Executive Summary

Much ado has been made about whether or not Linux is truly more secure than Windows. We compared Windows vs. Linux by examining the following metrics in the 40 most recent patches/vulnerabilities listed for Microsoft Windows Server 2003 vs. Red Hat Enterprise Linux AS v.3:

1. The severity of security vulnerabilities, derived from the following metrics:
    a. damage potential (how much damage is possible?)
    b. exploitation potential (how easy is it to exploit?)
    c. exposure potential (what kind of access is necessary to exploit the vulnerability?)
2. The number of critically severe vulnerabilities

The results were not unexpected. Even by Microsoft's subjective and flawed standards, fully 38% of the most recent patches address flaws that Microsoft ranks as Critical. Only 10% of Red Hat's patches and alerts address flaws of Critical severity. These results are easily demonstrated to be generous to Microsoft and arguably harsh with Red Hat, since the above results are based on Microsoft's ratings rather than our more stringent application of the security metrics. If we were to apply our own metrics, it would increase the number of Critical flaws in Windows Server 2003 to 50%.

We queried the United States Computer Emergency Readiness Team (CERT) database, and the CERT data confirms our conclusions by a more dramatic margin. When we queried the database to present results in order of severity from most critical to least critical, 39 of the first 40 entries in the CERT database for Windows are rated above the CERT threshold for a severe alert. Only three of the first 40 entries were above the threshold when we queried the database about Red Hat. When we queried the CERT database about Linux, only 6 of the first 40 entries were above the threshold.

Consider also that both the Red Hat and Linux lists include flaws in software that runs on Windows, which means these flaws apply to both Linux and Windows. None of the alerts associated with Windows affect software that runs on Linux.

So why have there been so many credible-sounding claims to the contrary, that Linux is actually less secure than Windows? There are glaring logical holes in the reasoning behind the conclusion that Linux is less secure. It takes only a little scrutiny to debunk the myths and logical errors behind the following oft-repeated axioms:

1. Windows only suffers so many attacks because there are more Windows installations than Linux, therefore Linux would be just as vulnerable if it had as many installations
2. Open source is inherently less secure because malicious hackers can find flaws more easily
3. There are more security alerts for Linux than for Windows, therefore Linux is less secure than Windows
4. There is a longer time between the discovery of a flaw and a patch for the flaw with Linux than with Windows

The error behind axioms 3 and 4 is that they ignore the most important metrics for measuring the relative security of one operating system vs. another. As you will see in our section on *Realistic Security and Severity Metrics*, measuring security by a single metric (such as how long it takes between the discovery of a flaw and a patch release) produces meaningless results.

Finally, we also include a brief overview of relevant conceptual differences between Windows and Linux, to offer an insight into why Windows tends to be more vulnerable to attacks at both server and desktop, and why Linux is inherently more secure.

Busting The Myths

Myth: There's Safety In Small Numbers

Perhaps the most oft-repeated myth regarding Windows vs. Linux security is the claim that Windows has more incidents of viruses, worms, Trojans and other problems because malicious hackers tend to confine their activities to breaking into the software with the largest installed base. This reasoning is applied to defend Windows and Windows applications. Windows dominates the desktop; therefore Windows and Windows applications are the focus of the most attacks, which is why you don't see viruses, worms and Trojans for Linux. While this may be true, at least in part, the intentional implication is not necessarily true: That Linux and Linux applications are no more secure than Windows and Windows applications, but Linux is simply too trifling a target to bother attacking.

This reasoning backfires when one considers that Apache is by far the most popular web server software on the Internet. According to the September 2004 Netcraft web site survey,[1] 68% of web sites run the Apache web server. Only 21% of web sites run Microsoft IIS. If security problems boil down to the simple fact that malicious hackers target the largest installed base, it follows that we should see more worms, viruses, and other malware targeting Apache and the underlying operating systems for Apache than for Windows and IIS. Furthermore, we should see more successful attacks against Apache than against IIS, since the implication of the myth is that the problem is one of numbers, not vulnerabilities.

Yet this is precisely the opposite of what we find, historically. IIS has long been the primary target for worms and other attacks, and these attacks have been largely successful. The Code Red worm that exploited a buffer overrun in an IIS service to gain control of the web servers infected some 300,000 servers, and the number of infections only stopped because the worm was deliberately written to stop spreading. Code Red.A had an even faster rate of infection, although it too self-terminated after three weeks. Another worm, IISWorm, had a limited impact only because the worm was badly written, not because IIS successfully protected itself.

Yes, worms for Apache have been known to exist, such as the Slapper worm. (Slapper actually exploited a known vulnerability in OpenSSL, not Apache). But Apache worms rarely make headlines because they have such a limited range of effect, and are easily eradicated. Target sites were already plugging the known OpenSSL hole. It was also trivially easy to clean and restore infected site with a few commands, and without as much as a reboot, thanks to the modular nature of Linux and UNIX.

Perhaps this is why, according to Netcraft, 47 of the top 50 web sites with the longest running uptime (times between reboots) run Apache.[2] None of the top 50 web sites runs Windows or Microsoft IIS. So if it is true that malicious hackers attack the most numerous software platforms, that raises the question as to why hackers are so successful at breaking into the most popular desktop software and operating system, infect 300,000 IIS servers, but are unable to do similar damage to the most popular web server and its operating systems?

Astute observers who examine the Netcraft web site URL will note that all 50 servers in the Netcraft uptime list are running a form of BSD, mostly BSD/OS. None of them are running Windows, and none of them are running Linux. The longest uptime in the top 50 is 1,768 consecutive days, or almost 5 years.

This appears to make BSD look superior to all operating systems in terms of reliability, but the Netcraft information is unintentionally misleading. Netcraft monitors the uptime of operating systems based on how those operating systems keep track of uptime. Linux, Solaris, HP-UX, and some versions of FreeBSD only record up to 497 days of uptime, after which their uptime counters are reset to zero and start again. So all web sites based on machines running Linux, Solaris, HP-UX and in some cases FreeBSD "appear" to reboot every 497 days even if they run for years. The Netcraft survey can never record a longer uptime than 497 days for any of these operating systems, even if they have been running for years without a reboot, which is why they never appear in the top 50.

That may explain why it is impossible for Linux, Solaris and HP-UX to show up with as impressive numbers of consecutive days of uptime as BSD -- even if these operating systems actually run for years without a reboot. But it does *not* explain why Windows is nowhere to be found in the top 50 list. Windows does not reset its uptime counter. Obviously, no Windows-based web site has been able to run long enough without rebooting to rank among the top 50 for uptime.

Given the 497-rollover quirk, it is difficult to compare Linux uptimes vs. Windows uptimes from publicly available Netcraft data. Two data points are statistically insignificant, but they are somewhat telling, given that one of them concerns the Microsoft website. As of September 2004, the average uptime of the Windows web servers

[1] See References section below for the Netcraft URLs from which this data was drawn.
[2] See References section below for the Netcraft URL for this data

that run Microsoft's own web site (www.microsoft.com) is roughly 59 days. The maximum uptime for Windows Server 2003 at the same site is 111 days, and the minimum is 5 days. Compare this to www.linux.com (a sample site that runs on Linux), which has had both an average and maximum uptime of 348 days. Since the average uptime is exactly equal to the maximum uptime, either these servers reached 497 days of uptime and reset to zero 348 days ago, or these servers were first put on-line or rebooted 348 days ago.

The bottom line is that quality, not quantity, is the determining factor when evaluating the number of successful attacks against software.

Myth: Open Source is Inherently Dangerous

The impressive uptime record for Apache also casts doubt on another popular myth: That open source code (where the blueprints for the applications are made public) is more dangerous than proprietary source code (where the blueprints are secret) because hackers can use the source code to find and exploit flaws.

The evidence begs to differ. The number of effective Windows-specific viruses, Trojans, spyware, worms and malicious programs is enormous, and the number of machines repeatedly infected by any combination of the above is so large it is difficult to quantify in realistic terms. Malicious software is so rampant that the average time it takes for an unpatched Windows XP to be compromised after connecting it directly to the Internet is 16 minutes -- less time than it takes to download and install the patches that would help protect that PC. [3]

As another example, the Apache web server is open source. Microsoft IIS is proprietary. In this case, the evidence refutes both the "most popular" myth and the "open source danger" myth. The Apache web server is by far the most popular web server. If these two myths were both true, one would expect Apache and the operating systems on which it runs to suffer far more intrusions and problems than Microsoft Windows and IIS. Yet precisely the opposite is true. Apache has a near monopoly on the best uptime statistics. Neither Microsoft Windows nor Microsoft IIS appear anywhere in the top 50 servers with the best uptime. Obviously, the fact that malicious hackers have access to the source code for Apache does not give them an advantage for creating more successful attacks against Apache than IIS.

Myths: Conclusions Based on Single Metrics

The remaining popular myths regarding the relative security of Windows vs. Linux are flawed by the fact that they are based only on a single metric -- a single aspect of measuring security. This is true whether the data comes from actual research, anecdotal information or even urban myth.

One popular claim is that, "there are more security alerts for Linux than for Windows, and therefore Linux is less secure than Windows". Another is, "The average time that elapses between discovery of a flaw and when a patch for that flaw is released is greater for Linux than it is for Windows, and therefore Linux is less secure than Windows."

The latter is the most mysterious of all. It is an imponderable mystery how anyone can reach the conclusion that Microsoft's average response time between discovery of a flaw and releasing the fix for that flaw is superior to that of *any* competing operating system, let alone superior to Linux. Microsoft took seven months to fix one of its most serious security vulnerabilities (Microsoft Security Bulletin MS04-007 ASN.1 Vulnerability, eEye Digital Security publishes the delay in advisory AD20040210), and there are flaws Microsoft has openly stated it will *never* repair. The Microsoft Security Bulletin MS03-010 about the Denial Of Service vulnerability in Windows NT says this will never be repaired. More recently, Microsoft stated that it would not repair Internet Explorer vulnerabilities for any operating systems older than Windows XP. Statistically speaking, seven months between discovery and fix might not have an overly dramatic effect on the average response time if you can find enough samples of excellent response times to offset anomalies like this, assuming they are anomalies. But it only takes one case of "never" to upset the statistical average beyond recovery.

This unsolvable mystery aside, consider whether it is meaningful to suggest that Linux is a greater security risk than Windows because the average time between the discovery of vulnerability and the release of a patch is greater with Linux than with Windows. Ask yourself this question: If you experienced a heart attack at this very moment, to which hospital emergency room would you rather be taken? Would you want to go to the one with the best average response time from check-in to medical treatment? Or would you rather be taken to an emergency room with a poor record for average response time, but where the patients with the most severe medical problems always get immediate attention?

One would obviously choose the latter, but not necessarily because the above information proves it is the better emergency room. The latter choice is preferable because it includes two metrics, one of which is more important to you at that precise moment. It is safe to assume that most people would avoid a hospital if they also knew they were likely to die of a heart attack waiting for a doctor to finish setting someone's fractured pinky, no matter how impressive the average response time for every medical emergency may be. The problem is that the above example doesn't give you sufficient information to make the best decision. It doesn't tell you how well the

[3] Unpatched PC "Survival Time" Just 16 Minutes, by Greek Keiser, TechWeb News. See references section below for URL.

hospital with the best average response time prioritizes its cases. You would also benefit from knowing things like the mortality rate of emergency cases, the average skill of the resident physicians, and so on.

Obviously, the only way to produce a useful recommendation is to gather as many important metrics as possible about local emergency rooms, and then balance these metrics intelligently. It would be inexcusably irresponsible to recommend an emergency room for a heart attack based only on a single metric such as the average response time for all medical emergencies, especially when the other important information that would lead to a more ideal choice is readily available.

It is equally irrational and irresponsible to make a recommendation or a serious business decision based solely on a single metric such as the average elapsed time between a flaw's detection and fix for a given operating system, or the number of security alerts for any given product.

Any single metric is misleading in terms of importance. Let's consider the statement that there are more alerts for Linux software than Windows. This statistic is meaningless because it leaves the most important questions unanswered. Of all the security alerts, how many of the reported flaws represent a tangible risk? How severe are those risks? How likely are they to expose your systems to serious damage? These questions are important. Which is preferable: An operating system with 100 flaws that expose your systems to little or no damage and cannot be exploited by anyone except local users with a valid login account and physical access to your machine? Or would you prefer an operating system with 1 critical flaw that allows any malicious hacker on the Internet to wipe out all of the information on your server? Clearly, the number of alerts alone is not a meaningful metric for the security of one operating system over another.

Windows vs. Linux Design

It is possible that email and browser-based viruses, Trojans and worms are the source of the myth that Windows is attacked more often than Linux. Clearly there are more desktop installations of Windows than Linux. It is certainly possible, if not probable, that Windows *desktop* software is attacked more often because Windows dominates the desktop. But this leaves an important question unanswered. Do the attacks so often succeed on Windows because the attacks are so numerous, or because there are inherent design flaws and poor design decisions in Windows?

Many, if not most of the viruses, Trojans, worms and other malware that infect Windows machines do so through vulnerabilities in Microsoft Outlook and Internet Explorer. To put the question another way, given the same type of desktop software on Linux (the most often used web browsers, email, word processors, etc.), Are there as many security vulnerabilities on Linux as Windows?

Windows Design

Viruses, Trojans and other malware make it onto Windows desktops for a number of reasons familiar to Windows and foreign to Linux:

1. Windows has only recently evolved from a single-user design to a multi-user model
2. Windows is monolithic, not modular, by design
3. Windows depends too heavily on an RPC model
4. Windows focuses on its familiar graphical desktop interface

WINDOWS HAS ONLY RECENTLY EVOLVED FROM A SINGLE-USER DESIGN TO A MULTI-USER MODEL

Critics of Linux are fond of saying that Linux is “old” technology. Ironically, one of the biggest problems with Windows is that it hasn’t been able to escape its “old” legacy single-user design. Windows has long been hampered by its origin as a single-user system. Windows was originally designed to allow both users and applications free access to the entire system, which means anyone could tamper with a critical system program or file. It also means viruses, Trojans and other malware could tamper with any critical system program or file, because Windows did not isolate users or applications from these sensitive areas of the operating system.

Windows XP was the first version of Windows to reflect a serious effort to isolate users from the system, so that users each have their own private files and limited system privileges. This caused many legacy Windows applications to fail, because they were used to being able to access and modify programs and files that only an administrator should be able to access. That’s why Windows XP includes a compatibility mode – a mode that allows programs to operate as if they were running in the original insecure single-user design. This is also why each new version of Windows threatens to break applications that ran on previous versions. As Microsoft is forced to hack Windows into behaving more like a multi-user system, the new restrictions break applications that are used to working without those restraints.

Windows XP represented progress, but even Windows XP could not be justifiably referred to as a true multi-user system. For example, Windows XP supports what Microsoft calls “Fast User Switching”, which means that two or more people can log into a Windows XP system on a single PC at the same time. Here’s the catch. This is only possible if *and only if* the PC is *not* set up to be part of a Windows network domain. That’s because Microsoft networking was designed under the assumption that people who log into a network will do so from their own PC. Microsoft was either unable or unwilling to make the necessary changes to the operating system and network design to accommodate this scenario for Windows XP.

Windows Server 2003 makes some more progress toward true multi-user capabilities, but even Windows Server 2003 hasn’t escaped all of the leftover single-user security holes. That’s why Windows Server 2003 has to turn off many browser capabilities (such as ActiveX, scripting, etc.) by default. If Microsoft had redesigned these features to work in a safe, isolated manner within a true multi-user environment, these features would not present the severe risks that continue to plague Windows.

WINDOWS IS MONOLITHIC BY DESIGN, NOT MODULAR

A monolithic system is one where most features are integrated into a single unit. The antithesis of a monolithic system is one where features are separated out into distinct layers, each layer having limited access to the other layers.

While some of the shortcomings of Windows are due to its ties to its original single-user design, other shortcomings are the direct result of deliberate design decisions, such as its monolithic design (integrating too many features into the core of the operating system). Microsoft made the Netscape browser irrelevant by integrating Internet Explorer so tightly into its operating system that it is almost impossible not to use IE. Like it or not, you invoke Internet Explorer when you use the Windows help system, Outlook, and many other Microsoft and third-party applications. Granted, it is in the best business interest of Microsoft to make it difficult to use anything but Internet Explorer. Microsoft successfully makes competing products irrelevant by integrating more and more of the services they provide into its operating system. But this approach creates a monster of inextricably interdependent services (which is, by definition, a monolithic system).

Interdependencies like these have two unfortunate cascading side effects. First, in a monolithic system, every flaw in a piece of that system is exposed through all of the services and applications that depend on that piece of the system. When Microsoft integrated Internet Explorer into the operating system, Microsoft created a system where any flaw in Internet Explorer could expose your Windows desktop to risks that go far beyond what you do with your browser. A single flaw in Internet Explorer is therefore exposed in countless other applications, many of which may use Internet Explorer in a way that is not obvious to the user, giving the user a false sense of security.

This architectural model has far deeper implications that most people may find difficult to grasp, one being that a monolithic system tends to make security vulnerabilities more critical than they need to be.

Perhaps an admittedly oversimplified visual analogy may help. Think of an ideally designed operating system as being comprised of three spheres, one in the center, another larger sphere that envelops the first, and a third sphere that envelope the inner two. The end-user only sees the outermost sphere. This is the layer where you run applications, like word processors. The word processors make use of commonly needed features provided by the second sphere, such as the ability to render graphical images or format text. This second sphere (usually referred to as “userland” by technical geeks) cannot access vulnerable parts of the system directly. It must request permission from the innermost sphere in order to do its work. The innermost sphere has the most important job, and therefore has the most direct access to all the vulnerable parts of your system. It controls your computer’s disks, memory, and everything else. This sphere is called the “kernel”., and is the heart of the operating system.

In the above architecture, a flaw in the graphics rendering routines cannot do global damage to your computer because the rendering functions do not have direct access to the most vulnerable system areas. So even if you can convince a user to load an image with an embedded virus into the word processor, the virus cannot damage anything except the user’s own files, because the graphical rendering feature lies outside the innermost sphere, and does not have permission to access any of the critical system areas.

The problem with Windows is that does not follow sensible design practices in separating out its features into the appropriate layers represented by the spheres described above. Windows puts far too many features into the core, central sphere, where the most damage can be done. For example, if one integrates the graphics rendering features into the innermost sphere (the kernel), it gives the graphical rendering feature the ability to damage the entire system. Thus, when someone finds a flaw in a graphics-rendering scheme, the overly integrated architecture of Windows makes it easy to exploit that flaw to take complete control of the system, or destroy the entire system.

Finally, a monolithic system is unstable by nature. When you design a system that has too many interdependencies, you introduce numerous risks when you change one piece of the system. One change may (and usually does) have a cascading effect on all of the services and applications that depend on that piece of the system. This is why Windows users cringe at the thought of applying patches and updates. Updates that fix one part of Windows often break other existing services and applications. Case and point: The Windows XP service pack 2 already has a growing history of causing existing third-party applications to fail. This is the natural consequence of a monolithic system – any changes to one part of the machine affect the whole machine, and all of the applications that depend on the machine.

WINDOWS DEPENDS TOO HEAVILY ON THE RPC MODEL

RPC stands for Remote Procedure Call. Simply put, an RPC is what happens when one program sends a message over a network to tell another program to do something. For example, one program can use an RPC to tell another program to calculate the average cost of tea in China and return the answer. The reason it’s called a *remote* procedure call is because it doesn't matter if the other program is running on the same machine, another machine in the next cube, or somewhere on the Internet.

RPCs are potential security risks because they are designed to let other computers somewhere on a network to tell your computer what to do. Whenever someone discovers a flaw in an RPC-enabled program, there is the potential for someone with a network-connected computer to exploit the flaw in order to tell your computer what to do. Unfortunately, Windows users cannot disable RPC because Windows depends upon it, even if your computer is not connected to a network. Many Windows services are simply designed that way. In some cases, you can block an RPC port at your firewall, but Windows often depends so heavily on RPC mechanisms for basic functions that this is not always possible. Ironically, some of the most serious vulnerabilities in Windows Server 2003 (see table in section below) are due to flaws in the Windows RPC functions themselves, rather than the applications that use them. The most common way to exploit an RPC-related vulnerability is to attack the service that uses RPC, not RPC itself.

It is important to note that RPCs are not always necessary, which makes it all the more mysterious as to why Microsoft indiscriminately relies on them. Assume for a moment that you create a web site using two servers. One server is a dedicated database server, and the other server is a dedicated web server. In this case, it is necessary for the database server to use RPCs, because the web server is on a separate machine and must be able to access the database server over the network connection. (Even in this case, one should configure the database server to "listen" only to the web server, and no other machine.) If you run both the database server and web server on the same machine, however, it is not only unnecessary for the database server to use RPCs, it is unwise to do so. The web server should be able to access the database server directly, because the two are running on the same machine. There is no technical or logical reason to expose the database server to the network, because it presents an unnecessary security risk.

We raise the issue of database servers because the Slammer worm, one of the most profoundly dangerous worms ever to hit the Internet, exploited one of the most inappropriate uses of RPC-like network communications ever implemented by Microsoft. Slammer infected so many systems so quickly that it practically brought the Internet to a standstill.

The Slammer worm caused havoc by exploiting two flaws in Microsoft SQL Server, a client/server SQL database server. One flaw was a most improbable feature of Microsoft SQL Server – one that allows you to run more than one instance of the database server at a time on a single machine. Here is why it is improbable. If you're not familiar with database servers, picture it this way. Under normal conditions, it makes no sense to run multiple instances of a database server on a single machine, because one instance is all that is needed, even if many different applications use it. One would be as likely to want to run two copies of Windows XP on a single machine at the same time as want to run multiple database servers on a single machine at the same time. One rarely runs multiple instances of a database server on purpose, except in high-end applications or for testing and development. [4]

The easy way to allow multiple instances of SQL Server to run simultaneously without interfering with one another is to create an RPC mechanism that sorts out requests for data, so that a fax application queries its own copy

[4] We suspect we know why Microsoft chose to implement this as the default behavior of SQL Server. Many third-party applications use the SQL Server engine by default. If everyone who wrote applications for SQL Server assumed that there would be a single instance of SQL Server running on the machine, Microsoft would have to provide an easy way for the installation programs to detect that SQL Server was already installed and running, and then provide an easy way to install, integrate and administer the applications' specific requirements for its own database and tables running on the existing server. This is the elegant solution, and it uses up a minimum of resources because only one instance of SQL Server is ever needed. But this approach would require a good deal of extra work on the part of Microsoft or on the part of the third-party developers. It was much easier to design a way to allow third party applications to avoid bothering with the issue of whether or not SQL Server is already installed. Given the design Microsoft implemented, any third party can simply install its own copy of SQL Server without worrying about whether or not SQL Server already exists on the target machine, what version of SQL Server is already installed, or how the SQL Server is already configured. In short, rather than do things right, and in an effort to entice third parties to use SQL Server, Microsoft took the lazy way out and designed a system where any application could install its own private copy of SQL Server without its operation interfering with the other copies of SQL Server running on the same system. This led to the desire to run several instances of SQL Server with RPC enabled, which should actually have a very narrow audience. This lazy approach had terribly unfortunate consequences. If Microsoft had designed SQL Server to run as a single instance without network connections by default, the Slammer worm would not have been able to find enough machines running SQL Server to do any significant damage.

of SQL Server, and a time-billing application queries yet another copy of SQL Server. To complicate matters, Microsoft development tools encourage the same monolithic approach Microsoft uses, so a broad range of applications – time-billing software, fax software, project management – almost 200 applications, many of them desktop applications, use the unnecessarily vulnerable SQL Server engine. As a result, hundreds of thousands, if not millions, of people use *desktop* applications that depend on the SQL Server engine with multiple network services enabled, many of which are exposed to the Internet. One could hardly concoct a better recipe for disaster.

As a result, Slammer found countless machines to attack because these features are enabled by default on every SQL Server engine. While SQL Server is not yet integrated into Windows, its ubiquity across applications from fax software to time billing software made it effectively a part of a larger monolithic system, thus opening the way to an attack path that is symptomatic of a monolithic system. Unfortunately, SQL Server is likely to be tightly integrated into the upcoming new Windows File System WinFS originally slated for Longhorn. If anyone thinks integrating SQL Server into the operating system is a good idea, they should consider what happened with the Slammer worm.

WINDOWS FOCUSES ON ITS FAMILIAR GRAPHICAL DESKTOP INTERFACE

Microsoft considers its familiar Windows interface as the number one benefit for using Windows Server 2003.[5] To quote from the Microsoft web site, *"With its familiar Windows interface, Windows Server 2003 is easy to use. New streamlined wizards simplify the setup of specific server roles and routine server management tasks…"*

By advocating this type of usage, Microsoft invites administrators to work with Windows Server 2003 at the server itself, logged in with Administrator privileges. This makes the Windows administrator most vulnerable to security flaws, because using vulnerable programs such as Internet Explorer expose the server to security risks.

[5] See References section for URL to the "Top 10 Benefits of Windows Server 2003" page at the Microsoft web site.

Linux Design

According to the Summer 2004 Evans Data Linux Developers Survey, 93% of Linux developers have experienced two or fewer incidents where a Linux machine was compromised. Eighty-seven percent had experienced only one such incident, and 78% have never had a cracker break into a Linux machine. In the few cases where intruders succeeded, the primary cause was inadequately configured security settings.

More relevant to this discussion, however, is the fact that 92% of those surveyed have never experienced a virus, Trojan, or other malware infection on Linux.

Viruses, Trojans and other malware rarely, if ever, manage to infect Linux systems, in part because:

1. Linux is based on a long history of well fleshed-out multi-user design
2. Linux is mostly modular by design
3. Linux does not depend upon RPC to function, and services are usually configured not to use RPC by default
4. Linux servers are ideal for headless non-local administration

Keep in mind when reading the summaries below that there are variations in the default configurations of the different distributions of Linux, so what may be true of Red Hat Linux may not be true of Debian and there may be even more differences in SuSE. For the most part, all the major Linux distributions tend to follow sane guidelines in the default configurations.

LINUX IS BASED ON A LONG HISTORY OF WELL FLESHED-OUT MULTI-USER DESIGN

Linux does not have a history of being a single-user system. Therefore it has been designed from the ground-up to isolate users from applications, files and directories that affect the entire operating system. Each user is given a user directory where all of the user's data files and configuration files are stored. When a user runs an application, such as a word processor, that word processor runs with the restricted privileges of the user. It can only write to the user's own home directory. It cannot write to a system file or even to another user's directory unless the administrator explicitly gives the user permission to do so.

Even more important, Linux provides almost all capabilities, such as the rendering of JPEG images, as modular libraries. As a result, when a word processor renders JPEG images, the JPEG rendering functions will run with the same restricted privileges as the word processor itself. If there is a flaw in the JPEG rendering routines, a malicious hacker can only exploit this flaw to gain the same privileges as the user, thus limiting the potential damage. This is the benefit of a modular system, and it follows more closely the spherical analogy of an ideally designed operating system (see the section *Windows is Monolithic by Design, not Modular*).

Given the default restrictions in the modular nature of Linux; it is nearly impossible to send an email to a Linux user that will infect the entire machine with a virus. It doesn't matter how poorly the email client is designed or how badly it may behave – it only has the privileges to infect or damage the user's own files. Linux browsers do not support inherently insecure objects such as ActiveX controls, but even if they did, a malicious ActiveX control would only run with the privileges of the user who is running the browser. Once again, the most damage it could do is infect or delete the user's own files.

Even services, such as web servers, typically run as users with restricted privileges. For example, Debian GNU/Linux runs the Apache server as the user "www-data", who belongs to a group with the same name, "www-data". If a malicious hacker manages to gain complete control over the Apache web server on a Debian system, that hacker can only affect files owned by the user "www-data", such as web pages. In turn, the MySQL SQL database server often used in conjunction with Apache, runs with the privileges of the user "mysql". So even if Apache and MySQL are used together to serve web pages, a malicious hacker who gains control of Apache does not have the privileges to exploit the Apache hole in order to gain control of the database server, because the database server is "owned" by another user.

In addition, users associated with services such as Apache, MySQL, etc., are often set up with user accounts that have no access to a command line. So if a malicious hacker somehow breaks into the MySQL user account, that hacker cannot exploit that vulnerability to issue arbitrary commands to the Linux server, because that account has no ability to issue commands.

In sharp contrast, Windows was originally designed to allow all users and applications to have administrator access to every file on the system. Windows has only gradually been re-worked to isolate users and what they do from the rest of the system. Windows Server 2003 is close to achieving this goal, but the methodology

Microsoft has employed to create this barrier between user and system is still largely composed of constantly changing hacks to the existing design, rather than a fundamental redesign with multi-user capability and security as the foundational concept behind the system.

LINUX IS MODULAR BY DESIGN, NOT MONOLITHIC

Linux is for the most part a modularly designed operating system, from the kernel (the core "brains" of Linux) to the applications. Almost nothing in Linux is inextricably intertwined with anything else. There is no single browser engine used by help systems or email programs. Indeed, it is easy to configure most email programs to use a built-in browser engine to render HTML messages, or launch any browser you wish to view HTML documents or jump to links included in an email message. Therefore a flaw in one browser engine does not necessarily present a danger to any other application on the system, because few if any other applications besides the browser itself must depend on a single browser engine.

Not everything in Linux is modular. The two most popular graphical desktops, KDE and GNOME, are somewhat monolithic by design; at least enough so that an update to one part of GNOME or KDE can potentially break other parts of GNOME or KDE. Neither GNOME nor KDE are so monolithic, however, as to require you to use GNOME or KDE-specific applications. You can run GNOME applications or any other applications under KDE, and you can run KDE or any other applications under GNOME.

The Linux kernel supports modular drivers, but it is essentially a monolithic kernel where services in the kernel are interdependent. Any adverse impact of this monolithic approach is minimized by the fact that the Linux kernel is designed to be as minimal a part of the system as possible. Linux follows the following philosophy almost to a point of fanaticism: "Whenever a task can be done outside the kernel, it must be done outside the kernel." This means that almost every useful feature in Linux ("useful" as perceived by an end user) is a feature that does not have access to the vulnerable parts of a Linux system.

In contrast, bugs in graphics card drivers are a common cause of the Windows blue-screen-of-death. That's because Windows integrates graphics into the kernel, where a bug can cause a system failure. With only a few proprietary exceptions (such as the third-party NVidia graphics driver), Linux forces all graphics drivers to run outside the kernel. A bug in a graphics driver may cause the graphical desktop to fail, but not cause the entire system to fail. If this happens, one simply restarts the graphical desktop. One does not need to reboot the computer.

LINUX IS NOT CONSTRAINED BY AN RPC MODEL

As stated above in the section on Windows, RPC stands for Remote Procedure Call. Simply put, an RPC allows one program to tell another program to do something, even if that other program resides on another computer. For example, one program can use an RPC to tell another program to calculate the average cost of tea in China and return the answer. The reason it's called a *remote* procedure call is because it doesn't matter if the other program is running on the same machine, another machine in the next cube, or somewhere on the Internet.

Most Linux distributions install programs with network access turned off by default. For example, the MySQL SQL database server is usually installed such that it does not listen to the network for instructions. If you build a web site using Apache and MySQL on the same server machine, then Apache will interact with MySQL without MySQL having to listen to the network. Contrast this to SQL Server, which listens to the network whether or not it is necessary to do so. If you want MySQL to listen to the network, you must turn on that feature manually, and then explicitly define the users and machines allowed to access MySQL.

Even when Linux applications use the network by default, they are most often configured to respond only to the local machine and ignore any requests from other machines on the network.

Unlike Windows Server 2003, you can disable virtually all network-related RPC services on a Linux machine and still have a perfectly functional desktop.

LINUX SERVERS ARE IDEAL FOR HEADLESS NON-LOCAL ADMINISTRATION

A Linux server can be installed, and often should be installed as a "headless" system (no monitor is connected) and administered remotely. This is often the ideal type of installation for servers because a remotely administered server is not exposed to the same risks as a locally administered server.

For example, you can log into your desktop computer as a normal user with restricted privileges and administer the Linux server through a browser-based administration interface. Even the most critical browser-based security vulnerability affects only your local user-level account on the desktop, leaving the server untouched by the security hole.

This may be one of the most important differentiating factors between Linux and Windows, because it virtually negates most of the critical security vulnerabilities that are common to both Linux and Windows systems, such as the vulnerabilities of the Mozilla browser vs. the Internet Explorer browser.

Realistic Security and Severity Metrics

One needs to examine many metrics in order to evaluate properly the risks involved in adopting one operating system over another for any given task. Metrics are sometimes cumulative; at other times they offset each other.

There are three very important metrics, represented as risk factors, which have a profound effect on one another. The combination of the three can have a dramatic impact on the overall severity of a security flaw. These three risk factors are *damage potential*, *exploitation potential,* and *exposure potential.*

Elements of an Overall Severity Metric

Damage potential of any given discovered security vulnerability is a measurement of the potential harm done. A vulnerability that exposes all your administrator passwords has a high damage potential. A flaw that makes your screen flicker would have a much lower damage potential, raised only if that particular damage is difficult to repair.

Exploitation potential describes how easy or difficult it is to exploit the vulnerability. Does it require expert programming skills to exploit this flaw, or can almost anyone with rudimentary computer experience use it for mischief?

Exposure potential describes the amount of access necessary to exploit a given vulnerability. If any hotshot hacker (commonly referred to as "script kiddies") on the Internet can exploit a flaw on a server you have protected by a firewall, that flaw has a very high exposure potential. If it is only possible to exploit the flaw if you are an employee within the company with a valid login ID, using a computer inside the company building, the exposure potential of that flaw is significantly less severe.

OVERALL SEVERITY METRIC AND INTERACTION BETWEEN THE THREE KEY METRICS

One or more of these risk factors can have a profound affect on the overall severity of a security hole. Assume for a moment that you are the CIO for a business based on a web eCommerce site. Your security analyst informs you that someone has found a flaw in the operating system your servers are running. A malicious hacker could exploit this flaw to erase every disk on every server on which the company depends.

The *damage potential* of this flaw is catastrophic.

Worse, he adds that it is trivially easy from a technical perspective to exploit this flaw. The *exploitation potential* is critical.

Time to press the panic button, right? Now suppose he then adds this vital bit of information. Someone can only exploit this flaw with a key to the server room, because this particular security vulnerability requires physical access to the machines. This one key metric, if you'll pardon the pun, makes a dramatic difference in the overall severity of the risk associated with this particular flaw. The extremely low *exposure potential* shifts the needle on the severity meter from "panic" to "imminently manageable".

Conversely, another security vulnerability might be exposed to every script kiddy on the Internet, but still be considered of negligible severity because the *damage potential* for this flaw is inconsequential.

Perhaps you can begin to appreciate why it is misleading, if not outright irresponsible to measure security based on a single metric like the number of security alerts. At the very least, one must also consider these three risk factors. Would you rather rely on an operating system with a history of hundreds of flaws of negligible severity, or one with a history of a dozens of flaws with catastrophic severity? Unless you factor the overall severity of the flaws into the evaluation, the number of flaws is irrelevant at best, misleading at worst.

THE EXCEPTION TO THE RULE

The Overall Severity Metric has three aforementioned "main" ingredients. We've demonstrated how a low *damage potential* or a low *exposure potential* can effectively negate the other high risk factors. The *exploitation potential* is an exception to this rule. A flaw that requires expert programming skills to exploit does far less to offset a high *damage potential* or a high *exposure potential.*

The reason for this is simple. If one must break into a computer room in order to exploit a flaw, that problem is not only difficult to overcome, any attempt to break into the computer room increases the risk for the intruder to *get caught*. That is also why a flaw that can only be exploited by an employee within the company who must log in to a local computer with his valid login ID is less severe than a flaw that can be exploited by any script kiddy on the Internet. The employee is far more likely to get caught.

On the other hand, anonymous malicious hackers with only mediocre programming skills can spend weeks or months developing a program to exploit a security hole with little or no risk of getting caught. The only significant challenge presented to such an intruder is how to activate the malicious program without having its origin traced back to its creator.

One look at the current state of malicious software should make this exception self-evident. Not many people blast their way into a computer room with a bazooka in order to crack into the servers. But there are countless Trojans, worms, and viruses that are still infecting systems today, in part because programmers, talented or not, were willing to tackle the technical challenge of writing malicious code or re-writing the malicious code of others. Technical difficulty obviously does not necessarily offset an otherwise high-risk flaw.

APPLYING THE OVERALL SEVERITY METRIC

Once you can evaluate the overall severity of any given flaw, you can begin to add meaning to metrics such as “how many security alerts does Windows have vs. Linux”, or “how long does one have to wait for a fix after a flaw is discovered when using Windows vs. Linux”.

Suppose one operating system has far more security alerts than another. The only reason that metric may have meaning is if it also has more security alerts *that point to flaws with a high overall severity level.* It is one thing to be plagued on a regular basis by a myriad of minor low-risk annoyances, quite another to be plagued on a regular basis by only a few flaws that put your entire company at risk.

Suppose one operating system has a better record for time to delivery of a fix once a flaw is discovered. Once again, the only reason this metric may have meaning is *if the delays are related to flaws with a high overall severity level.* It is one thing to wait months for a fix to an exploit that would cause little or no damage on a few computers. It is quite another to wait months for a fix for a flaw that puts your entire company at risk.

Means Of Evaluating Metrics

EXPOSURE POTENTIAL

This metric takes into account the measures one must take to access a machine in order to exploit security vulnerabilities. This typically falls into one of the following categories. The actual order of some of these categories can vary in practice, but this should prove to be a useful guideline. It should also be noted that there are several unusual complexities not listed here. For example, a patched flaw in Windows Server 2003 was not itself a serious exposure, but it allowed a malicious hacker to open the system to serious exposure. In short, it was a single step in a chain of exposure vulnerabilities. Given that these are roughly defined categories, they are listed in terms of severity, ordered from least to greatest.

1. You need physical access to the machine, but not a valid user login account.
2. You need physical access to the machine and must have a valid user login account.
3. You need a valid user login account, but do not need physical access to the target machine. Local network access (from inside the company network) is sufficient.
4. You need a valid user login account, but do not need physical access to the target machine. The target machine is accessible via the Internet from a remote location.
5. You can exploit a flaw remotely from the Internet without a valid login account for the target machine, but you cannot reach the flaw directly. Another barrier is in place, such as a router or firewall. This category is difficult to place in the correct order of severity, since a well-configured firewall may provide 100% protection, but not always. A poorly configured firewall may not present a barrier at all.
6. You can exploit a flaw remotely from the Internet without a valid login account for the target machine, but you cannot reach the flaw directly. Another less intrusive barrier is in place. This barrier may be another program (for example, the flaw is in Microsoft SQL Server, but must be exploited by embedding an ActiveX control or Javascript within a web page accessed by Microsoft Internet Information Server. In some cases, you must entice the user into an action in order to gain indirect access. For example, you must send a user an email that directs them to a web page that includes the malicious control or code. To use another common practice, the user is enticed to open an attachment to an email. The severity of this category varies depending on how cleverly the enticement is disguised as an innocent action.
7. You can exploit a flaw remotely from the Internet without a valid login account for the target machine, but you cannot reach the flaw directly. Nevertheless, the flaw is exploited indirectly but automatically. For example, a flaw in the Windows operating system is exploited immediately and automatically as soon as a user opens an email message in Outlook.
8. You can exploit a flaw remotely from the Internet simply by sending information directly to the target machine via the network. For example, one might be able to exploit a Denial Of Service (DoS) vulnerability simply by sending special network packets to a target web site, rendering that web site unavailable to other Internet users.

EXPLOITATION POTENTIAL

This metric takes into account the technical difficulty involved in exploiting a security flaw. This typically falls into one of the following categories, in terms of severity, ordered from least to greatest (the actual order of some of these categories can vary in practice, but this should prove to be a useful guideline):

1. The flaw exists but it has not yet been discovered. This flaw either requires infinite knowledge or a lucky accident to exploit.
2. The flaw requires expert programming skills and profound knowledge of the operating system, but its existence is not known well enough that many such attackers would be likely to exploit it.
3. The flaw is known by and requires attackers with expert programming skills and profound understanding of how the target software and operating system works in order to exploit.
4. The flaw requires expert programming skills, but someone has already created a virus, Trojan, or worm as a foundation. The programmer must only modify the code in order to exploit a new flaw, or modify the code in order to make the virus more dangerous.

5. The flaw required expert programming skills to create, but the code is available and it requires only mediocre programming skills to improve or modify the code in order to exploit the existing flaw, or future flaws.
6. The flaw requires only mediocre or novice programming skills, or rudimentary computer knowledge to exploit.
7. It is irrelevant how difficult it is to exploit the flaw, because someone has done the hard work of solving the means of exploiting the flaw, and made a intrusion kit publicly available for use by novices.
8. Anyone can exploit the flaw simply by typing simple text at a command line or pointing a browser to a URL.

DAMAGE POTENTIAL

This metric is the most difficult to quantify. It requires at least two separate sets of categories. First, it takes into account how much damage potential a flaw presents to an application or the computer system. Second, the damage potential must be measured in terms of "what it means" to the company affected. For example, there is a single metric where a flaw allows an attacker to read unpublished web pages. That flaw is relatively minor if no sensitive information is present in the system. However, if an unpublished web page contains sensitive information such as credit card numbers, the overall damage potential is quite high even though the technical damage potential is minimal. Here are the most important factors in estimating technical damage potential for any given flaw, in order of severity from least to worst:

1. The flaw affects only the performance of another computer, but not significantly enough to make the computer stop responding.
2. The flaw only affects the attacker's own programs or files, but not the files or programs of other users.
3. The flaw exposes the information in co-worker's files, but not information from the administrator account or information in any system files.
4. The flaw allows an attacker to examine, change or delete a user's files. It does not allow the attacker to examine, change or delete administrator or system files.
5. The flaw allows an attacker to view sensitive information, whether by examining network traffic or by getting read-only access to administrator or system files.
6. The flaw allows an attacker to gain some but not all administrator-level privileges, perhaps within a restricted environment.
7. The flaw allows an attacker to either crash the system or otherwise cause the system to stop responding to normal requests. This is typically a Denial Of Service (DoS) attack. However, the attacker cannot actually gain control of the computer aside from stopping it from responding.
8. The flaw allows an attacker to change or delete all privileged files and information. The attacker can gain complete control of the target system and do virtually any amount of damage that a fully authorized system administrator can do.

OVERALL SEVERITY RISK

Given the above three factors, the overall severity risks range from minimal to catastrophic. It would be impossible to consider all the permutations, but a few examples may prove useful. These examples are based on the damage potential categories, combined with assorted selections from exposure and exploitation potential.

1. If an anonymous hacker on the Internet can degrade your company's system performance, this can range from a minor annoyance to a devastating financial impact, depending on how critical system performance may be to the mission of your company.
2. Attacking your own account is silly, but self-destructive behavior can cause needless restoration work by the IT department.
3. The potential severity of viewing another user's files is minimal if you can only view the files of a co-worker in the same building, even if this flaw is trivially easy to exploit. The severity is increased if the co-worker's files contain sensitive information, and decreased the more likely the attacker may be to get caught. On the other hand, if any anonymous malicious hacker on the Internet (high exposure

potential) can view sensitive files of a user within your company, the overall severity is dramatically more serious.

4. Again, if the flaw allows an attacker to change or delete the files of a co-worker in the same building, the severity is minimized by how well the company performs backups, and how easily the attacker will get caught. If the attacker can change files on a remote computer's user account, the severity varies with the importance of that user account and the service it provides. For example, the severity may range from the embarrassment of having your web pages defaced to having your web pages deleted entirely.

Additional Considerations

APPLICATION IMBALANCE

One factor that is often overlooked in the grand debate about the superiority of one operating system over another hinges on the fact that security vulnerabilities almost always revolve around applications. This presents a problem when comparing Windows to Linux, because the two are not at all equal with respect to application portability and availability.

On the one hand, most of the popular Microsoft Windows applications are Microsoft applications, and they *only* run on Windows. When a flaw is found in Microsoft Exchange, one can be reasonably certain that this problem only affects Windows customers. Microsoft Exchange does not run on Linux, Solaris, or anything else but Windows.

The Apache web server, on the other hand, may be most often associated with Linux, UNIX or other UNIX-like systems, but Apache does run on Windows, as well. So when one compares the overall security of Windows vs. Linux, is a flaw in Apache a blemish on Linux only? Or does it reflect negatively on both Linux and Windows?

To complicate matters, there are several cases where a flaw in Apache poses little or no danger on Linux, but is a serious vulnerability on Windows. The reverse is rarely, if ever, the case. Should the overall security ranking of Windows suffer because it is more adversely affected than Linux when using software that is most commonly associated with Linux?

One is obligated to question if any of these factors have been considered when comparing the security of Windows to Linux.

SETUP AND ADMINISTRATION

Finally, the difference between the Linux philosophy to server setup and administration vs. the Windows philosophy to setup and administration is, as stated earlier, perhaps the most critical differentiating factor between the two operating systems.

Windows encourages you to use the familiar interface, which means administering Windows Server 2003 at the server itself. Linux does not rely on or encourage local use of a graphical interface, in part because it is an unnecessary waste of resources to run a graphical desktop at the server, and in part because it increases security risks at the server. For example, any server that encourages you to use the graphical interface at the server machine also invites you to perform similar operations, such as use the browser at the server. This exposes that server to any browser security holes. Any server that encourages you to administer it remotely removes this risk. If you administer a Linux server remotely from a desktop user account, a browser flaw exposes only the remote desktop user account to security holes, not the server. This is why a browser security hole in Windows Server 2003 is potentially more serious than a browser security hole in Red Hat Enterprise Server AS.

A Comparison of 40 Recent Security Patches

The following sections document the 40 most recent patches to security vulnerabilities in Windows Server 2003 (arguably the most secure version of Windows) and Linux Red Hat Enterprise AS v.3 (arguably the competitive equivalent of Windows Server 2003). The data for the Windows Server 2003 patches and vulnerabilities was taken directly from the Microsoft web site, and the data for Red Hat Enterprise AS v.3 was taken from the Red Hat web site.

Windows Server 2003 has experienced the most severe security holes. Microsoft's own classification of the flaws shows that 38% of the patched programs are rated as Critical. If we apply the metrics outlined in the previous sections, we would have to raise that to between 40-50%. Many of the flaws that are assigned the Critical rank in Windows XP or other versions are downplayed for Windows Server 2003 simply because the default settings for Internet Explorer and Outlook are now severely restrictive – so restrictive that these programs are practically unusable without reversing at least some of these defaults.

In sharp contrast, of the 40 vulnerabilities listed by Red Hat, only 4 are rated as Critical by our metrics (Red Hat does not list a severity rank for its alerts). That means 10% of the most recent 40 updates are of Critical severity. This score is actually generous to Microsoft, since two of the flaws could easily be argued to rank lower than Critical, thus also lowering the percentage of Critical flaws to 5%.

Patches and Vulnerabilities Affecting Microsoft Windows Server 2003

The following table contains information about the vulnerabilities from the 40 most recent security patches made available by Microsoft. [6]

Microsoft marks fifteen of the 40 vulnerabilities as Critical. That means by Microsoft's own subjective analysis, 38% of the most recent problems reported and patched are of Critical severity, the highest rating possible.

There are two serious problems with the way Microsoft has rated the severity of its flaws, however:

1. Microsoft often ranks a security flaw as *Critical* for all Windows operating systems *except* Windows Server 2003, in which case it is ranked at the lower value, *Important*. The reason given for this difference is that Windows Server 2003 has different default settings than other versions of Windows. Here is Microsoft's own description of the different settings:[7]

- *Security level for the Internet zone is set to High. This setting disables scripts, ActiveX controls, Microsoft Java Virtual Machine (MSJVM), HTML content, and file downloads.*
- *Automatic detection of intranet sites is disabled. This setting assigns all intranet Web sites and all Universal Naming Convention (UNC) paths that are not explicitly listed in the Local intranet zone to the Internet zone.*
- *Install On Demand and non-Microsoft browser extensions are disabled. This setting prevents Web pages from automatically installing components and prevents non-Microsoft extensions from running.*
- *Multimedia content is disabled. This setting prevents music, animations, and video clips from running.*

While some of these default settings (such as disabling multimedia content) are perfectly logical for a server, it is nearly inconceivable that anyone who uses Windows Server 2003 will leave the settings described in the first item unchanged. These settings make the Internet Explorer browser nearly useless to the server administrator who wants to perform any browser-based administrative tasks, download updates, etc. To lower the severity rank based on the assumption that Windows Server 2003 users will leave these default settings as they are is a fantasy, at best. If Windows Server 2003 users were encouraged to administer the server remotely, that might mitigate this risk. But Microsoft clearly promotes the local familiar Windows desktop as the prime advantage to Windows Server 2003.

2. There are flaws in the list below that, when exploited, are limited in severity according to the privileges of the user. We have faithfully recorded these cases by specifying "User" in the category "Privileges". However, since Windows Server 2003 is, indeed, a server, it stands to reason that most people who directly interact with any machine running Windows Server 2003 will be administrators. Even assuming everyone follows best practices at the desktop; Windows Server 2003 administrators are obviously going to log in with administrator privileges. So in the

[6] See Resources for URL for page from which data was extracted
[7] See Resources for URL for page from which text is quoted

cases where the severity of flaws are "limited" by user privileges, most of the time the severity will actually be unlimited, because the user will have administrator privileges. Accordingly, to cite one example, the flaw described in Microsoft Security Bulletin MS04-015 deserves a rating of Critical rather than Important. Ironically, similar flaws in Linux deserve a lower rating because Linux does not encourage administrators to work at the server with a graphical desktop.

All things considered, we would rank, at minimum, five more of the vulnerabilities to be Critical. That means 50% of the listed flaws would be rated as Critical according to this report's own severity metrics as described in the previous sections. We listed in parenthesis those vulnerabilities that should be rated as Critical, given that the average administrator is likely to change the default settings that Microsoft uses to lower the severity. We did not, however, count these as Critical in our overall comparison. The parenthetical comments are there to illustrate that Microsoft is deliberately underestimating the severe nature of these flaws due to an unreasonable condition – that the default settings of Windows Server 2003 make the difference.

Date	Windows Server 2003	Description	Method	Pathway	Access	Privileges	Damage	User Interaction	Microsoft Severity Rating
September 14, 2004	Microsoft Security Bulletin MS04-028	Buffer Overrun in JPEG Processing (GDI+) Could allow code execution	Specially crafted JPEG image	Dozens of applications	Remote Internet	Admininstrator	Complete control, Unlimited, DoS (Server stops responding)	Required	Critical
July 30, 2004	Microsoft Security Bulletin MS04-025	Navigation Method Cross-Domain Vulnerability	Malicious Web Site	IE	Remote Internet	User	Complete control, Unlimited, DoS (Server stops responding)	Required	Moderate (should be Critical)
July 30, 2004	Microsoft Security Bulletin MS04-025	Malformed BMP File Vulnerability	Malicious Web Site	IE	Remote Internet	User	Complete control, Unlimited, DoS (Server stops responding)	Required	None
July 30, 2004	Microsoft Security Bulletin MS04-025	Malformed GIF File Vulnerability	Malicious Web Site	IE	Remote Internet	User	Complete control, Unlimited, DoS (Server stops responding)	Required	Critical
July 13, 2004	Microsoft Security Bulletin MS04-024	Vulnerability in Windows Shell Could Allow Remote Code Execution	HTML Email, Visit Malicious Web Site	IE	Remote Internet	User	Complete control, Unlimited, DoS (Server stops responding)	Required	Important (Should be Critical)
July 13, 2004	Microsoft Security Bulletin MS04-023	Vulnerability in HTML showHelp Could Allow Code Execution	HTML Email, Visit Malicious Web Site	IE, Help and Support Center	Remote Internet	User	Complete control, Unlimited, DoS (Server stops responding)	Required	Critical
July 13, 2004	Microsoft Security Bulletin MS04-023	Vulnerability in HTML Help Could Allow Code Execution	HTML Email, Visit Malicious Web Site	IE, Help and Support Center	Remote Internet	User	Complete control, Unlimited, DoS (Server stops responding)	Required	Critical
July 13, 2004	Microsoft Security Bulletin MS04-018	Cumulative Security Update for Outlook Express	Specially Crafted E-mail Header	Outlook Express 6	Remote Internet	User	Denial of Service (Causes Outlook	No	Moderate

							Express to fail)		
June 8, 2004	Microsoft Security Bulletin MS04-017	Vulnerability in Crystal Reports Web Viewer Could Allow Information Disclosure and Denial of Service	Specially Crafted HTTP Request	Visual Studio .Net, IIS	Remote Internet	Service	Delete files, Access Privileged Information, Denial of Service (DoS)	No	Moderate
June 8, 2004	Microsoft Security Bulletin MS04-016	Vulnerability in DirectPlay Could Allow Denial of Service	Send a malformed packet to server	IDirectPlay4	Remote Internet	Service	Denial of Service (DoS) of Multiplayer Game Server	No	Moderate
May 11, 2004	Microsoft Security Bulletin MS04-015	Vulnerability in Help and Support Center Could Allow Remote Code Execution	HTML Email, Visit Malicious Web Site	IE, Help and Support Center	Remote Internet	User	Complete control, Unlimited, DoS (Server stops responding)	Required	Important (should be Critical)
April 13, 2004	Microsoft Security Bulletin MS04-014	Vulnerability in the Microsoft Jet Database Engine Could Allow Code Execution	Send specially crafted query to Jet (SQL) engine	Jet Engine (SQL Server), IIS	Remote Internet	Service	Complete control, Unlimited, DoS (Server stops responding)	No	Important
April 13, 2004	Microsoft Security Bulletin MS04-013	Cumulative Security Update for Outlook Express	HTML Email, Visit Malicious Web Site	MHTML Handling of Outlook Express	Remote Internet	Administrator	Complete control, Unlimited, DoS (Server stops responding)	Yes	Critical
April 13, 2004	Microsoft Security Bulletin MS04-012	RPC Runtime Library Vulnerability	Send an RPC message	RPC	Remote Internet	Administrator	Complete control, Unlimited, DoS (Server stops responding)	No	Critical
April 13, 2004	Microsoft Security Bulletin MS04-012	RPCSS Service Vulnerability	Send a specially crafted message	RPCSS	Remote Internet	Service	DoS (RPCSS Service stops responding)	No	Important
April 13, 2004	Microsoft Security Bulletin MS04-012	RPC over HTTP Vulnerability	Send a specially crafted message	IIS/COM Internet Services	Remote Internet	User, Service	DoS (Server stops responding)	No	Low
April 13, 2004	Microsoft Security Bulletin MS04-012	Object Identity Vulnerability	Send a specially crafted message, needs valid login ID	IIS/COM	Remote Internet	Service, Administrator	DoS (Need to restart IIS)	No	Low
April 13, 2004	Microsoft Security Bulletin MS04-011	LSASS Vulnerability	Send a specially crafted message	LSASS	Local Administrator Only	N/A	Complete control, Unlimited, DoS (Server stops responding)	Required	Low

April 13, 2004	Microsoft Security Bulletin MS04-011	PCT Vulnerability	Send a specially crafted TCP message	PCT/SSL, SSL-enabled apps (IIS)	Remote Internet	Administrator	Complete control, Unlimited, DoS (Server stops responding)	No	Low
April 13, 2004	Microsoft Security Bulletin MS04-011	Vulnerability in HTML Help Could Allow Code Execution	HTML Email, Visit Malicious Web Site	HTML Help	Remote Internet	User	Complete control, Unlimited	Required	Critical
April 13, 2004	Microsoft Security Bulletin MS04-011	H.323/ICF Vulnerability	Send a specially crafted message	NetMeeting	Remote Internet	Administrator	Complete control, Unlimited, DoS (Server stops responding)	No	Important
April 13, 2004	Microsoft Security Bulletin MS04-011	Negotiate SSP Vulnerability	Send a specially crafted message	IIS	Remote Internet	Administrator	Complete control, Unlimited, DoS (Server stops responding)	No	Critical
April 13, 2004	Microsoft Security Bulletin MS04-011	SSL Vulnerability	Send a malformed message	IIS/SSL	Remote Internet	N/A	DoS, Reboots System	No	Important
April 13, 2004	Microsoft Security Bulletin MS04-011	ASN.1 "Double Free" Vulnerability	Specially Crafted Authentication Request	ASN.1, used by many applications	Remote Internet	Administrator	Complete control, Unlimited, DoS (Server stops responding)	No	Critical
February 10, 2004	Microsoft Security Bulletin MS04-007	ASN.1 Vulnerability Could Allow Code Execution	Specially Crafted Authentication Request	ASN.1, used by many applications	Remote Internet	Administrator	Complete control, Unlimited, DoS (Server stops responding)	No	Critical
February 10, 2004	Microsoft Security Bulletin MS04-006	Vulnerability in the Windows Internet Naming Service (WINS) Could Allow Code Execution	Specially Crafted Message, Buffer overrun	WINS	Remote Internet	Administrator	Denial of Service (Causes WINS to stop responding), potential complete control	No	Important
February 2, 2004	Microsoft Security Bulletin MS04-004	Cross-Domain Vulnerability	HTML Email, Visit Malicious Web Site	IE	Remote Internet	User	Complete control, Unlimited	Required	Moderate
February 2, 2004	Microsoft Security Bulletin MS04-004	Drag-and-Drop Operation Vulnerability	HTML Email, Visit Malicious Web Site	IE	Remote Internet	User	Download programs without notification	Required	Moderate
February 2, 2004	Microsoft Security Bulletin MS04-004	Improper URL Canonicalization	HTML Email, Visit Malicious Web Site	IE	Remote Internet	User	Spoof web site	Required	Important
January 13, 2004	Microsoft Security Bulletin MS04-003	Buffer Overrun in MDAC Function Could Allow Code Execution	Spoof a local SQL Server	MDAC	Remote Internet	Service	Complete control, Unlimited, DoS (Service stops responding)	No	Important

January 13, 2004	Microsoft Security Bulletin MS04-001	Vulnerability in Microsoft Internet Security and Acceleration Server 2000 H.323 Filter Could Allow Remote Code Execution	Send Specially Crafted Message, Buffer overrun	Microsoft Firewall Service, Microsoft Internet Security and Acceleration Server	Remote Internet	Administrator	Complete control, Unlimited, DoS (Server stops responding)	No	Critical
November 11, 2003	Microsoft Security Bulletin MS03-048	Cross-Domain Vulnerability	HTML Email, Visit Malicious Web Site	IE	Remote Internet	User	Complete control, Unlimited	Required	Moderate (should be Critical)
November 11, 2003	Microsoft Security Bulletin MS03-048	XML Object Vulnerability	HTML Email, Visit Malicious Web Site	IE	Remote Internet	User	Attacker can read known files on system	Required	Low
November 11, 2003	Microsoft Security Bulletin MS03-048	Drag-and-Drop Operation Vulnerability	HTML Email, Visit Malicious Web Site	IE	Remote Internet	User	Complete control, Unlimited	Required	Moderate (should be Critical)
October 15, 2003	Microsoft Security Bulletin MS03-045	Buffer Overrun in the ListBox and in the ComboBox Control Could Allow Code Execution	Exploit flaw in graphical control	Windows API	Local user with valid login ID	User	Complete control, Unlimited	No	Low
October 15, 2003	Microsoft Security Bulletin MS03-044	Buffer Overrun in Windows Help and Support Center Could Lead to System Compromise	HTML Email, Visit Malicious Web Site	IE, Help and Support Center, HCP Protocol	Remote Internet	Administrator	Complete control, Unlimited, DoS (Server stops responding)	Required	Critical
October 15, 2003	Microsoft Security Bulletin MS03-043	Buffer Overrun in Messenger Service Could Allow Code Execution	Specially Crafted Message	Messenger Service, Disabled by default	Remote Internet	Administrator	Complete control, Unlimited, DoS (Server stops responding)	No	Critical
October 15, 2003	Microsoft Security Bulletin MS03-041	Vulnerability in Authenticode Verification Could Allow Remote Code Execution	Malicious ActiveX control used without permission under low-memory conditions	ActiveX Authentication	Remote Internet	User	Complete control, Unlimited	Required	Critical
September 10, 2003	Microsoft Security Bulletin MS03-039	Buffer Overrun In RPCSS Service Could Allow Code Execution	Specially Crafted Message	RPCSS	Remote Internet	Administrator	Complete control, Unlimited, DoS	No	Critical

Patches and Vulnerabilities Affecting Red Hat Enterprise Linux AS v.3

The following table contains information about the vulnerabilities from the 45 most recent security patches made available by Microsoft. [8]

Red Hat does not assign a severity rating. We used the metrics defined in this report to evaluate the severity of each item, along with the consideration that Linux servers are routinely administered from desktop systems, not from a graphical interface at the server itself. Many of the severity ratings are notated with brief explanations which may help the reader understand the rating.

Of the 40 vulnerabilities, only 4 are rated as Critical. That means 10% of the most recent 40 updates are of Critical severity.

It is arguable that two of these four items do not deserve to be rated as highly as they are, considering the software in question. These two items involve a program called Ethereal. Ethereal is one of several available network "sniffer" monitoring tools. One generally runs Ethereal on an as-needed basis, not as a regular service, so the chance that it will be running when someone attempts to attack its vulnerability is extremely low. If we chose to reduce the severity to Important for this reason, only 5% of the 40 most recent alerts would be considered Critical.

The IPSEC and Kerberos vulnerabilities are more deserving of the Critical status, because they are services that one would run continually.

Only a few of the vulnerabilities allow a malicious hacker to perform mischief at the administrator level. There are mitigating factors in most of these cases, however. For example, the Samba vulnerability (July 22, 2004, RHSA-2004:259-23) can only be exploited if someone configures inetd (via a file called hosts.allow) to allow a known user and host to access this service. Unless the system is poorly configured, no one except an authorized known user could reach the Samba configuration program in order to exploit the vulnerability. Otherwise, this would deserve a Critical severity rating. Other flaws that allow administrator access also require the flaw to be exploited by a known user with a valid ID. This reduces the risk and severity because of the significantly increased chances the malicious hacker will be caught.

Date	Red Hat Advanced Server	Description	Method	Pathway	Access	Privileges	Damage	User Interaction	Severity Rating
September 7, 2004	RHSA-2004:400-15	Updated gaim package fixes security issues	Send crafted data to a GAIM client	GAIM (Instant Messenger)	Remote Internet	User	Complete control, Unlimited	No	Important (Gaim not typically used on server)
September 1, 2004	RHSA-2004:323-09	An updated lha package fixes security vulnerability	Convince user to use a specially crafted command	Carefully crafted LHA archive, convince users to use a command	Download or otherwise receive lha-compressed file	User	Complete control, Unlimited	Yes	Low (lha is a rarely used outdated compression format)
September 1, 2004	RHSA-2004:349-10	Updated http packages fix mod_ssl security flaw	Abort an SSL request in a certain state	Apache 2.0.50 and earlier	Remote Internet	Service	Consume CPU resources (potential DoS)	No	Important
September 1, 2004	RHSA-2004:436-07	Updated rsync package fixes security issue	Send crafted rsync command	rsync 2.6.2 and earlier	Remote Internet	Service	Read/write files not defined as accessible by rsync	No	Important (rsync not a common publicly accessible service, and chroot negates this vulnerability)
August 31, 2004	RHSA-2004:350-12	Updated krb5 packages fix security issues	Send crafted authentication request	Kerberos authentication	Remote Internet	Administrator	Complete control, Unlimited, DoS (Server stops	No	Critical

[8] See Resources for URL for page from which data was extracted

							responding)		
August 26, 2004	RHSA-2004:432-08	Updated acrobat package fixes security issues	Crafted uuencoded file	Acrobat Reader	Remote Internet	User	Complete control, Unlimited	Yes	Important (Acrobat not typically used on server)
August 20, 2004	RHSA-2004:414-19	Updated qt packages fix security issues	Crafted image file	Qt (toolkit used by KDE)	Remote Internet	User	Crash Qt, possibly execute code	Yes	Important
August 5, 2004	RHSA-2004:378-08	Updated Ethereal packages fix security issues	Send malicious packets	Ethereal network monitor	Remote Internet	Administrator	Crash Ethereal, possibly execute code	No	Critical
August 4, 2004	RHSA-2004:373-13	GNOME VFS updates address extfs vulnerability	Convince a user to open a special URI	GNOME-VFS	N/A	User	Perform actions as the target user	Yes	Low
August 4, 2004	RHSA-2004:402-08	Updated libpng packages fix security issues	Create carefully crafted png file, entice user to web site	libpng	Remote Internet	User	Complete control, Unlimited	Yes	Important (Browser not typically used on server)
August 4, 2004	RHSA-2004:421-17	Updated mozilla packages fix security issues	Several openings, including malicious javascript	Mozilla browser	Remote Internet	User	Complete control, Unlimited	Yes	Important (Browser not typically used on server)
August 3, 2004	RHSA-2004:413-07	Updated kernel packages fix security vulnerabilities	Access large amounts of memory	Kernel	Local user with valid ID	N/A	DoS (Server stops responding)	Yes	Low
July 29, 2004	RHSA-2004:308-06	Updated ipsec-tools package	Verify X.509 certificate	ipsec-tools	Remote Internet	N/A	Does not abort key exchange if verification fails	No	Important
July 29, 2004	RHSA-2004:409-05	Updated sox packages fix buffer overflows	Specially crafted WAV file	sox (Sound eXchange)	Remote Internet	User	Complete control, Unlimited, DoS (Server stops responding)	Yes	Important
July 22, 2004	RHSA-2004:259-23	Updated samba packages fix vulnerabilities	Specially crafted HTTP authentication	Samba (Windows services)	Administrator	Administrator	Complete control, Unlimited, DoS (Server stops responding)	Yes	Low (user is pre-authenticated by inetd/hosts.allow)
July 19, 2004	RHSA-2004:392-13	Updated php packages fix security issues	Obscure hash attack	PHP	Remote Internet	Service	Execute code as Apache user	No	Low (extremely difficult to exploit, depends on site construction)

July 6, 2004	RHSA-2004:342-10	Updated httpd packages fix security issues	Fake SSL certificate authority that SSL is configured to trust, or consume memory	Apache with SSL	Remote Internet	Service	Execute code as Apache user, possible DoS	No	Moderate (due to possible DoS attack)
July 2, 2004	RHSA-2004:360-05	Updated kernel packages fix security vulnerabilities	Mount NFS file system from a vulnerable machine	Kernel	Local user with valid login ID, NFS must be running	Group	Possibly change a file to be owned by a different group	No	Low
June 18, 2004	RHSA-2004:249-07	Updated libpng packages fix security issue	Create carefully crafted png file, entice user to web site	libpng	Remote Internet	User	Complete control, Unlimited, DoS (Server stops responding)	Yes	Important
June 17, 2004	RHSA-2004:255-10	Updated kernel packages fix security vulnerabilities	Run functions such as fsave and frstor	Kernel	Local user running programs designed to make the kernel fail	N/A	Denial of Service (Server stops responding)	Yes	Low (attacker must run programs on the server)
June 14, 2004	RHSA-2004:240-06	Updated SquirrelMail package fixes multiple vulnerabilities	Mail user can run crafted URL	PHP, Squirrelmail	Remote Internet user with valid login ID	Service	Modify contents of database, run as other web mail users	No	Important (requires user with valid account)
June 9, 2004	RHSA-2004:233-07	Updated cvs package fixes security issues	Send crafted instructions to CVS	CVS	Remote Internet user with valid login ID	Service	Execute code with CVS user privileges	No	Important (requires user with valid account)
June 9, 2004	RHSA-2004:234-06	Updated Ethereal packages fix security issues	Send malicious packets	Ethereal network monitor	Remote Internet	Administrator	Complete control, Unlimited, DoS (Server stops responding)	No	Critical
June 9, 2004	RHSA-2004:236-14	Updated krb5 packages available	Use malformed authentication names	Kerberos authentication	Remote Internet	Administrator	Unknown	No	Low (Kerberos on Red Hat is not configured by default to have this vulnerability)
June 9, 2004	RHSA-2004:242-06	Updated squid package fixes security vulnerability	Send overly long password	Squid proxy and cache	Local user with valid ID	Service	Execute code with Squid user privileges	No	Low (Requires valid user; Squid is not configured with this vulnerability by default)
May 26, 2004	RHSA-2004:174-09	Updated utempter package fixes vulnerability	If utempter is active, write application that exposes this flaw	utempter	Local or remote user with valid ID	Administrator	Overwrite privileged files with symlink	No	Low (Requires valid user; utempter is obscure, exploit is difficult)

May 26, 2004	RHSA-2004:219-07	Updated tcpdump packages fix various vulnerabilities	Specially crafted ISAKMP packets	tcpdump	Remote Internet	N/A	Causes tcpdump to crash	No	Low (tcpdump is simply a utility administrators use to examine tcp traffic)
May 21, 2004	RHSA-2004:064-11	Updated samba packages fix security vulnerability	Accidental change to samba account	Samba (Windows services)	N/A	N/A	May change a user's password to something more easily guessed	Yes	Low (extremely unlikely accident, unlikely consequences)
May 21, 2004	RHSA-2004:120-12	Updated OpenSSL packages fix vulnerabilities	Send crafted SSL packets	OpenSSL	Remote Internet	N/A	May cause OpenSSL to crash, Denial of Service (OpenSSL stops responding)	No	Important (due to possible DoS attack)
May 19, 2004	RHSA-2004:180-10	Updated libpng packages fix crash	Craft a special png image, entice to web site	libpng	Remote Internet	N/A	Causes application displaying image to crash	Yes	Low (restart application after it crashes)
May 19, 2004	RHSA-2004:190-14	Updated cvs package fixes security issue	Craft a special CVS command	CVS	Local or remote user with valid ID	Service	Execute code with CVS user privileges	No	Important (requires user with valid account)
May 19, 2004	RHSA-2004:192-06	Updated rsync package fixes security issue	Send crafted rsync command	rsync	Remote Internet	Service	Read/write files not defined as accessible by rsync	No	Important (rsync not a common publicly accessible service, and chroot negates this vulnerability)
May 17, 2004	RHSA-2004:222-11	Updated kdelibs packages resolve URI security issues	Crafted URI, entice user to web site	KDE	Remote Internet	User	Complete control, Unlimited	Yes	Important
May 11, 2004	RHSA-2004:165-09	Updated ipsec-tools package fixes vulnerabilities in ISAKMP daemon	attacker crafts an ISAKMP header with a extremely large value	ipsec-tools	Remote Internet	N/A	Denial of Service (Server stops responding)	No	Critical
May 11, 2004	RHSA-2004:188-14	Updated kernel packages available for Red Hat Enterprise Linux 3 Update 2	Bug-fix release. Most serious bug is possible privilege escalation when mounting Netware volumes	Kernel	Local or remote user with valid ID	N/A	N/A	No	Low (obscure bug fixes)
April 22, 2004	RHSA-2004:183-03	Updated kernel packages fix security vulnerabilities	Write program to gain root (administrator) privileges	Kernel	Local user with valid ID	Administrator	Complete control, Unlimited	No	Important (requires user with valid account)

April 17, 2004	RHSA-2004:153-09	Updated CVS packages fix security issue	Fake paths to overwrite files	CVS	Local or remote user with valid ID	Service	Overwrite files outside CVS directories	No	Important (requires user with valid account)
April 14, 2004	RHSA-2004:133-12	Updated squid package fixes security vulnerability	Craft URLs to view restricted web sites	Squid proxy and cache	Local or remote user with valid ID	N/A	View web pages Squid is configured to block	No	Moderate (Basically a way to trick Squid into allowing access to restricted sites, such as porn sites, but may also be used to access blocked Intranet pages)
April 14, 2004	RHSA-2004:160-05	Updated OpenOffice packages fix security vulnerability in neon	Craft format strings, entice user to visit site	OpenOffice	Remote Internet	User	Execute code	Yes	Moderate (OpenOffice not typically used on server)

CERT Vulnerability Notes Database Results

The United States Computer Emergency Readiness Team (CERT) uses its own set of metrics to evaluate the severity of any given security flaw. A number between 0 and 180 expresses the final metric, where the number 180 represents the most serious vulnerability. The ranking is not linear. In other words, a vulnerability ranked 100 is not twice as serious as a vulnerability ranked at 50.

CERT considers any vulnerability with a score of 40 or higher to be serious enough to be a candidate for a special CERT Advisory and US-CERT technical alert.

We queried the CERT database using the search terms "Microsoft", "Red Hat", and "Linux".[9] While the CERT web search capabilities do not produce perfectly desirable results in terms of granularity or longevity. This is especially true for the search results for "Red Hat" and "Linux". The "Linux" search results include a number of Oracle security vulnerabilities that are common to Linux, UNIX, and Windows. The details of the most severe "Red Hat" entry does not even list Red Hat as a vulnerable system. The results for the "Microsoft" search seem to be almost entirely accurate, inasmuch as both the details and entries refer to flaws in Microsoft-specific software. As a result, the results are somewhat unfairly skewed against Linux and Red Hat. Nevertheless, even if one takes the results at face value and ignores the skewed results for Linux and Red Hat, Microsoft still produces the most entries in the CERT database, and the list of entries contain the most severe flaws.

The CERT results for "Microsoft" returned 250 entries, with the top two entries containing the severity metric of 94.5. Thirty-nine entries have a severity rating of 40 or greater. The average severity rating for the top 40 entries is 54.67. (We chose to average 40 entries instead of 50 or more because the Red Hat search only returned 49 results.)

The CERT results for "Red Hat" returned 46 entries. The top entry has a severity metric of 108.16. Only 3 (vs. 39 for Microsoft) entries have a metric of 40 or greater. The average severity for the top 40 entries is 17.96.

The CERT results for the "Linux" search returned 100 entries. The top entry has a severity metric of 87.72. Only 6 of the entries carry a severity metric of 40 or greater. The average severity for the top 40 entries is 28.48.

These results cannot be expected to mirror our own analysis of recent vulnerability patches. The CERT search criteria and date ordering is different, and the CERT search does not confine the products to Windows Server 2003 and Red Hat Enterprise Linux AS v.3. But the CERT results reflect how Windows security flaws tend to be far more frequently severe than those of Linux, which echoes our conclusions.

[9] See the References section below for the full URLs we used to perform these searches.

References

Netcraft Web Survey for September 2004
http://news.netcraft.com/archives/2004/08/31/September_2004_seb_server_survey.html

Netcraft Top 50 Servers With Longest Uptime (results may differ since the information changes daily)
http://uptime.netcraft.com/up/today/top.avg.html

Unpatched PC "Survival Time" Just 16 Minutes, Gregg Keizer, TechWeb News
http://www.internetweek.com/breakingNews/showArticle.jhtml?articleID=29106061

Top 10 Benefits of Windows Server 2003
http://www.microsoft.com/windowsserver2003/evaluation/whyupgrade/top10best.mspx

Microsoft Security Bulletin, Current Downloads
http://www.microsoft.com/technet/security/CurrentDL.aspx

Default Settings Different on Windows Server 2003
These settings are enumerated on several alert pages under "Frequently Asked Questions, What is Internet Explorer Enhanced Security Configuration?" The following is one such URL.
http://www.microsoft.com/technet/security/bulletin/ms03-032.mspx

Red Hat Enterprise Linux Advance Server v.3 Security Advisories
https://rhn.redhat.com/errata/rhel3as-errata-security.html

CERT search for Microsoft Alerts
http://www.kb.cert.org/vuls/bymetric?searchview&query=microsoft&searchorder=4&count=100

CERT search for Red Hat Alerts
http://www.kb.cert.org/vuls/bymetric?searchview&query=red*hat&searchorder=4&count=100

CERT search for Linux Alerts
http://www.kb.cert.org/vuls/bymetric?searchview&query=linux&searchorder=4&count=100