Introduction to:

Basic Security Concepts

A Guide for Administrators and Home Users on the use and design of security for your network.

By: Robert H. Williams III
CompTIA A+\Net+, Microsoft MCP\MCSA\MCSE (2000), Cisco CCNA

Version 0.8.0

Document Contents:

Introduction

About this paper:

This document is about the basics of information security, design and implementation. This will be covering system and network security, but it is mostly for Windows security. Some of the information contained here-in is about abstract concepts in security design. Other bits of information are about actual configuration steps taken to lock down Microsoft Windows. To understand what needs to be configured, you should first look at why it needs configured.

The level of knowledge required to use this document is low. While some items covered may apply to certifications such as Security+, MCSE:Security, and even CISSP, this should not be considered a study guide for any of these certifications. It's more of a tie-in, to help the reader grasp the relationship of all the information presented for these levels, and an introduction for home users to higher levels of security.

Introduction:

Security is a hot topic for debate, from Congress and your company's local IT department, to Internet forums and even other home users, you can't seem to escape the topic of security. With areas ranging from *SOX* and *HIPAA*, down to the latest browser worm, there's a lot to cover, and much more ground to cover if you wish to understand it. And because most media outlets just do not understand what security is about, perceptions on security are skewered. This paper is designed to help bring some security concepts to light.

Prerequisites:

Most of this papers implementation section is on Windows, so you should use a Windows based machine for this part. Of the Windows machines, most of this only applies to the NT line of systems, NT, 2000, XP, and 2K3. And of course you will need to be the administrator on said machine. Being able to make network wide changes will help too.

But because most of this paper is about security design ideas, and risk mitigation, it can be performed on any OS. Much of the paper covers network design in general, and will never touch on the underlying OS.

Design Topics

What is IT Security about?

Information Technology (IT) security is all about information and protecting it. However, to protect the data, you need to know what is needed to protect it. Protecting data on your publicly viewed web site isn't the same as protecting your payroll data. One, you want people to see it, and the other, only a few people can see it. This is where the data types come into play.

What are the four data types?

There's 4 types of data a company can have, Public, Internal, Confidential, and Secret. The damage is done just by exposing this information in some cases. In others, it is only damaging if the data is changed.

Public:

If public data is exposed, then it's doing what it should be. If public data is changed or destroyed, however, you lose something you can remember by the letters *PTR*. This is not Pointer, but *Prestige*, *Trust*, and *Revenue*. Public data needs to be accessible, but only a few users or machines should be able to change it.

Internal (Private):

Internal data is data company workers generally know, but outsiders don't. It's items such as *PINs* for doors, internal procedures, or the like. It's information that most company workers can find out. Discovering this information is normally not a risk in itself, but it allows for better attacks. The main risk is modification, either by an attacker, or most cases, accidentally by an internal user. This will generally affect the operations of a business, and not much else. Most files on your OS would actually fall under this, and damage to them will only affect operations. But again, this can also be a stepping stone to attacks to other forms of data. Removing internal data from the view of workers can cause damages to business *Operations*, performing a *Denial of Service* (*DoS*) in extreme cases. For a home user, this could be almost anything you private you do not wish the rest of the family to know about, from personal information kept in a diary, emails to another person, or even URL history inside Internet Explorer.

Confidential:

This is the data used by a limited number of internal users, and should not be known to the majority of workers. This is the class Human Resources (HR) data and Payroll Information falls under. Read access to this data is limited to a few users, and write access is generally restricted even more. If this becomes public internally, *Operations* and *Internal Trusts* are at stake, while if reviled externally, you lose your *PTR*, along with *Operations* and *Internal Trusts*. OS files dealing with security also fall into this area in most cases.

Secret:

This is the data most people think of when they hear about breaches in information. This data is your trade secrets, intellectual property, and *External Secrets*, such as info held in trust for others (Partner company's, or customers). Loss of this data may cause critical damage to the company, and could very well be the downfall of it. Besides the *PTR* loss, and maybe loss of *Operations*, there's fines and legal actions to think of here.

While this may seem like only businesses would have data that fall in these four classes, all information can be placed inside them, sometimes into more then one class. As stated before, most of the files used by your operating system would fall under *Internal* data. It's not something that needs to be kept secret so much as needs to be kept from being changed. Music files on your machine? They have an effect on the *Operation* of how you run your life, and so fall under operations. Credit Card info could be considered *Secret*

What type of security model do most companies use?

Many businesses, and almost all home users, have a security model based on an egg shell. While the outer surface is hardened to keep people out, once there's a breach, there's nothing internal to prevent or limit access. And unlike the mighty chicken, companies have no way to make this security eggshell without holes in it. If a computer is online and networked, then any flaws in the firewall, along with any programs that access the network, form the security holes. If the computer is not online, physical access is still the hole in the shell. If it's a server in a locked room, the door itself is the hole in the shell.

This area of the shell you can attack is known as the *Attack Surface*. As you can imagine, a smaller surface is better, but this is only a tiny part of security. Most people and many companies think that the attack surface is the most important part of security, or even that it's all there is to security. This is putting all your egg-shelled networks in one basket. While it may be hard to get into the network, once in, everything is at the mercy of the attacker. It only takes one hole in the shell to get in, and as stated before, you can not prevent all holes in a shell.

The primary goal in IT security is to limit, not prevent, damage. While preventing damage is great, you can never prevent 100% of damage. Rather, you must try to make it harder to cause the damage, and work on lowering how often that damage can happen. While this concept is covered more in-depth later, a quick overview is as follows: all damage done can be assigned a monetary value. If your home computer was completely destroyed by fire, how much money would the damage be? While there is the cost of the computer itself, there is also the data inside the computer, damaged. Even if you recovered the physical cost of the computer, the data contained within could have more value of the computer itself. However, if you had an up-to-date backup of the data that was not destroyed in the fire, then the damage done would be the cost of a new computer, and effort required to restore said backups While you have not lowered how often the event may happen, you have lowered the damage done by it.

The egg-shell rule, while it applies to most companies, it applies to home users too. Think for a moment. Chances are, you're browsing the Internet right now, from the internal servers on your home or work network, ones with secret level data, as an administrator! And you have programs running as servers, such as yahoo messenger, with full administrative rights, and no damage control.

While having a strong shell is not only desirable, but almost required for security, other security steps must be taken. These include secondary shells under the first, access controls, not having systems trust other systems (Or even allowing one subsystem to trust another sub system).

How do most security breaches happen?

If you were expecting to see the word "hackers" then you've been watching too much television. Most security related issues are accidental issues. Blame does not always fall on a person, mind you. If the operating system crashes and is unrecoverable, chances are it was caused by a hardware error, but could have been caused from something as simple as installing a bad bit of software. While finding the true cause will help you prevent it from happening again, tossing blame around at anything you think caused it will not. If it's not an object that you can control, then blaming it is no good. If a hurricane wipes out your data center, do you blame the hurricane? No. But you can place blame on the location being in Florida, and to prevent this, you could move the base of operations to a new location. The problem with this, you may open it up to tornadoes or worse.

The most common security breech you hear about is cause by automated attacks, such as *Worms, Viruses,* and *Trojans*. Because these are automated attacks, they can take out a lot of machines very quickly, but they also can not adapt beyond what they were programmed to do, and can not compensate to defenses they were unprepared for. For example, if a worm was to try to log in as administrator, and you renamed the account, it would fail. However, any attacker worth his salt would be able to see the administrator account was renamed with ease. When implementing security, you should keep in mind, is the effort worth the gain? And will this effort make it harder to use the system, thereby raising the operational cost?

And damage is not always caused by external sources. Internal damage is a major issue. In terms of companies, most security breaches are internal, caused by users. This applies to things you may not take into consideration. It could be the VP of marketing meant to send an email to a friend, talking about how cute the new female hire is, only to accidentally send it to the whole office she works in. It could be that the newest patch for WinAMP accidentally changes some system files, preventing Windows from booting. Even hardware failures, from power outages, to blown CPUs, fall under security due to disruptions of operations.

The security triad: the other CIA.

When talking about information security, there's three things to keep in mind*: Confidentiality*, *Integrity,* and *Availability,* or *CIA*. Confidentiality comes into play when you do not wish for data to be known, and is generally controlled by encryption, access control lists (ACLs), firewalls, and the like. Integrity is making sure the data has not been altered or damaged. Signing of the data is the main way this checked, but access controls, and other security conventions, are required to prevent the change of data. Availability means that data must be up when needed, and can be referenced when required. Backups of data are the biggest thing here, but access control and redundant hardware are also major players here.

Returning to the four types of data, you need to know how the CIA applies to each. *Public* data does not need *Confidentiality*, but requires *Integrity* and *Availability.* Private data needs some *Confidentiality*, from outside sources. It also needs *Integrity,* and *Availability. Confidential* and *Secret* data need high levels of all three types of security.

Design vs. Implementation.

Note: This section is lengthy, and does not cover any topics of major importance. It is here just to help show you how poor design can be overcame with good implementation, and good design can be brought down by bad implementation. And a good design can help limit the damage done by a bad implementation.

When talking security, or many other issues in the IT field, Design vs. Implementation comes into play. Design is how well it's thought out, while Implementation is how you actually do it. When talking about Design vs. Implementation, one source that's always nice to read is the Tanenbaum-Torvalds Debate. If you did not know about it, Torvalds, as in Linus Torvalds, creator of the Linux operating system, and Andy Tanenbaum, creator of the Minix operating system. Linux's basic kernel design is that of a monolithic kernel, where there is a lot of unneeded functions stuffed inside the kernel. Minix is a micro kernel, where only the most needed of functions are in the kernel. The debate is interesting, because while Tanenbaum was preaching about the virtues of a micro kernel, Torvalds agreed with him. So why did Torvalds implement a monolithic kernel? Ease of implementation. While the micro was a better design, it was too hard to correctly develop for, and as such, it would never have been a completed operating system. So, while a monolithic kernel has inherent design issues, it's simpler to make work around for these design issues then on a micro kernel. While no one will say a monolithic kernel is more secure then a micro of the same level of completeness and same quality of design, in general the micro kernel took much longer to develop. The implementation helped overcome the design.

This should also let you see some issues. While it was simpler to develop, the monolithic design of Linux has limited it in many regards for security. Like with most consumer level OS's such as Unix and Windows, security was an after thought of Linux, and added in later in the development. The design of it isn't as secure as some other operating systems designed for security, such as Green Hill's Integrity operating system. But the security of each operating system's implementation is up to debate. Is a micro kernel with 1 security issue that allows administrative access more or less secure then a monolithic kernel with no issues? And what happens when the micro kernel's 1 security issue is now patched? Or, to put it in perspective, say operating system A has 2,000 known issues in it's lifetime, all stemming from about 80 flaws. System B is lesser known, and has about 450 known issues, but they stem from over 200 separate design flaws. What's more secure?

Well based on that information, and that information only, they are the same. What's easier to enter, a house with one door, or five? Well since you can only enter one door at a time, they're the same. But like all things, this isn't quite the whole truth. In operating systems, say you had 50 holes,. while the other operating system had 20. Well, that's 50 items you have to block, vs. 20. But say all 50 holes can be blocked via a firewall, while 1 of the other operating system's 20 issues can not be blocked via firewall, because it's inside a needed service. While the firewall isn't part of the OS, it allows it to block all the holes in the OS, while the other OS can not block all the holes.

One hole, or fifty, it doesn't matter, it's still a way in. Rather then plugging holes in your egg shell, then waiting for new ones that were always there to be discovered, then plugging them again, set up your defenses farther in, on the items needing protected themselves, then work outward, blocking all you can. The goal is to make it so no holes "line up", meaning what can get past one line of defense will get past the next. Say you have two firewalls in a line, both Cisco PIX, and an attacker knows a way to bypass them. Well if he can bypass the first, chances are he can bypass the second. By swapping the second PIX with another vendor, such as a ISA 2004 appliance, you shift where your "holes" are located. Now they need past the PIX and ISA. And if there is a new issue with your ISA server, the PIX will block it. If there's an issue with PIX, the ISA will stop it. Keep in mind though, if you have any sort of web access, you've poked your own holes in both of these shells, holes that lead right past both firewalls into your network.

Introduction to Risk Assessment.

Note: This is a very basic coverage of one type of risk assessment. It's not this simple, but this will give you the idea of how basic risk assessment works. There are other types of risk that can not be used on this scale. For example, risk that comes tied with potential gain, such as a new business venture, does not mesh well with this formula.

Risk Assessment is the practice of identifying risks to your business or assets, assessing the potential damage done, and recording how often said risk will occur. In *Risk Assessment* it's not if an event will happen, it's how often. There are three principle items to remember here, *Single Loss Expectancy, Annualized Rate Of Occurrence,* and *Annualized Loss Expectancy.*

Single Loss Expectancy (SLE) – This is the value of damage the average event of this type will happen. For example, if the average downtime of a server would be, say, 4 hours, and this down time causes $1,000 in loss per hour, then your *SLE* for this event is $4,000.

Annualized Rate Of Occurrence (ARO) – This is how often per year an event will happen. If your server goes down once for 4 hours per month, then you have an ARO of 12. If it went down for 4 hours every 4 years, then your ARO for this even is .25.

Annualized Loss Expectancy (ALE) – This is how much, per year, you are losing from this event. The *ALE* is generated by the *SLE* x *ARO* to generate your *ALE*. For example, with the *SLE* of $4,000, and an *ARO* of 12, your *ALE* is $48,000. If the *ARO* was .25, then the *ALE* would be $1,000.

Please note, this example was flawed, and treat yourself to bonus points if you saw why. Rather then assigning the *SLE* of $4,000 to a 4 hour event, and recording an *ARO* for each 4 hour event, you should assign it to a smaller variable, for example, a loss of $1,000 per hour. Then the *ARO* should be the total down time expected for the server, per year. This allows you to put in all server downtime as one item, and will ease the burden. But it seems real life follows my flawed example more than I would like. If server downtime is for extended periods of time, the *SLE* may actually raise. For example, many company have level 1, level 2, and level 3 downtimes. If a level 2 downtime happens, you begin calling in more people, people who may be on overtime. And on a level 3 downtime, it may be standard policy to call the manufacturer of the server, for instance. This causes a staggered rate for your *SLE* and can complicate matters. Because you have higher *SLEs* after longer outages, reducing the time of your average outage will pay off more then your average company. This is a time when talking to other departments, including the IT departments arch-enemy the sales department, and asking them what they think the loss will cause.

If your server had a 99.9% uptime in a *24-7, 365 day* operation, then out of 8,760 hours in the year, it was down for 8.768 hours. This translates into $1,000 x 8.768 = $8,768 average loss due to downtime. The way most people go about trying to lower this would be to raise uptime. If the uptime was brought up to 99.999%, downtime is 0.0876 hours, or about 5 minutes per year. This would result in the *ALE* of $1,000 x .0876 = $87.60. Sounds good right?

Well to get to a 99.999% uptime, or a downtime of 5 minutes per year, you would need a pretty hefty expense account. Chances are, your power company doesn't average 5 minutes of downtime per year to your location. A good UPS can only last so long, and backup generators can cost a bundle for servers. And this is just one item out of your control that can cause downtime. This magical *5 nines* of uptime is not an easy goal to reach year after year. To go from a 99.9% uptime to 99.999%, chances are very good that it will cost a lot. How much? Depends on a lot of items, but chances are, more then your $8,768 per year you lost per year. Is working on the uptime worth it? Might be, but remember, there's two parts to every *ALE*.

I've talked about security being not about just preventing something from happening, but limiting the damage done when it does happen. Now you can see the math. Say your server has a *SLE* of not $1,000, but $6,000 per hour. And you averaged 9 hours of down time last year, so your *ARO* is a nice 9. Your *ALE* for the year was $6,000 x 9 = $54,000. It was estimated that the cost to cut the downtime in half, to 4.5 hours per year, would be in the range of $250,000. By lowering the downtime, your *ALE* would drop to $6,000 x 4.5 = $27,000, and would save your company a like amount, 27,000 per year. With this rate, the changes to the uptime would start to pay off by the 10^{th} year, beyond the expected lifetime of the current setup. Next you look at lowering damage caused per outage. If you can lower the damage done per hour from $6,000 to $4,000, for example by having a less powerful backup server, then the *ALE* would become $4,000 x 9 = $36,000, a savings of $18,000 per year.

What's simpler to do, lower the *SLE* or the *ARO*? Depends on the events in question. If you're in the banking industry, and are processing information from external company, uptime is a requirement to begin with, and you may very well be forced to strive to have huge amounts of uptime. In this case, worrying about the *SLE* may not be an issue, since the *ARO* is already required to be at a set level. But it does not mean you should place all your eggs into one basket, to speak of. Sometimes the *SLE* of an event could become so high, if it does happen, critical damage to the company could happen. In this cause, you must try to find a way to limit the damage caused. It's a trade off in every aspect, so keep in mind the benefits of looking at the problem from every angle. You may find out you've missed a few easy fixes.

This principal, lowering the damage done, or how often damage is done, applies to most aspects of security. Lowering the damage an attacker can do will often pay off better then trying to keep them out. The reason is simple, when an attacker does get in, the damage could be critical to the company. You may have the *ARO* raised to the point the event will happen once in a billion years, but you can never be sure that the one in a billionth year won't be today. And if it takes all of your assets with it, then tell me just what is the *ALE* of Infinity x .000000000001 is?

Implementation Topics

Backing up of data: types of backups and data formats.

In security, the biggest thing to do to prevent and limit damage is to set up a backup strategy. This limits the damage done by loss of *Availability*, and also allows you to reverse changes in *Integrity.* A good backup strategy would be a full back up every weekend, with an incremental or differential backup every day. This means no matter what happens, you only lose one days worth of data, plus any data you would have gained during the recovery process.

There are three major types of backups, *Full Backup*, *Incremental Backup*, and *Differential Backup*. The major difference between these three is what they do with a file depending on it's *Archive Attribute*. When a file is changed, the *Archive Attribute* is turned on. This is used by your backup software to determine if a file should be backed up or not.

When you perform a *Full Backup*, it backs up every file, no matter what the *Archive Attribute* is set to. It then clears the *Archive Attribute*. To perform *Incremental* and *Differential Backups*, you need a *Full Backup* to start from.

When you perform an *Incremental Backup*, it only backs up files with the *Archive Attribute* set on. It then clears the *Archive Attribute*, so during the next *Incremental Backup*, the files will not be backed up unless they have changed.

When you perform a *Differential Backup*, it works like an *Incremental Backup* in that it backs up only files with the *Archive Attribute* set. However, it does not reset the *Archive Attribute*.

Why use one type of backup over the others? Well, they all have advantages and disadvantages, and the disadvantages can be limited by using more then one type. You always need a full backup. A *Full Backup* is slow to perform, and takes a lot of space. But if you wanted to recover from a system issue, the *Full Backup* is the fastest. An *Incremental Backup* is the fastest to backup to perform, but may be the slowest to restore from. A *Differential Backup* is the inverse of the *Incremental Backup,* its up more space then *Incremental Backup,* but recovers faster. Remember, *Incremental* and *Differential Backups* contain all changes since the attribute bit was changed, via *Full Backup* or *Incremental Backup*. Using *Incremental Backups* daily mean each backup only contains that days backup. Using *Differential Backups* every day means

The concept of how these three backups come into play is best described via example. Say every Sunday, you perform an automated *Full Backup*. Every Monday, Tuesday, Thursday, and Friday, you perform a *Differential Backup*, and perform no backup on Saturday, due to the offices being closed. And every Wednesday, you perform an *Incremental Backup*. Each day, 50 megabytes of data is changed, and the total data on the server is 1 gigabyte. Well, every Sunday, you make a backup of 1 gig in size, with every bit of info saved, and then reset the *Archive Attribute*. On Monday, you perform your *Differential Backup*, and back up the 50 megs work of changes, but do not change the *Archive Attribute*. On Tuesday, you perform your second *Differential Backup*. This backup will record all changes to the files since Sunday, so in this case, it will back up 100 megs, not 50. Wednesday comes along, and you perform your *Incremental Backup*. This requires you to back up 150 megs of data, 50 each from Monday, Tuesday, and Wednesday. But on Thursday, you only back up the 50 megabytes of files that changed since Wednesday, and your Friday backup is 100 megabytes.

Using this rotation, you have the following backup sizes:

Day:	Backup Type:	Backup Size:	Data Stored:
Sunday	Full Backup	1024 Megabytes	All Data
Monday	Differential Backup	50 Megabytes	Monday
Tuesday	Differential Backup	100 Megabytes	Monday\Tuesday
Wednesday	Incremental Backup	150 Megabytes	Monday\Tuesday\Wednesday
Thursday	Differential Backup	50 Megabytes	Thursday
Friday	Differential Backup	100 Megabytes	Thursday\Friday
Total:		1,474 Megabytes	

Now then, say you need to restore the server to Tuesday's backup. First you would apply the last *Full Backup*, Sundays. Next, you would apply Tuesday's backup. If you wanted to restore to Friday, you would apply your full, then Wednesdays, then Fridays. Another rotation would be all *Incremental Backups* during the week:

Day:	Backup Type:	Backup Size:	Data Stored:
Sunday	Full Backup	1024 Megabytes	All Data
Monday	Incremental Backup	50 Megabytes	Monday
Tuesday	Incremental Backup	50 Megabytes	Tuesday
Wednesday	Incremental Backup	50 Megabytes	Wednesday
Thursday	Incremental Backup	50 Megabytes	Thursday
Friday	Incremental Backup	50 Megabytes	Friday
Total:		1,274 Megabytes	

This backup strategy has the smallest total size, and the backup is performed the fastest during the week, only needing 50 megabytes per backup. The disadvantage in it is recovery, while recovering to Monday is just as fast as recovering to Monday using a *Differential Backup*, it gets slower by Friday, since you not only have to recover from Sunday's *Full Backup*, but also restore every single day up to Friday.

And the last format, all *Differential Backups* during the week:

Day:	Backup Type:	Backup Size:	Data Stored:
Sunday	Full Backup	1024 Megabytes	All Data
Monday	Differential Backup	50 Megabytes	Monday
Tuesday	Differential Backup	100 Megabytes	Monday\Tuesday
Wednesday	Differential Backup	150 Megabytes	Monday\Tuesday\Wednesday
Thursday	Differential Backup	200 Megabytes	M\T\W\Thursday
Friday	Differential Backup	250 Megabytes	M\T\W\Th\Friday
Total:		1,774 Megabytes	

This type of backup uses up the most space, but it is the fastest type to recover from, baring *Full Backups* daily. If you need to recover Fridays data, just pop in Sundays tape, then pop in Fridays tape.

What format should you use? That's up to you. How often do you need to recover? And how quickly do you need to recover? Also, keep in mind that Differential Backups assume that each day, it's 50 megabytes worth of different files being changed. Chances are, the same file was changed more then once. Because of this, the final backup on Friday will be smaller than you would have expected.

Now that you know about the types of data backups, there's the media to worry about, and what to do with the backups. In general, this will be completely different from a home user and office worker backing up a server.

<!----------TBC----------!>

Slimming down the attack surface: Firewalls

While backups can control the damage done to a system, you need some sort of security to keep the majority of attacks from getting in. This is where firewalls come into play.

When looking at an objects security, you have a limited number of ways in. This is called the *Attack Surface.* When dealing with automated attacks, a smaller attack surface is better, since it limits the number of attacks that will work, causing many worms from getting into the network. But against a determined attacker who knows what he's doing, these are generally just going to delay him, not stop him., because all it takes is one path in to compromise the system.

Even with that doom and gloom there, a firewall is one of the most important features of security. There is nothing that can shrink down the *Attack Surface* of a networked machine better then a firewall. But firewalls come in different types, and perform different functions, and can be configured and deployed in so many different ways, it's mind boggling. And many people seem to think of devices that are not true firewalls, such as your NAT-enabled router, are protecting them, when in most cases it's trivial to bypass them, mostly via blind spoofing attacks.

Types of Firewalls: Network vs. Client

Note: This is a very basic, very quick overview of firewall types. It's recommended the reader looks into more in-depth information on the subject. The firewall is the single most important step in controlling your attack surface, and it should be considered a mission-critical item.

Most firewalls fall under two types of classification, *Network Firewalls* and *Client Firewalls*. A *Network Firewall* goes on the outer-edge of a *Zone,* or *Network,* and protects it from outside attacks, while maybe also preventing internal traffic from making it outside. A *Client Firewall* runs on a local machine, and protects the local machine from attack, and possibly protecting other systems from attacks by it. It's important to keep in mind that these two types of firewalls perform different duties, and should be configured as such. But a *Client Firewall* can be used as a *Network Firewall* if required. This is generally done inside home networks, where a dedicated *Network Firewall* would have crossed the line of diminishing returns.

Network Firewalls

When using a *Network Firewall*, your goal is to protect the network behind it from attack, and maybe to keep sensitive information on the inside from leaking out. A *Network Firewall* would be any firewall at the entry point to a network (at the default gateway\gateway of last resort). *Network Firewalls* come in many types of configurations, but there are three main configurations of *Network Firewalls* talked about in this paper: *Single Firewalls*, *Three-Pronged (*also called *Three-homed*), and *Demilitarized Zone Firewalls* (*DMZ*).

The classification of these firewalls is simple. A *Single Firewall* is just simply one *Network Firewall* preventing access to the network, and contains two sides: *Internal* and *External*. A *Three-Pronged Firewall* is a firewall with three sides, not the two of a traditional firewall. It contains the *Internal* and *External* connections, but also contains a third connection. This third connection is used for *Public Servers*, and generally as less stringent security rights on data heading to the servers then data headed to the internal side of the network. It should be noted, many home-based routers use the term *DMZ* in place of a *Three-Pronged Firewall*, but they are not true *DMZ* setups. A *DMZ* is not actually one firewall, but two network firewalls in a row, one connecting to the *External Network*, and the actual *Demilitarized Zone* itself, and a second firewall connecting from the *Demilitarized Zone* to the *Internal Network.* The *Demilitarized Zone* between the two firewalls is where you would place *Public Servers*. This setup allows you protection for your *Internal Network* if the first firewall is breached, or is one of the *Public Servers* are compromised. It should be noted, of these three configurations, there is only one firewall at all times between *Public Servers* and the *External Network*.

When configuring your *Network Firewalls* in a *DMZ* setup, it's recommended that the two firewalls be of different brands. This is in case one of these brands can be bypassed by a vulnerability, you still have the second firewall up to protect you. And because different *Network Firewalls* have different abilities, this allows you more control over your network. Examples of *Network Firewalls* would be Cisco's PIX, Microsoft's ISA, Linux's IPTables, and the offerings from Checkpoint.

Client Firewalls

Client Firewalls, unlike *Network Firewalls*, are just to protect the local resources on a device. They generally offer more options for outbound control then *Network Firewalls*, but do not have the capabilities of their “big brothers”. A *Client Firewall* isn't as important for security as a *Network Firewall*, and they should not be used in place of them, with a few exceptions. But a *Client Firewall* is a nice piece of software to run inside the network, as it can prevent compromised devices from compromising other devices on the network. This can limit the infection of a *Worm* or *Virus* from spreading to other machines, and add in the difficulty of an attacker moving from one machine to another. But because remote administration can be blocked by their use, they add one more layer of complexity that may be of diminishing returns to the user.

In a home network, *Client Firewalls* take on new importance. Because few people would wish to configure a dedicated *Network Firewall*, they run a *Client Firewall* in its place. Sometimes, they have one primary machine connected to the *External Network*, and it uses it's *Client Firewall* to block any attacks to itself. All client computers then connect to this machine, turning the entire first device into a *Network Firewall*. However, since this first computer has more functionality then a traditional dedicated firewall, there are more holes in its *Attack Surface*. Because setting up a dedicated *Network Firewall* can be as simple as downloading *IPCop* and placing it on an old computer, this practice of using a *Client Firewall* isn't recommended, but for a home network, it's value on returns is debatable.

It should be noted, most home routers are not valid *Network Firewalls*. Most of these devices are simply *routers* that perform *NAT.* Many of these devices will route packets from the *External* side to the *Internal* side, even when not desired. For example, say your internal network is 192.168.0.0 network, and the *External* interface of your router received a packet with the source IP spoofed to be 192.168.0.8. Many home routers would route this packet into the internal side. To make things even worse, many *Client Firewall* would trust all clients on the local network, allowing this packet to bypass almost all home security. Because *IP Spoofing* is a *Blind Attack*, the value of this attack is somewhat limited, but the threat does exist. And because many ISPs have large *Local Loops*, there is a surprisingly large number of attackers who could perform such an attack. Remember, in many cases, you're only as secure as the person next to you.

Security in the Microsoft Windows operating system.

Note: Most of this applies only to the Windows NT line, such as NT, 2K, XP, and 2K3. Some may even only apply to XP and 2K3. Anything just on 2K3 and XP will be marked.

How do you limit security issues on a Windows system? Well, first the network it is on must be secured. Sometimes however, this can't be done. If you're a home user with a cable modem and router, chances are you're not going to buy two hardware firewalls for a DMZ configuration. So you have to make due.

To secure your Windows systems, we'll start with some of the biggest issues, and work from there. Firewalls have been already covered at the network level, and backups have also have been covered, so these are assumed to be in place. Some information directly pertaining to Windows' built in firewall will be talked about later, but for now we'll cover some new topics.

Running Microsoft Windows under a Least User Privilege account.

Note: Vista has changed how user accounts work when using administrative rights. These steps no longer apply the same on Vista as older Windows operating systems. Vista for example, will prompt you for an administrators name and password automatically when requesting administrative rights, making many of these tasks simpler.

One of the most important steps in running a secure Windows device is to not run programs while you're on an Administrative account. In Windows, when you log in, it creates a security token. This token has your user name, lists what rights you have, and what groups you belong to. After the token is created, any application spawned from the application holding the token will use a copy of the original applications token. Any application my drop rights from it's token, or generate a new, lesser token to pass on to other applications, to limit what they can and can not do. But you generally can not add rights to a token, you need to authenticate to the system, and be handed a new token. This is why when you add yourself to a group, you need to log off, and back onto the system, to generate a new token. In truth, this is only partly true, you only need to log in, you can actually skip the log off part. This trick is used later in the MakeMeAdmin.bat file that will allow you to generate a command prompt running as your current user, but as an administrator. And do not worry, this batch file is not a security risk, you still need passwords to do this.

For home users, setting up a least-user privileged account, in other words, a normal, not administrative-user account, seems like a daunting task. And it's generally thought of that most programs won't run as non-administrators. I actually can think of no programs that will not run as a non-administrator, baring programs designed for administrative functions. Installing said programs is a different story, mind you, but that's easy to overcome.

The simplest way to change to being a least privileged user is to make a new administrative account, if you do not already have a new one. This new account must have a password, and not just because of security reasons, but due to built-in security in XP and 2K3 that will prevent you from using it in our examples. Accounts with no passwords can not log in remotely, and using the run-as command uses terminal services, and acts like a remote log-in. Once you have made this new account, you simply remove your account from the administrative users group (Or set it to a limited account, depending on the terminology you wish to use). Now when you log on, all applications should still work the same. One major program that may give you trouble is Nero Burning Rom. It actually has a work-around though, an application called Nero Burning Rights. Download it, run as an administrator, and it will allow you to grant rights to your normal account.

When running as a non-administrator, you now have two problems, installing programs, and performing system tasks. And, a few rare, older programs toss in a new problem: Settings that are set per-user, but can only be changed by an administrator. While you could always log in as an administrator to perform installs and system tasks, a simpler way would be to give yourself a way to make your own user account an administrator. This is where the batch file MakeMeAdmin.bat comes into play. This batch file is as follows:

```
setlocal
set _Admin_=%COMPUTERNAME%\<ADMINISTRATIVE ACCOUNT>
set _Group_=Administrators
set _Prog_="cmd.exe /k Title *** %1 as Admin *** && cd c:\ && color 4F"
set _User_=%USERDOMAIN%\%USERNAME%
if "%1"=="" (
		runas /u:%_Admin_% "%~s0 %_User_%"
		if ERRORLEVEL 1 echo. && pause
) else (
		echo Adding user %1 to group %_Group_%...
		net localgroup %_Group_% %1 /ADD
		if ERRORLEVEL 1 echo. && pause
		echo.
		echo Starting program in new logon session...
		runas /u:%1 %_Prog_%
		if ERRORLEVEL 1 echo. && pause
		echo.
		echo Removing user %1 from group %_Group_%...
		net localgroup %_Group_% %1 /DELETE
		if ERRORLEVEL 1 echo. && pause
)
endlocal
```

To use this on your own system, you must change <ADMINISTRATIVE ACCOUNT> (Second line) to your administrative account's name. When you need to be an administrator, rather then using runas for the new account, you run this batch file. It can be a bit of a pain, you need to enter in the administrative account's password, then your own password (both accounts must have passwords for this to work). After completing this, a new CMD window pops up. It is running as your user account, but it is in the administers group, and has administrative rights because of it. Now the thing to keep in mind, when one program (Such as CMD, or Explorer) launches a program, it generally copies it's own security token to the new program. Any program launched from this CMD window is ran as an administrative user. If you need to install a program, or make changes as an administrator, run this batch, then launch the program. If you need to click on something as an administrator because there's no command line application that you know of to do it, such as something in the control panel, it's easy. Run %programfiles%\Internet Explorer\IExplore.exe and type the address of the program. Want to set permissions without using cacls? Enter the address into IExplore's address bar, such as c:\program files. To get to the control panel, just type Control Panel in the address bar. From there you can run any applets.

Too much work on your low physical security home machine? Get an application that runs in the tray, and has a file command. One is taskmgr, built into windows, so we'll cover this one. Launch taskmgr from the administrative CMD window, then in view, select hide when minimized. Then you can hit file, new task any time you want an administrative window, no need to retype passwords. This is not recommended on a production machine, but for a home machine it works just fine.

Locking down the browser: Internet Explorer Configuration

Internet Explorer is the default browser for Windows. People talk a lot about security issues with Internet Explorer,
(**********)

Next step. Stop installing so many damn programs. This is like, #1 thing I think for most home users, ever more so then browser security. Install programs in a test bed.

Now lockdown the browser. I know people think this is because they like to think I'm an MS fanboy, but I recommend Internet Explorer for security. Open tools -> Internet Options -> Security. Set internet zone to High security. Set Trusted zone to mid, then hit add site. It the require HTTPS is checked, uncheck it, then add in *.Jinx.com. Using the same setup, add in any sites you trust that need scripting. For example, *.microsoft.com (They made your freaking os, so you better trust software they put on there.)

Guess what? Done. While there's a lot of things you can do, such as assigning services to limited user accounts and the like, this is the major parts right here.
<!----------!>

Locking down the hard drive: NTFS Permissions

One of the most powerful security tools in Windows is the basic NTFS permission. It's also a very misunderstood security device. Besides a lack of general info on the web, many sites just list the process of permissions wrong, giving users the impression that something shouldn't work, only to find it does. This section will look into NTFS permissions in depth, and cover how other permissions, such as registry and share permissions, work.

In order to use NTFS permissions, you of course need to be running NTFS. To get the GUI based editor for of NTFS, you may have to jump through a few hoops. But believe me, this is one item you need the GUI for, as you're not going to getaway with a simple chmod 777.

To begin with, there is three basic levels each permission could have: Allow, Deny, and Revoke, or no permission. While it may seem like Deny and Revoke work the same, they do not, and this can cause trouble for some users. Allow, as you can understand, means the user can access this function. Deny means the user can not access it. Revoke means that nothing is set on this level, and so it should check this objects parent object, until the root object is found. If the user has no permission anywhere up to the root, then permission is denied.

An example would work best here. Say within the folder c:\Data\Sales\Market Data, there is a file, October.csv. Bob, a VP in the sales department, wishes to delete this file. Windows would first check to see if Bob has permission to delete it listed on the file itself. If no setting was found (Revoke) then it checks the files parent folder, Market Data. In this case, there is still no setting found, so Windows checks the next object in the tree, Sales. In this case, we'll say that the group Sale VP Group, a group Bob is a member of, has Full Control of the Sales folder. Bob, however, is known for not being the best when it comes to following security rules, and is also a member of untrusted users. This group is denied the delete permission, and to prevent the deletion from happing anywhere by any member of this group, the deny permission is added on the Data folder, one level higher then Bob's Allow. So, what will happen?

Bob will be allowed full access and can delete whatever he wishes. Once Windows finds an entry, be it Allow or Deny, it stops processing any more permissions. The permission closest to the object wins. This applies to all Windows security systems, with one exception. When processing group policy permissions, permissions can be set at Site, Domain, Parent OU, Current OU, and on the Local Machine. While Local Machine is the closest, it's permissions only come into play if no other settings come into play. If Local Machine policy overwrote the domain policies, then you could never be able to apply any sort of polices at higher levels, the default polices would always be in place.

While it seems so far that permissions are cut and dry, there's a few special rules. One, a Deny overrides an Allow. If you have Deny and Allow at the same level, the Deny wins. But as shown in this example, is an Allow is closer to the object then a Deny, then the Allow wins.

Next rule is something you must be very careful using. You can block inheritance of permissions on any object, causing it to become it's own root. For example, say you had a folder, c:\Allow Everyone\Deny Everyone. On the Allow Everyone folder, you had given full control over to the group Everyone. On Deny Everyone folder, you decided to block inheritance from the parent folder, and pick the clear option when prompted. You then set up a share on the Deny Everyone folder. On the Deny Everyone folder, you grant everyone full control of the Share permission, and grant the Everyone group the Read permission.

Making your own programs support Windows security models.

There is a lot to learn about Windows security if you wish to administer a system. If you want to make applications for Windows, you should understand Windows security, in addition to being able to program. But it is a rare person indeed who programs for a living, and understands system security. Hence, so many Windows applications break simple security rules, because they think the way they're doing it is better, without knowing about the items they break in doing so.

This section will cover one tiny part of application security in Windows. It's not talking about how to make your application crash proof, or how to prevent buffer overflows, or how to protect your heap. This is about where you should save your data, and why.

Many applications pride themselves in not modifying the registry in Windows, instead, they save all data and settings inside the programs subdirectory. While this is nice for smaller applications, it actually causes complications for administrators, and has security related issues that can prevent the application from being deployed over a network, and adds extra work for the administrator.

You may be wondering why? Simple, permissions, and per-user settings. When a program is installed in the Program Files folder, normal users only have read and execute permissions within this directory. This is designed for a few reasons: prevent users from changing applications, prevent users from installing applications, and prevent viruses infecting a user process from changing the contents of the Program Files folder.

To understand how Program Folders is intended to work, you need to keep in mind, data and settings for programs should be per-user. Global settings should only be changed by administrators. Because normal users have no right to write to the Program Files folder, no program should ever write to this folder once installed. The only exception would be to store global settings, but these should be stored within the registry, for reasons to be covered soon.

All data from an application should be stored inside the users profile by default. If you want data from an application to be shared, the user could place it inside their shared documents folder, if they have one. Otherwise, a share or folder would need setup with permissions to allow the correct users access to it. Learn how to find the users profile (It's not always c:\documents and settings\<username>) and use this for the default location. And make sure your program can handle the error if the user tries to save, and the save fails.

Now you know where data should be stored, what about settings? Settings should always be stored inside the registry. Why? Because it is much easier to apply changes to settings in the registry then via files. Besides being able to apply .reg files, using the REG command line, and hand editing local and remote machines, the Windows Registry has one very powerful editing mechanism: Group Policy. Ever edit anything within Local Security Policy snap-in? Most of these options are stored in the Registry. The local security policy snap-in just makes some of the most common items easier to see. But that is just a GUI, it doesn't actually add any sort of power. The real power comes when you save these changes into a security template, and\or place them within a Group Policy Object (GPO). Security Templates are registry edits, registry permissions, and NTFS permissions. You can then apply them to the local machine, giving you a stored baseline. You can also audit machines against your security templates, to make sure the settings you say should be in place, are in fact the settings in place. While this is a nice function, it pales in comparison with the power of Active Directory and Group Policy.

A Microsoft Windows Domain is called an Active Directory (AD) Domain. Active Directory is a powerful tool, and has been one of the major reasons Microsoft has such a strong group on the desktop market for businesses. Active Directory has within it containers, called Organizational Units, or OUs. OUs store User Accounts or Computer Accounts, and can be used to group together people and machines that need settings the same. You apply group policy to OUs, and it is applied to every user and device within the OU. This means that anything you set via Local Security Policy can be applied to one, ten, or even ten thousand machines, with the same amount of work. But it gets better.

The options in local security policy, and in Group Policy, are simple text files that you can make on your own. This means you can add in your own Group Policy, and allow your program to be maintained via Group Policy.

To learn more about the Microsoft Windows Registry, and how you can make your program Group Policy compatible, please see the document http://www.security-forums.com/viewtopic.php?t=34277

Common Security Misconceptions: Unlearning what you know.

There are many misconceptions in information technology security. Some ideas are just plain wrong, some were good but have since become outdated, and others are just good ideas with poor logic behind the implementation. This section will be covering a few of these misconceptions, with hopes of clearing some of them up.

Misconception #1: Biometrics *are the best security development ever* - Biometrics *are the worst security idea ever.*

In security, you will find two opposing views on the topic of *Biometrics*, those that love it, and those who hate it. The ones who love it are generally the ones who are new to security, and have learned most of it from mass media. They think it should be used to replace passwords and pin numbers as a way for granting access. The ones who hate it are the ones with slightly more info, and have learned it from peers with little to know security background. The most common complaint is the lack of ability to change "passwords". So, the truth of the matter?

The truth is, *Biometrics* are a great asset to security, and will used more and more over the years, it's just the implementation of biometrics is different then what people think. Biometrics is not a password, and it should not be used as one. Biometrics is used to prove that the person entering the password is the one given the password.

Most authentication systems use Usernames\Passwords, or smartcards, or some other method that works on the same principals. The problem with username and password combinations, they are stored just as knowledge, and can be freely given away. While you can prove that it was X person's account that was used, you can not always prove it was X person who did it. Smartcards are slightly different, they are physical items. If you give one away, you lose your own access to the system, and so you are likely to report a missing card rather quickly. Many smart cards have photos of the user on them, so they can be matched to the person using them, but these photos are almost never used by the authentication device itself.

Biometrics allows you to, with a reasonable amount of error, say that X person was the one who accessed the services. You can not give away a finger print or facial scan, and so it helps to proves you are the person you say you are. It is not, by itself, the means of authentication , that should still be performed by a password or pin number.

The biggest problem with biometrics currently isn't the design, as most design issues, such as reply attacks, have solutions. It's the implementation of the biometrics themselves. Some biometrics use anti-reply schemes inside them, where if a scan of the data PERFECTLY matches another scan, it is rejected. This however, is a rare function for most scanners. Another feature is the ability to rescan the stored key, your finger print for example, using different points of reference, creating a new hash value in case the old one is compromised. This only comes into play if the scanner was attacked itself, and the data was sent directly to the authentication device.

Some issues that people quote about biometrics that are not issues is the ability to change the "password". If the "password" is compromised, they think you can not change it. This isn't true, you just have a limited number of changes before you have to reuse passwords: 9 changes to be exact. There is no reason why you must use the same finger all the time, and few networks ever enable a password history enforcement of more then 9 passwords (But the default is 24 in a Windows Active Directory network, so those who do enforce password history will generally have the history higher then 9 before allowing repeats) Also note that the biometrics is not the *Secret* in the authentication, it takes the place of the user name. User names are not secret, and do not need changed. In many companies, they are generated based on the users name, and are used for the users email address.

Many lesser quality scanners can be fooled via simple tricks, even to the point of photocopied finger prints. It should be noted that tricks such as these only work on cheaper models, and as the technology advances, detection of said attacks will improve. Newer scanners can even examine the pattern of blood vessels using infrared, making simple lifting of a finger print useless.

Glossary of Terms:

Attack Surface – The area that an attack can come through, or, the visible area of a target the attacker can try to compromise. It is the sum of all the possible ways in an attacker can use to get into the the system.

SOX - The Sarbanes-Oxley Act of 2002, also known as Public Company Accounting Reform and Investor Protection Act. This is a United States federal law, it's primary goal is to force companies to evaluate their *Internal Controls* for finical reporting, and that a third party (In general, an Auditing Firm consisting of *CPAs)* confirm that reported practices and implemented practices do indeed match. See: http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=107_cong_bills&docid=f:h3763enr.tst.pdf and http://www.legalarchiver.org/soa.htm for more detailed information.

HIPAA -

Denial of Service (*DoS*) - An attack that prevents normal usage of a service. It could be caused by using up all of your companies bandwidth from the ISP, or purposely logging into an account repeatedly with the wrong password to lock out the account.

References used:

Microsoft Course #2050A
Designing A Secure Microsoft 2000 Network

Open Sources: Voices from the Open Source Revolution: Appendix A - The Tanenbaum-Torvalds Debate
http://www.oreilly.com/catalog/opensources/book/appa.html

IPTables Connection Tracking - FTP
http://www.sns.ias.edu/~jns/wp/2006/01/24/iptables-how-does-it-work/?p=20

Linux-PAM modules etc. page
http://www.kernel.org/pub/linux/libs/pam/modules.html

Backup types
http://www.backup4all.com/backup_types.php

Applying the Principle of Least Privilege to User Accounts on Windows XP
http://www.microsoft.com/technet/prodtechnol/winxppro/maintain/luawinxp.mspx

Using a Least-Privileged User Account
http://www.microsoft.com/technet/security/secnews/articles/lpuseacc.mspx

The Services and Service Accounts Security Planning Guide
http://www.microsoft.com/technet/security/topics/serversecurity/serviceaccount/default.mspx

MakeMeAdmin -- temporary admin for your Limited User account
http://blogs.msdn.com/aaron_margosis/archive/2004/07/24/193721.aspx

MakeMeAdmin follow-up
http://blogs.msdn.com/aaron_margosis/archive/2005/03/11/394244.aspx

Notes:

Sed Quis Custodiet Ipsos Custudios, "Who will guard the guards?