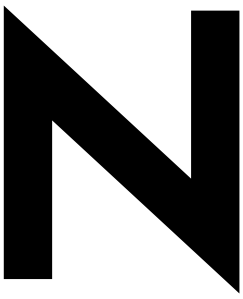
Microsoft Windows*:

An Inherently Insecure Platform

www.novell.com

WHITE PAPER



Novell®

Table of Contents

Executive Summary

Microsoft Windows* security, or the lack thereof, is synonymous with famous viruses and worms, long delays in patching critical security vulnerabilities, and large costs to users for installing patches and service packs (SPs) to resolve the aforementioned security flaws. According to one leading analyst firm, security is important to Microsoft—but only to the extent that security does not inhibit the adoption of Microsoft's products.[1]

Recently, Microsoft has made a significant investment in improving the security of its Windows platform. It has released thousands of security code fixes, and it has purchased several small companies that each patch a specific security vulnerablity inside the Windows platform; however, these fixes are a case of too little, too late. The security flaws in the Microsoft Windows platform exist at a deeper level—in the fundamental architecture of the platform.

Many of the security problems in Windows operating systems can be traced back to the code that was developed from 1992 –1996 for the Windows NT* platform—in other words, code that was developed before the Internet became pervasive. Because every release of the Windows platform leverages the code base of the previous generation, there are many years of Windows code—most likely, tens of millions of lines—that was written before security in Windows became a primary concern for Microsoft. The bad news for today's users is that much of this code is included in Windows Server 2003 and will be included in Longhorn, the next generation of the Microsoft Windows platform.

In this *White Paper*, we present arguments, based on fundamental operating system security research, that show that Windows operating systems are inherently insecure and will almost certainly remain insecure. The basic problem is that Windows operating systems were created without security as a primary design objective, and as a result, *it is nearly impossible to make Windows secure no matter how much money is spent trying to do so.* Attempting to make Windows secure is similar to trying to plug a leak in a poorly designed dam holding back water—plug one hole and another develops.

The four primary reasons that Windows is an inherently insecure operating system can be summed as:

1. Windows was designed as a single-user operating system.

[1] **http://www.themacobserver.com/article/2000/08/16.9.shtml**.

2. Windows was designed without security as a primary goal.
3. Microsoft's competitive strategy led it to embed more functionality and application integration into Windows in an attempt to thwart the competition, resulting in a very large and complex operating system with many highly dependent pieces of code.
4. Microsoft's basic lack of attention to security after the initial Windows implementation.

MICROSOFT WINDOWS SECURITY TODAY

Every day, two conflicting scenarios play out again and again:

- Systems administrators who work with both Linux* and Windows say that Linux is more secure and easier to maintain and patch than Windows
- Microsoft executives tell customers how much work Microsoft is doing to make Windows secure and how much more secure Windows is than Linux based on the meaningless metric—the number of vulnerability bulletins released each month or year

Microsoft tries to deflect Windows security problems through marketing campaigns and hiring marketing research companies to write white papers indicating that Windows is more secure than Linux. But the final decision maker on security is you—the end user. So who do you believe—other end users like yourself, or executives from Microsoft who talk about how much money Microsoft is spending on Windows security?

In a panel discussion at the LinuxWorld Expo in Boston in February 2005, Steve Doherty of the Cambridge Health Alliance (CHA) said that his healthcare organization has about 200 servers. Most of CHA's servers run Windows operating systems, but CHA runs some mission-critical applications on Linux. When asked to compare Linux and Windows security, Doherty said that Windows is less secure and more difficult to patch and maintain than Linux.

"We have fewer patches on Linux than on Windows," Doherty said. "It is a tremendous amount of work to patch Windows. With most Windows patches, we have to reboot the system, but with most Linux patches we do not have to reboot. We find that there are additional security costs with Windows in addition to the costs of downtime for patching—anti-virus costs and other add-on products."

In a June 2004 report, Burton Group, a marketing research firm, stated "Although Microsoft Windows Server 2003 reduces some risks, it increases other risks. This is due to its greater size and complexity, the widespread use of insecure application deployment models and other style of use issues, and proven vulnerability to large-scale effects from worms and viruses. These security concerns have implications when considering how appropriate Windows Server 2003 is in roles such as application server or remote access server."[2]

WHY WINDOWS IS AN INHERENTLY INSECURE OPERATING SYSTEM: PROOF POINTS

Microsoft's philosophy about security is one of ambivalence. Too many platform design decisions

[2] *Burton Group*, Windows Server 2003 Security: Making Progress, But Serious Concerns Remain, *February 24, 2004.*

were made without regard for security and the impact of the Internet on security. Instead of designing security into the kernel of the Windows operating system, Microsoft places much of the responsibility for security on the user. For example, when it comes to account management, Microsoft expects users to understand the importance of renaming built-in accounts or creating a backup administrator account, instead of building the functionality directly into the shipping product.

Windows Server 2003 still provides large amounts of functionality that is unnecessary, too complex or too general-purpose for some operating system roles. For example, the Windows Server 2003 operating system exposes DCOM (Distributed Component Object Module) functionality through the same remote procedure call port that was used successfully by the Blaster worm to attack other Windows operating systems. A risk for Active Directory* replication exists because it requires the use of DCOM. To mitigate this problem, Windows Server 2003 disables IE and Internet Information Services (IIS) by default so that a domain administrator cannot infect the controller with hostile ActiveX or other mobile code while browsing the Internet. But there is no guarantee that IE, IIS, and ActiveX—still resident in the code base—won't be toggled back on by a careless administrator or by a malicious program.

Basic U.S. government-funded operating system security research completed in the late 1970s and early 1980s at places like the non-profit Mitre Corporation concluded that:

- It is not possible to appreciably improve security in an operating system (not designed with security in mind) by just adding on software packages such as Microsoft Service Packs (SPs)
- The larger and more complex the operating system, the more impossible the task of making it secure

Below, we identify a number of weaknesses in Windows security, some of which were derived from Burton Group's February 2004 paper on Windows Server 2003 security[3], that indicate why Windows is an inherently insecure operating system. The features of Windows operating systems that prevent them from becoming as secure as Linux include:

1. **Windows was designed as a single-user operating system**. Originally, this design allowed both users and applications almost unlimited access to the entire system. Linux was developed as a multi-user system; therefore, it was designed from the ground up to isolate users from applications, files and directories that affect the security of the entire operating system.
2. **Windows' monolithic design integrates too many features (e.g., Internet Explorer) into the core of the operating system**. This monolithic design creates many inter-dependencies. For example, any flaw in IE can expose the Windows operating systems to risks well beyond what users should be able to do with a browser.
3. **Windows operating systems are large and complex.** Complexity is the single largest

[3] *Burton Group*, Windows Server 2003 Security: Making Progress, But Serious Concerns Remain, *February 24, 2004.*

enemy of security.[4] Windows operating systems consist of between 40-million and 60-million lines of code. Comparitively, the 2.6 Linux kernel consists of 5.7 million lines of code). Secure operating system design requires that the kernel of an operating system be as small and as simple as possible.

4. **Windows relies heavily on the Remote Procedure Call (RPC) Model**. RPCs are security risks because they are designed to allow other computers attached to a network to tell your computer what services it wants it to perform. Windows operating systems depend on RPC, even when they are not on a network, because many Windows services depend on RPC mechanisms.
5. **Windows' focus on developers**. This focus makes flexibility, integration and extensibility more important goals than security. Focusing on developers and not security is not necessarily a bad thing. But it means that Windows operating systems should not be used in high-risk situations or for hosting mission-critical applications.
6. **Basic lack of attention to security**. Microsoft has provided users with many features to make Windows more attractive and more competitive, but in many cases it has done so without regard to the security implications.

We will now examine each of these six design areas in more depth.

A Single User Operating System

Microsoft operating systems have their roots in single-user designs, and Microsoft has gradually tried to evolve them to provide multi-user server operating systems. On the other hand, UNIX* and Linux were designed to be multi-user systems from the beginning. Because Windows was developed as a single-user operating system that allowed users and applications access to the entire system, leftover single-user security holes exist in Windows operating systems, including Windows Server 2003, and some of these holes will undoubtedly be in Longhorn.

Windows XP was the first Windows operating system to make a serious effort to isolate users from the system so that users each have their own private files and limited system privileges. But this caused some legacy Windows applications to fail because they were accustomed to being able to access and modify programs and files that only an administrator should be able to access. This design flaw is 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. As Microsoft provides service packs to make Windows act more like a multi-user operating system, many applications break because they are used to working without these restraints. Windows Server 2003 has made some progress toward true multi-user capabilities, but there are still holdover security issues such as ActiveX.

Monolithic/Tight Integration

Windows is a monolithic, non-modular design resulting in the integration of too many features into the core of the operating system. In the early Windows operating systems—Windows 95, 98 and

[4] *Karger, Paul A. and Schell, Roger A.* Thirty Years Later: Lessons from the Multics Security Evaluation, *(***http://www.acsac.org/2002/papers/classic-multics.pdf***)*.

2000—Microsoft focused on ease of use and integration as competitive features. Security was not on the road map in any significant way.

Two of the best examples of tight integration are IE and IIS.[5] This type of integration was done by Microsoft partly to prevent competitive software (e.g., Netscape*) from being used on Windows and serves as a good example of Microsoft's design philosophy: building Windows to prevent competition at the expense of security. This philosophy was good for creating market share, but it resulted in making Windows extremely vulnerable to attackers and inappropriate for high-risk applications.

Focusing on ease of use and tight integration with Windows has not only resulted in serious security flaws, but it has increased the cost of patching Windows because of the interdependencies caused by the tight integration. Tight integration of applications results in longer times to create fixes because regression testing must include the entire operating system.

During a conversation with CNET News.com, a Gartner analyst noted that Microsoft should rebuild IE with security in mind from the bottom up, rather than making "evolutionary" security improvements to the browser software.[6] It is unlikely that Microsoft will re-architect IE, if ever, to detach it from the Windows core code because it would lower its competitive advantage over other browsers such as Firefox*. IE is invoked/used by the Windows help system, Outlook, and several other Microsoft and third party applications. Thus, flaws in IE have a greater impact than flaws in a stand-alone browser.

In a Linux client, the browser (and other clients) are generally not part of the operating system—they run in user space and not in kernel space. Browsers are stand-alone applications hosted by the operating system, making it more difficult to use browsers on Linux as entry points for circumventing operating system security. This design strategy also means that the browser software can be installed as a non-root (non-administrator user). Unlike patches to IE, patches to a browser running on Linux do not involve interdependency issues with the Linux kernel.

Windows is Large and Overly Complex

Complexity is the enemy of security. The more complex an operating system, the more points that exist for a hacker to attack. The size and complexity of the underlying operating system results in a larger number of attackable vulnerabilities over a period of time, and a higher number of hours spent per month to keep secure or recover from mass attacks. Windows is comprised of 40 to 60 million lines of code that includes applications tightly integrated with the Windows kernel. The Linux kernel is less than six million lines of code, or 15 percent of the size of the Windows kernel.

RPC Dependence

Windows users cannot disable RPC because Windows operating systems and services depend upon it, even when a computer is not connected to a network. Some of the most serious vulnerabilities in Windows are due to flaws in how the RPC protocol is implemented in Windows and not in the protocol itself.[7] The most common way to

[5] *IIS version 6 was re-architected and partially re-written for the Windows Server 2003 release, making it somewhat more secure than previous versions of IIS.*

[6] **http://news.com.com/Gartner+takes+Microsoft+to+task/2100-7355_3-5582742.html**.

[7] **http://www.theregister.co.uk/security/security_report_windows_vs_linux**. *Much of the material around RPC dependence was taken from this article written by Nicholas Petreley. For more information on this important topic, we suggest that the reader download the article.*

exploit an RPC-related vulnerability is to attack the service that uses RPC, not the RPC itself.

Perhaps one of the most dangerous worms to hit the Internet was the Slammer worm in January 2003. It exploited the Microsoft implementation of RPC by exploiting flaws in Microsoft SQL Server—multiple instances of SQL Server are able to run on a single machine. Microsoft implemented this concept via RPCs. SQL Server is not integrated into Windows in the manner that IE is, but there is a good chance that it will be integrated into the new Windows file system, WinFS, in Windows Longhorn Server by the 2008—2009 timeframe. This integration would be a huge mistake. In August 2003, the Blaster-A worm emerged to take advantage of the Windows RPC-DCOM (DCOM is another area of vulnerability of Windows). Interestingly enough, older versions of Windows that lacked the RPC function were not affected by the worm.

The Danger of Developer Focus

Microsoft has tried to ensure that Windows is the most widely used operating system in the world. This goal has led Microsoft to make Windows very developer-friendly and easy to use, but at the expense of tight security. An operating system such as Linux, however, is evidence that an operating system can be developer friendly, relatively easy to use, and very secure. According to Evans Data's Summer 2004 Linux Developer Survey[8], 92 percent of survey respondents (500 developers) indicated that their Linux systems have never been infected with a virus. The respondents attributed the high degree of security to the open source development methodology—"more eyes on the code."

Microsoft has always focused on developers, making flexibility and extensibility paramount goals for its operating systems. Microsoft extended that philosophy from its client operating systems to its server operating systems and its application server architecture. The symmetry of the client and server development environments has been one of the most attractive features of the Windows client and server operating systems because it allows developers to leverage a common set of tools, interfaces and skills. These benefits have a downside, however, in that Microsoft and developers within Windows sometimes sacrifice security in favor of symmetry, flexibility, integration and extensibility.

A General Lack of Attention to Security

According to the Burton Group, Windows operating systems allow, in fact encourage, mixing application data and executable code.[9] Likewise, good auditing habits and least-privilege programming techniques are less prevalent in Windows than they are in Linux. These type-of-use differences, combined with the weakness of code access control in ActiveX*, make Windows servers generally more vulnerable in the application server role than Linux servers.

ActiveX, one of the most vulnerable concepts available in Windows operating systems, is a component that allows users to attach computer programs to Web pages. Users tend to like ActiveX because it allows Web pages to be much more dynamic and interactive than they could be otherwise. But, ActiveX introduces security risks—ActiveX controls have full access to the Windows operating system potentially allowing an intruder

[8] **http://www.evansdata.com/n2/pr/releases/Linux04_02.shtml**.

[9] *Burton Group*, Windows Server 2003 Security: Making Progress, But Serious Concerns Remain, *February 24, 2004.*

to take control of a machine. An ActiveX control containing a virus can be written, distributed and activated from a Web page, and the viewer of the control might never know.

Windows, including Windows Server 2003, has several built-in accounts, at different levels of security, that users can use by default. Guest and Administrator are examples of built-in accounts. Because of these default accounts, there is a common thread for attack. An attacker knows the names of the accounts, and hence needs only the passwords. Many of the default accounts cannot be deleted, but users can rename them. Built-in accounts are there for ease of use, but they increase the ease of attacking Windows.

COMPARING WINDOWS SECURITY WITH LINUX SECURITY

While Linux is increasingly viewed as a potential way to reduce enterprise IT expenditure, IT managers and CIO's should also view Linux as a strong resource in their overall security implementation. The past two releases of the Linux kernel (2.4 and 2.6) have seen such dramatic increases in securability features that organizations such as the United States National Security Agency (NSA) have launched multiple, focused efforts to leverage the higher security that Linux now affords.

The security of open source software has been both idealized and made the subject of targeted disinformation. Generally, two philosophies exist: that open source is more secure because it is more rigorously reviewed; and, that proprietary software is more secure because access to the source code is limited. While seeming contradictory, both schools of thought have validity depending on circumstances.

Open source doxology states that open source software cannot rely on obscurity for security—because the code is transparent, security requirements are much more stringent. Also, open collaboration is thought to result in the earlier discovery and correction of security flaws—an aspect of the thesis that "given enough eyeballs, all bugs are shallow."

Even the most ardent open source believers would say that neither of these two claims actually guarantees the security of all open source code. Having enough eyeballs reviewing the code depends on the open source project having a strong community, with many sharp individuals contributing to reviewing the source code.

Linux market share is rapidly growing, and some claim that the operating system may become scrutinized more closely for vulnerabilities, creating the possibility of more exploits; however, this scrutiny certainly has a benign effect, as well. Turnaround times for patches in Linux and other popular open source offerings have traditionally been very rapid, which allows proactive organizations to more quickly reap the benefits of a strong patch management strategy. Linux truly offers security by transparency—so you always know who is trying to access your data, and you can keep the bad guys out.

CONCLUSIONS

As noted above, the primary reasons that Windows is an inherently insecure operating system can be summed up in four points:

1. Windows was designed as a single user operating system.
2. Windows was designed without security as a primary goal.
3. Microsoft's competitive strategy led it to embed more functionality and application integration into Windows in an attempt to thwart the competition resulting in a very large and complex operating system with many highly dependent pieces of code.
4. Microsoft's basic lack of attention to security after the initial Windows implementation.

The impact of these four items is, *Microsoft has made Windows into an operating system that is basically impossible to make secure*. For Windows to become a secure operating system, Microsoft would have to re-architect and rewrite the code with an emphasis on separating application code, such as Internet Explorer, from the core of Windows. We do not expect Microsoft to do this, even with Longhorn, because it would remove the competitive advantages that it has with close integration of applications with Windows core code.

Microsoft is attempting to improve Windows security by adding code in the form of service packs such as Windows XP SP 2 and Windows Server 2003 SP 1. This approach cannot rectify the basic problem surrounding Windows security—Windows was not designed with security as a primary goal. This "add-on approach" only increases the size and complexity of Windows, making it even more difficult to make secure. The early computer security engineers stressed that the size of the core of the operating system should be as small as possible and applications should not be integrated with the kernel. Microsoft has violated these two basic rules of secure operating system design, and it cannot recover without re-architecting and rewriting the Windows operating system.

Novell Product Training and Support Services

For more information about Novell's worldwide product training, certification programs, consulting and technical support services, please visit:
www.novell.com/ngage

For More Information

Contact your local Novell Solutions Provider, or visit the Novell ' site at:
www.novell.com

You may also call Novell at:

1 888 321 4272 US/Canada
1 801 861 4272 Worldwide
1 801 861 8473 Facsimile

Novell, Inc.
404 Wyman Street
Waltham, MA 02451 USA

www.novell.com

Novell

462-001433-001