- Enforcing standards
- Controlling purchases
- Optimizing utilization
- Managing vendors

Tracking IT Assets

- Assigning value
- Tracking cost of ownership

Risk Management

- Tracking software licensing compliance
- Assigning and tracking ownership
- Tracking security risks

At a minimum, you should have a run book for each application in your environment. This run book should include what applications are running on what servers, when they were installed, what they do, what they interface with (especially databases), who is the subject matter expert for each, who is the business owner for each, what software language they are written in, are they public or internal facing, do they contain sensitive data, and the location of additional reference documents for each application. The more details, the better. One thing you might discover is applications installed by users and business units that have bypassed IT's normal change management processes. A configuration management tool can greatly help prevent these types of vulnerabilities entering your environment.

Lesson learned? Be sure and identify the owner for a system. Lack of assigned accountability and responsibility for a system results in legacy systems remaining in place long past their useful life. There is no one driving change for the system or accepting responsibility for the system should things go south. IT is usually left holding the bag on these systems. Yet, IT cannot speak for the business. Any attempts at changing the system are usually met with outcries from the users. In some cases, there are a very small number of users that should not realistically warrant keeping the system alive. The simplest solution for IT then becomes one of accepting the status quo even though this is not the correct decision. Why fight City Hall? Every system needs to have a business owner as well as an IT owner for the life of the system. If no one in the business feels strong enough to own a particular system, then it should be shut down.

Once you have a good inventory of your systems and run books created, you can then prioritize the systems that need attention from a security perspective based on potential risk. While every asset deserves attention from a security perspective, you simply do not have the time, resources, or money to address every server and application to the same degree. Obviously, if the system processes or stores sensitive data or is required to meet certain governmental requirements, it should be high on your list. If the loss or compromise of a system would cause business critical functions to fail, having a significant financial impact to the company, then these should sit high on your list.

How do I decide what needs to be fixed first? Each asset should be looked at from at least two perspectives when considering risk. The first is what is the likelihood that the asset can be breached? If it is behind a firewall and in a locked data center, its exposure is much less than a server sitting in the DMZ or one of your mobile assets. What is its current patch status? How many people have access to the asset? I'd like you to think in terms of a 1–10 score where 10 is the greatest likelihood that the asset could be breached. Keep that number in mind; we will use it in a couple of minutes.

The next perspective: What are the consequences to the company should this asset be breached? In other words, what is the impact to the company? Lastly, you need to consider what the probability of occurrence of a system breach is. This is similar to the risk exposure, though it focuses more on external factors, such as whether your line of business is a prime target for hackers and whether the asset in question is one a hacker would be interested in. Is it an operating system that is frequently targeted by hackers? Do you have a high degree of monitoring in place? Each of these factors can be classified as high, medium, and low or on a scale of 1 to 10. This is very subjective, but it can give you a relative idea of comparative risks.

Depending on your line of business, if you have systems that people's lives depend on, then they should go to the very top of the list. By lives depend on, I mean systems that could affect one's physical health and not some application; e-mail typically seems to be considered in this category, that people can't "live" without. I once worked at a company where they occasionally turned off the instant messaging system. The CEO strongly felt that people needed to actually talk to people, if not face to face then at least by voice. He also wanted to demonstrate that some systems aren't as critical as we might think they are.

Clearly, anything accessible from outside the company (web based or otherwise) is a higher risk to the company and needs to be given higher consideration. Age of the hardware and the applications also impacts risk. Putting old software on a new piece of hardware or vice versa does little to reduce overall risk. Just because an old system has run reliably for years without compromise is no guarantee that it will continue to do so. Older systems have most likely undergone a number of changes over the years. How thoroughly tested and documented these changes have been is a matter of concern. What were once considered minor configuration changes may have opened up large security holes that were not recognized at the time of the changes. Older systems tend to grow a multitude of interfaces over time. These tend to be poorly documented. If any of its interfaces are not secure, then it doesn't matter how secure it is itself.

If you are lucky enough to have access to the test documentation for the systems, you can learn a lot as to how secure they might be. The number and types of tests conducted will have a bearing on how secure a system is. A review will reveal additional tests that should be conducted to address new vulnerabilities that have been discovered since the system was first tested. Don't limit your review to just the test documentation. Look at all the system documentation that is available to see how security was addressed. The design may have been architected under the assumption that other things were going to be deployed in the environment that might never have occurred or have changed significantly over time.

In reviewing your systems, you need to think like a bad guy. What systems and data would they be most interested in getting access to? What ways are available to them to try and get to these systems and data? Be very creative in your thinking. The more unconventional the thinking, the more likely you are to discover potential vulnerabilities in your environment. Take advantage of your staff's extensive knowledge. Make it into a contest to see who can come up with the most ways to penetrate your environment or a particular system. If you have systems that have been breached in the past, these need special consideration. What has been done to fortify them? Are these measures still viable in today's threat environment? Is it possible that the vulnerability that permitted these systems to be breached may exist on other assets? Are these systems still needed?

Once you have identified your risks and prioritized them, you then need to concentrate on your mitigation strategies. When considering your options, more than just security concerns need to be evaluated. Business impact, resource limitations (human and system), system interfaces, technology changes, system age, functionality needs, documentation availability, subject matter expert availability, user interface, and maintenance needs come into play for the system in question and its mitigation alternatives. Like most problems, you will have to decide between several alternative solutions.

Using the previous 1–10 rating, you have a high-level view of the security risks posed by each system. By using a grid similar to Figure 4.5, you can map the scores of each vulnerability and generate an easily understood road map of what issues should be first on the "fix list." For instance, the issue identified by the bubble numbered one has a Likelihood of being exploited as a 10 and an impact of the company, if it happens, of 10. This could be a web server exposed to the Internet with an easily exploited vulnerability (10) that runs the company's e-commerce site. If it goes down, the company revenue is affected immediately (10). Plotting those

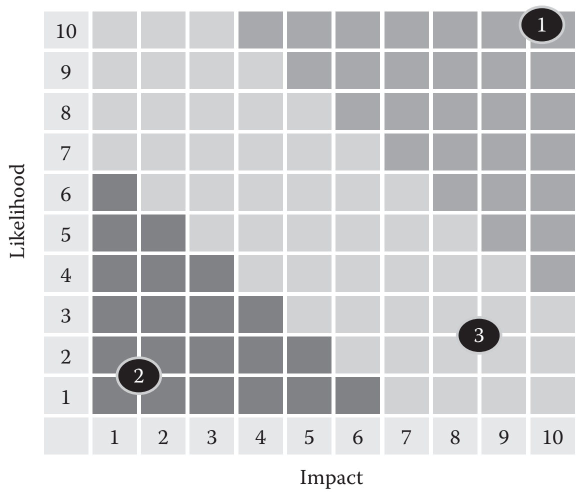

Figure 4.5 Risk matrix.

two points puts the issue in the upper right corner of the graph, meaning fix immediately. Issue two has a Likelihood of 1 and an Impact of 1, meaning that the issue is not a high priority and can be fixed at a later date. Issue three has a Likelihood of 2 and an Impact of 8. This means that while the likelihood is low, an exploit would result in a high-impact issue to the business. When you plot the issue, you can see that the 3 comes up as a Medium issue, meaning it must be fixed after the High issues are resolved.

Your firm needs to define what the colors mean. Personally, my starting point is that High issues must be resolved in two weeks, Medium in two months, and Low in six months. Don't finalize your definitions in a vacuum; form a team of business and IT leaders to develop the change windows and MONITOR the health of the program. If the process is not monitored and reported on, it will fade into oblivion, leaving the business with a false sense of security that risks are being addressed. Trust me, that is worse than never having started.

One option you always have is to do nothing. Doing nothing is usually more a matter of delaying a decision rather than a legitimate option. Opting to do nothing is usually chosen when a particular system is due to be upgraded soon or replaced. If there are real security concerns with a system, then doing nothing is never a smart strategy. We cannot ignore the inevitable. Even as a delaying tactic, doing nothing is not smart. The system is still exposed. The bad guys certainly won't be doing nothing while you procrastinate. As we all know, projects get delayed or even cancelled. If you have a security issue, it needs to be addressed now in some shape, manner, or form. Doing nothing only makes sense if the risk is small. While some may argue that the cost to mitigate may be too excessive to justify the particular option, the cost of any breach will greatly outweigh any potential solution you might ever come up with. At the very least, you need to find a way to reduce the risk if you can't eliminate it. I must caveat this in that you will never be able to completely eliminate all risk. It is an exponential curve where you eventually reach greatly diminishing returns on additional investments in risk mitigation.

Lesson learned? Not all mitigation strategies have to be costly or elaborate. Creating new policies and procedures may be a very simple and effective way to reduce the risk. Don't get caught in the techie trap where every risk needs an expensive hardware solution. Just applying additional monitoring will bring you down a bit on the risk curve. Training can always help reduce risk whether it is for your immediate staff or others. Making a few adjustments to your existing systems such as implementing a new group policy or firewall change can help reduce the risk without considerable expense.

Earlier in my career, as patching was becoming an issue, we identified a need to upgrade the web server software for an external site to address a security issue. As I talked to the developers, I found that there were a multitude of additional changes they wanted to do as long as "We were touching the code anyway." A few weeks later, I got a call from an irate product manager wanting to know why my patching requirement was going to cost about $200,000. Much to my surprise, I found out that a whole product rewrite had been included in my patching requirement.

Lesson learned? Many times, you may need to patch or fix a vulnerability in a system, especially in-house–developed systems. Be careful you are not drawn down a rabbit hole by developers and the business wishing to implement all kinds of new functionality along with the security fix. The primary motivation should be to fix the existing security flaw. Time is of the essence. Taking on new functionality not only slows down the process but may inadvertently introduce new security flaws in the rush to address the existing flaw. Developers will be more interested in working on the new functionality than the security fix. Security fixes are boring. Testing will likewise be skewed more toward the new functionality rather than the security fix. Better to just concentrate on the security fix and leave any new functionality enhancements to be addressed in a separate project.

Lesson learned? Avoid the perception of chaos. Sometimes called "Whack a Mole" management, this is where you have so many issues to be addressed that no one knows what to do first. This is compounded if you are also confused. Take responsibility, stand up a plan, and work it. Trust me, I've been in this situation many times during my career, sometimes my fault, sometimes the actions of others. Understand that you may discover new vulnerabilities or the same vulnerability scattered throughout your code. Vulnerabilities can't be ignored. Find them, list them, organize them, and kill them. Be a leader.

New vulnerabilities will always come up and must be prioritized against the degree of risk they present. To avoid scope creep and cost overruns, new vulnerabilities should be addressed and analyzed separately. If you try to do too many upgrades or patches at one time, it will only increase the chances something will be broken in the software. As you find new vulnerabilities, you need to assess where else in your environment this same code may be employed since developers often reuse code. A good software code library will help immensely in identifying where reused code is implemented.

Lesson learned? Don't alienate the Development folks. A challenge that security faces is how to properly communicate security issues to developers. Historically, the security engineers come out of network engineering, so there is a built-in "language barrier" from the start. These communication issues are not just pertinent to security, but they are definitely magnified due to the unique lingo of security professionals. Developers can more easily relate to business functions than security concepts. Your security staff needs to be adept at taking complex and obscure concepts and translating them into actionable requirements for the developers. This is not a skill that most security personnel readily possess. The employment of Use Cases can help developers more easily understand complex security requirements provided your security staff know how to develop Use Cases. Also, be aware that a developer that has worked on a project for months is proud of their achievement. Avoid calling their baby "ugly!" Tact and diplomacy are as important to a security professional as technical prowess. Build a partnership with developers, offer training, ask for input, and offer development basics training to the security staff.

Upgrading an existing system to address vulnerabilities takes lots of planning and time. Sometimes you are confronted with a decision to upgrade an existing product or move to a competitor product. My experience has been that if your existing product meets your needs and you have decent support from the vendor, then why switch? Switching is always harder and usually more costly than just upgrading. Switching should never be predicated solely on security concerns. The patching tool we were using would not support MS Server 12. Rather than upgrade the tool, a decision was made to purchase a different patching tool. We now have the new tool patching MS Server 12 assets and the legacy tool patching the rest. Getting off the old product has proven far more difficult than anticipated. Now that all assets are being patched, there is less urgency to consolidate on the one patching tool. Sadly, this scenario tends to repeat itself far too often where new systems are brought on line to replace old ones, yet in the end, both systems remain in production. What you don't want to do is throw the baby out with the bath water. While your system may be vulnerable, it may only be a small component of the system such as the database or user interface. Much like a car, you sometimes have to replace a part now and then to keep it in top running condition. Only after a fair amount of time and use does it make sense to replace the whole car. Of course, if it is just the radio that is working great, you don't necessarily want to replace everything around the radio. Designing and incorporating modular systems in your environment make this such a viable approach. Having good documentation, especially well-documented interfaces, is the key to this approach.

Lesson learned? Understand the financial impact of system support. There are a multitude of reasons why legacy systems are allowed to remain in production despite their direct impact on the foundation of your enterprise. Cost is always brought up as a key driver to stay with legacy systems. No one wants to foot the bill for an upgrade. Costs include both time and money. Plus, like the person who buys a car and drives it into the ground, there is a tendency for businesses to try and get as much as possible out of their initial investment.

The true lifetime costs of systems are rarely recognized when they are implemented. No real plans are ever presented up front that identify the end of life of the system and delineate how it will be replaced. What we have are systems that tend to never die.

In the 90s, I worked for a company that had an inventory application written in an obsolete language; eventually, the programmer retired. Every year, like clockwork, the application needed to be tweaked at inventory time, and the only person who could do it was the retired programmer. He knew he was the only resource and charged exorbitant rates to come in and do the work. Worse yet, as he got older, the company began to wonder whether he would be around for the next inventory run. After a few years, they had no choice but to rewrite the application, which should have been done years earlier. Sure, it only ran twice per year, but it calculated the inventory carrying charges passed back to the operating unit by corporate. These charges could amount to hundreds of thousands of dollars; an error was a significant financial risk to the business.

Delaying upgrades as a means to save money is not a good strategy. Older systems inherently require more personal attention to keep them running. They are more likely to result in downtime and tend to run slower over time. Users are deprived of enhanced functionality available in newer versions that could result in quicker processing and fewer errors. When vendors stop supporting their products, you need to hire specialized consultants when internal resources cannot solve a system problem. Hardware parts become harder to find and more costly. These sorts of costs tend to be overlooked.

Key Questions to Ask

- Are annual support and maintenance costs less than the system replacement costs for both hardware and software?
- If the current system support person were hit by a bus tomorrow, could you still do business?
- Can you quickly find people with the skills to maintain the hardware and/or software applications?
- Does the original manufacturer still have replacement parts for your hardware or are you using hard-to-find third-party or used resources for parts?
- Is the original vendor still in business and actively offering support for your version of the system?

If you answered "no" to some of these questions, then you probably should be on the lookout for upgrading or replacing your legacy systems before they cause serious harm to your business. If you answered "yes," to these questions, then your systems are probably fine for now, but stay alert for obsolescence by staying current on industry trends for your systems.

I worked for a company that had a large ERP system that they stopped paying support for over 10 years earlier, yet continued to use the product on which the entire company depended. The system had to use an unsupported database as well as run on an unsupported operating system to remain functional. It finally got to the point where system failures and extremely poor response times forced the business to make a decision to upgrade it. Unfortunately, in order to upgrade, the vendor required them to pay for all of the previous years of support plus the upgrade costs. In an attempt to save money, they put the users through a lot of pain and suffering, risked potential catastrophic system failure, and exposed the business to security breaches all for naught.

People know the current system and are comfortable with it. Things currently work OK. The general sentiment is if it isn't broke why fix it? People don't like change. Upgrading a system tends to have less resistance than replacing it with something entirely different. Employees who have worked on a particular system for years do not relish having to learn a whole new system. The time and costs involved to retrain people on the new or upgraded system are certainly a factor in staying with the status quo.

Lesson learned? Challenge the assumptions regarding legacy systems. My experience is that most of the assumptions have no basis in fact. If an assumption hangs around long enough, it is assumed to be fact, and no one challenges it. For example, the U-2 spy plane piloted by Francis Gary Powers was assumed to be untouchable by Soviet anti-aircraft missiles. The strategists were confident that no missiles could reach, let alone shoot down the U-2. Yet, on May 1, 1960, the unthinkable happened; the U-2 was shot out of the sky. Ultimately, this legacy system was breached. The military was forced to recognize the risks the legacy system imposed and had to upgrade and replace its hardware. The U-2 was ultimately replaced by the Lockheed SR-71 Blackbird in 1964. Had the Air Force been able to upgrade sooner, the U-2 incident would have never happened. No SR-71 was ever shot down.

While the rest of the world marches ahead with newer technology, you may find yourself in a situation where your legacy systems cannot interface with the new stuff. Your only option may be to build very expensive customized interfaces. These will prove to be difficult to maintain. Bringing on new systems will take longer.

Keeping legacy systems around may also hurt your employees. Their skills become dated. Your good resources will leave rather than hang around to work on dinosaurs. In-house knowledge of legacy systems will quickly diminish, further confounding support of these systems. Attracting new talent will also be difficult since no one wants to work on unsupported and outdated systems. If you find yourself in a situation where you have to frequently reboot your servers, this is a very good indication that it is time to upgrade or replace. Either your in-house resources do not have the requisite skills and knowledge to find the root cause of the issues with the system or the system has reached its tipping point.

Something else to consider. Your legacy system may have gone into production three years ago, but when did coding actually begin? Now consider that developers like to reuse code and take advantage of public domain code. How old is this predeveloped code? Your software may be a lot older than you think. It's like a 20-year-old getting a heart transplant from a 60-year-old. Things may work, but what imperfections did we inherit?

Your applications are only as secure as the hardware they reside on, the operating system they utilize, and the network they interact over. Each piece has to be secure. People tend to forget how important it is to keep your hardware's firmware up to date. Most patching tools cannot address hardware firmware. Different skill sets tend to be needed to address hardware vulnerabilities.

You need to upgrade or replace. That is the only way to firm up your foundation. New software, while certainly not invulnerable, has the advantage of learning from the mistakes of the past. Like the unsinkable *Titanic* that incorporated all of the best aspects of ship design, your newly installed software is very secure against hacking until it runs into its cyber equivalent of an iceberg. And like the *Titanic*, there are a number of factors that must come into play before your software is "sunk." The Internet is a very big ocean, and your company a relatively small ship on that ocean. A technological form of global warming is setting adrift a lot more icebergs across that Internet. You may be able to travel great distances and for a long time before you come across a cyber iceberg. You have the advantage of many other corporate ships at sea with you who can help you avoid these cyber icebergs. There will be far fewer "USS Legacy" ships at sea as time goes on.

Should you decide to replace an existing legacy system, then you need to reevaluate your current security architecture. It makes no sense to put a new system in an environment with security designed around the previous system. This is an excellent opportunity to make further improvements to your security posture. In all likelihood, the decision to replace the system was not made based on security concerns but business needs. You need to ensure that security requirements are enforced before any new system goes into production. This can be a painful battle with the business. Any deviations from the security requirements must be documented and the risks signed off by the business; otherwise, you will be accountable should a breach occur in the future. Let's face it, you will be held accountable regardless but at least you have documentation that you recognized the risk. Security design reviews, code reviews, security testing, and system penetration testing must be performed before a system goes live.

There will invariably be situations where you cannot patch or upgrade in a reasonable period of time. You will have to come up with some strategy to mitigate risk. Common options available to you are segmenting off the network where the system resides, virtualization, additional firewalls and/or firewall rules, and more intensive monitoring of the system in question. Many times, alternative solutions are provided for identified vulnerabilities, so if you cannot patch or upgrade, you must resort to using these alternative solutions.

We can't forget about the other critical piece of your foundation. The one that is exposed to the elements and is at a greater risk of failing. That is your mobile assets. While PCs are an obvious risk, mobile phones and tablets are now being rightfully recognized as very serious risks to the environment. All of your mobile end points need to be password protected and the data on them need to be encrypted. Users should not have any sensitive information on their mobile devices regardless of whether they are encrypted or not. The biggest concern would be passwords that may be stored on the devices. The less your employees have on their end points, the lower your risk.

Are your end points configured so they can be automatically updated with new software versions and antivirus signature files? Do you periodically scan the end points to ensure they have the latest software installed? More importantly, are you verifying that the software installed is legitimate? Automated update processes provide an opportunity for the bad guys to usurp the process and insert their own compromised software. Your networks should be configured to prevent any end point that does not have some minimum standard of software installed from connecting. Older versions of software were obviously replaced with newer versions due to security holes in the earlier versions. Why would you want to allow older versions to connect to your environment?

In summary: You are only as secure as your foundation. Like the foundation on a home, it requires periodic inspection and attention.

The bottom line is legacy software imposes significant risks on the company. The costs to address legacy software are small compared to the costs a security breach will incur. You have to make sure your foundation is intact. One small crack can bring an entire structure down.

Is your foundation well identified or is it scattered all over your environment? For example, are your sensitive data stored in a few key locations on your network or can sensitive information be found everywhere you look? Is it in databases or is it in files? Protecting a few critical databases is far easier than trying to protect thousands of files with sensitive information sitting all over your network. You should do everything you can to minimize the storage of sensitive information in files. Once a bad guy gets access to your network, then it is generally easier for them to peruse files on the network than it is to penetrate a database. At a minimum, sensitive files should be encrypted. If you must have sensitive files, then they should also be centrally stored with restricted accesses in place.

Remember, just because things appear to be working correctly does not necessarily mean it is true. A friend of mine owns an airplane. It had a slight vibration that was certainly troubling to him. He looked for all the obvious sources for the vibration for months with no luck. In the end, it turned out that one of the three propeller blades was corroded internally, so that it did not move to take a bigger bite out of the air as he climbed in altitude like the other two blades, thus throwing things out of balance. There was no way to actually determine this without getting inside the propeller operating mechanism.

As a CISO, it is your job to understand the legacy risk being experienced by your company. To see how secure your foundation is, you may have to tear a few things apart as well. Dig deep and don't just take a surface view of things. In industry, change is constant. Change requires vigilance. Keep your eye on both the inside and outside of your foundation.