A Quick Take on Windows Security Evolution

Hal Berghel, University of Nevada, Las Vegas

An informal analysis of changes in configuration options and default settings in recent Windows operating systems reveals how security concerns have changed over time. The results are both reassuring and alarming.

A while back I wondered how software developers' concerns about security evolved over time. Obviously, studying under-the-hood changes would be a major research undertaking—and frankly beyond my interest. But mightn't there be a feasible shortcut, a sort of poor man's time-series analysis of security concerns? Behind this question was my speculation that the tightening user-controlled security configuration options and default settings in operating systems indicate areas of critical concern to OS developers. I therefore decided to test that hypothesis on some Windows OSs—XP, Vista, 7, and 8—that roughly coincide with the time frame of interest to me.

In the Windows world, robustness really started with the NT kernel. Before that, Windows was little more than a shell in front of DOS. NT had its own hardware abstraction layer for several platforms, a worthy TCP/IP stack, full preemptive multitasking, a native logging system, its own secure native file system (NTFS), a native Windows API, support for 32- and 64-bit architectures, symmetrical multiprocessor support, and so forth. With the introduction of NT in 1993, Windows was beginning to look like a real OS.

NT was the existential moment for Microsoft OS development. While NT proved that Microsoft could build something approaching a real OS, it was XP, introduced in 2001, that added sophistication to the NT platform. The graphical user interface was intuitive and versatile, the NT kernel was sound, and the networking and multimedia support was beginning to mature. The rest, as they say, is history. XP went on to sell over a billion copies while it was still supported by Microsoft until 2014, and remains the fourth most popular Windows OS to this day, just slightly behind 8.1 (gs.statcounter.com/os-version-market-share/windows/desktop/worldwide).

So it's with XP that I began my analysis of changes in Windows OS configuration options and default settings as a possible indicator of Microsoft's security concerns. My operative principle was that adding a new configuration

0018-9162/17/$33.00 © 2017 IEEE

EDITOR **HAL BERGHEL**
University of Nevada, Las Vegas; hlb@computer.org

parameter, or changing the default setting of a previous one, would signify increased attention toward a particular security issue. Considering these configuration changes over time should—and did—reveal some interesting trends.

ACCOUNT POLICIES

The first thing to note is that Microsoft changed its default setting for guest and support accounts with XP Service Pack (SP) 2. Prior to that, guest accounts were shipped unlocked and usable. In Windows these accounts don't require a password by default, though the local administrator can add password protection. The problem with guest accounts is twofold. First, they invite elevation-of-privilege attack attempts. Second, in XP SP1 and earlier products, the Support_xxxxxx account—used to run scripts from the Microsoft Support Center—and the HelpAssistant account—used for remote assistance—were active and remained so through upgrades. This was never a good idea, which Microsoft confirmed by deprecating this feature in Vista. In general, a computer that supports guest access and support accounts should disable them by default for maximum security. That's the case in Vista and beyond (technet.microsoft.com/en-us/library/2007.06.acl.aspx).

You can check the status of your machine in the following way (I'm using classic Windows 7 as a reference point):

> *Guest and support accounts:* Start>ControlPanel>SystemandSecurity>Administrative Tools>Computer Management>Local Users and Groups>Users. Double-click on accounts of interest. The default should be account disabled and user can't change password. Letting the password expire affords some measure of additional protection, as an expired password that can't be changed by the user is logically more secure than one that never expires and can't be changed. Whether this works in practice, I can't say.
>
> *Remote access:* Start>Control Panel>Remote settings. Remote settings are prominently shown.

It goes without saying that changing these registry settings would be exceedingly unwise unless you fully understand the consequences.

Homegroups are a domain of a different color—they're a naive way to link computers through a domain controller but without being part of the Active Directory structure. The major advantage of homegroups is also their major disadvantage: resources (printers, computers, files, and so forth) are shared without many restrictions. Homegroups introduce a vulnerability because they assume complete trust. The principle of least privilege is more sensible.

> *HomeGroup:* Start>Control Panel>Network and Internet>HomeGroup. Check status.

There's also a friendly approach to file sharing called *access privileges* you'd be well served to inspect:

> *Hard Disk Privileges:* Windows Explorer> Right click on drive> Properties> Sharing tab. Look who's sharing. >Security tab. Look who's got full control and modification privileges.

In these cases, Windows has remained fairly open through its entire evolution from XP to 8 and beyond. Microsoft's position is that this openness is a feature that should be enabled by default and that users have the controls to disable each of them if they wish. This implies a caveat emptor outlook.

LOCAL SECURITY POLICIES

Account policies can be viewed by expanding the configuration groups at Start>Control Panel>System and Security>Administrative Tools>Local Security Policies. Microsoft still allows a lot of flexibility here, but it disabled reversible encryption storage by default with XP. Reversible encryption allows an intermediary password filter to store passwords so that some applications—for example, HTTP Digest Authentication and Chap—can recover the plaintext password for authentication. While not as lame as LAN Manager, using such password filters is risky. By 2009, pentester Niels Teusink had developed an automated tool, "revdump," to recover plaintext passwords (blog.teusink.net/2009/08/passwords-stored-using-reversible.html). We can conclude that Microsoft justifiably became uneasy about reversible encryption as it disabled it by default from XP onward.

Account lockout policy allows the user to define the threshold and duration of account lockout after failed login attempts. Not much has changed with Microsoft's configurations in the past 15 years, which suggests that they feel they've nailed this issue.

AUDITING AND USER RIGHTS ASSIGNMENT

Windows has had a strong auditing system since NT. The auditing capability can be enabled/disabled for both success and failure for events such as login, account management, directory service access, object access linked to access control lists, policy changes, backup-and-restores, process tracking, system events, and security events. The auditing framework was robust enough in XP that no configuration changes seem to have been made—a product of the framework's flexibility. Microsoft anticipated that information overload would result from unnecessary audits

(for example, successful logins, successful directory accesses, and process tracking), so it made it possible for users to determine their own comfort zone. If you want to check your settings, expand Audit Policy under Local Policies in the Local Security Policies window described above.

More variation is to be found with user rights assignment (User Rights Assignment in Local Security Policies). This is the mechanism Microsoft provides to control which groups can participate in what services and events: who can log on locally and remotely, change the system time, change schedule priorities and memory quotas, and so forth. There were a couple of noteworthy changes made in Windows 7. First, Windows added a post-XP credential manager that is assigned to the Winlogon service for backup and restore. No accounts should be considered a trusted caller under normal circumstances (although Windows allows the user to add accounts). The same applies to a second control over remote logins. Both of these controls are more nuanced than in XP, though in principle the features have been present since Windows 2000 SP2.

Three related additional configuration changes have to do with which users have permission to create global objects, permanent share objects, and symbolic links. Global objects are shared by all users, so potential conflicts arise if this isn't limited to trusted users. By default, Windows restricts the creation of permanent shared objects to the kernel to avoid conflict. Control over symbolic links was a necessary configuration expansion because of a long-time vulnerability in the file extension handler as far back as 2000 (www.cert.org/historical/incident_notes/IN-2000-07.cfm). This problem loomed so large—it was the vulnerability that enabled the USB memory stick malware injector in later versions of Stuxnet—that it deserves some exposition.[1]

By default, Windows (XP through 8) suppresses file extensions in two ways. For known file types such as .doc, .exe, and .txt it simply suppresses the extension altogether unless the user requests that hidden files, folders, and drives be shown by clicking the Show box in the Folder Options menu (Control Panel>Appearance and Personalization>Folder Options). But even if this feature is enabled, any attendant symbolic links are still suppressed including .lnk, .url, .pif, .scf, .shs, .shb, and .xnk. This created a major vulnerability because Windows' icon handler incorrectly parsed shortcuts. The problem is that suppressed .lnk and .shb extensions can point to an executable, and a suppressed .shs file can include executable code. The icon handler should have checked for this but didn't. When the user opened a file like <harmless> or <harmless.txt> with a suppressed .scb, the icon handler would ignore the extension and load the executable at the end of the link, <harmless.txt.scb> (www.askvg.com/tip-how-to-show-file-extensions-of-shortcuts-lnk-url-pif-in-windows-explorer). Stuxnet's .lnk injector exploited this design flaw in the icon handler and Windows Shell, and it became the primary hack in later versions of the malware (www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-2568)

These problems led Microsoft to change the way it handled symbolic links. By default, Windows 7 reserves permissions to create symbolic links to the administrator (technet.microsoft.com/en-us/itpro/windows/keep-secure/user-rights-assignment). Control over client impersonation after authentication was added with XP SP2 and retained.

A post-XP control dictates who can increase process working sets of memory pages. The default in Windows 7 is users, but the control can be elevated to the administrator for safety. Increasing working sets consumes CPU resources, so restricting to the highest privilege level would diminish the threat levels of CPU hog-type hacks.

SECURITY OPTIONS

Microsoft has significantly expanded the number of security configuration options, from 56 in XP to 90 in Windows 7—a nearly 60 percent increase.

Vista introduced a force audit policy to override audit policy category settings that are set in Group Policy by the domain controller. So, if a more rigorous audit policy is required on some computers, that can be changed without affecting Group Policy. This is a useful feature for suspect computers with anomalous behavior. A few user-interface security features introduced after XP include suppression of the last login username and a mechanism to provide a logon message text/title that doesn't require the domain controller.

More refined control has been added over remote access of registry paths and subpaths. Although paths were controllable in XP, Vista added subpaths. In this way, you can allow network access to System\CurrentControlSet\Control\Terminal Server while denying access to System\CurrentControlSet\Control\ContentIndex regardless of the specifications in the access control list.

Windows 7 finally provided the means to kill off the LAN Manager (LM) password-hashing protocol with two settings that break free from legacy Windows OSs like 9x, ME, NT, and OS/2. The first setting allows the local system to demand compatibility with the newer NTLM hash that uses long passwords and salt (LM truncated to 14 characters without salt, which in Microsoft's own words was "relatively weak and prone to attack"—an understatement). LM actually separated the already weak 14-character string into two 7-character strings before hashing, thereby reducing one simple decryption problem into two trivial ones. By the early 2000s, brute-force cracking of alphanumeric LM passwords was routinely accomplished on a notebook in a few hours or less with L0phtcrack and its descendants, LC3 and LC4. The default setting for Vista

<ALT>-FAQs

For those of you who've followed the Vault 7 leak of CIA documents released by Wikileaks, it might feel like déjà vu all over again. Once again we learn that the US government has developed technology to circumvent Fourth Amendment constitutional safeguards against surveillance of US citizens without benefit of court order. It should be remembered that this was specifically made illegal in the 1970s and why Congress created the Foreign Intelligence Surveillance Act (FISA) court in 1978. The FISA court was given permission to approve warrants in near real time with completely ex parte deliberations—targets and defense lawyers were excluded by law from participating in its deliberations.1 Of course, breaches of constitutional safeguards are nothing new with the three-letter agencies. The FBI and CIA have both engaged in this activity since their inception in 1908 and 1947, respectively. Put that in your "weeping angel" and surveil it!

So now we have another 8,761 pages of low-level classified documents from 2013 to 2016 from the CIA's extra double secret super-private archive. This should give us all pause on whether our surveillance state is actually working for or against us. As I've said before, the prevailing view of people who know how the surveillance state works but don't derive personal gain from it seems to be that Deep State tradecraft is of questionable value, and the veil of secrecy it hides behind is used primarily to cover up criminal activity, malfeasance, and incompetence.[2]

On the bright side, the recent Wikileaks exposé drew attention to two important problems. First, bug bounty programs aren't working because they're in direct competition with governments. Even though the bounties have increased this year (by 50–100 percent by some estimates[3]) they're not even close to being as remunerative as the malware gray market[4] that caters to nation-state intelligence agencies. Second, Wikileaks founder Julian Assange has promised to release the leaked source code to the tech companies so that they can patch their systems. Given that the bug bounty program isn't working, anything that leads to more secure tech products is a good thing.

As for the rest of the Vault 7 revelations—CIA development of vehicle telematics hacks, software to turn mobile devices into real-time surveillance platforms without users' knowledge, compromises of encryption systems, spying on our allies, and so on—it's much ado about nothing, so far at least.

References

1. H. Berghel, "Moral Hazards, Negative Externalities, and the Surveillance Economy," *Computer*, vol. 47, no. 2, 2014, pp. 11–15.
2. H. Berghel, "Secretocracy," *Computer*, vol. 49, no. 2, 2016, pp. 63–67.
3. W. Wei, "Google Increases Bug Bounty Payouts by 50 percent and Microsoft Just Doubles It!," *The Hacker News*, 3 Mar. 2017; thehackernews.com/2017/03/google-bug-bounty.html.
4. C. Miller, *The Legitimate Vulnerability Market: Inside the Secretive World of 0-day Exploit Sales*, internal report, Independent Security Evaluators, 6 May 2007; www.econinfosec.org/archive/weis2007/papers/29.pdf.

and beyond is to disable the storage of LM hash values, thus permanently closing that attack vector. The second setting is the LM authentication level that allows the user to restrict challenge/response authentication to some version of NTLM when supported. The minimum standard for Windows 7 and Server 2008 R2 is NTLMv2 authentication with 128-bit encryption—a major break with XP and Vista. For reasons of backward compatibility, several NTLM restrictions regarding auditing, authentication, and communication with remote computers are undefined (not used by default). With these settings, Microsoft now leaves the responsibility for breakage to users.

Microsoft's commitment to secure login and communication became noteworthy with Windows 7. Several new features appeared including rules for requiring password authentication for key management together with Federal Information Processing Standard (FIPS)-compliant algorithms (carried over from XP); elevated permissions requirements for system objects for reading, modifying, and starting shared objects; the required processing of certificate rules prior to software execution; and several new user account controls including access shutoff and gradations of consent and credentials for elevation of privileges. Taken together, 10 new user account controls show Microsoft's increased interest in beefing up access control.

As an aside, Microsoft introduced the undefined default when it allowed local administrator control over features without taking a position on an optimal default value. Undefined configuration features are disabled by default. Examples include restricting access to tertiary storage (remote media, floppy and backup drives) to local (versus network) access, password age control, whether server administrators may schedule tasks, and

whether domain controllers may refuse local password change requests. There are optimal default values and best practices, to be sure, but they're infrastructure-dependent based on considerations like required support of legacy Windows systems, compatibility with other networked computers, whether it's reasonable to assume a Windows-only environment, and so forth. The addition of these undefined features indicates Microsoft's awareness of particular vulnerabilities.

TRENDS

Several clear trends emerged from my ad hoc analysis of Windows configuration changes from XP through and beyond.

With Vista, Microsoft sought to plug remote login accounts like HelpAssistant and Support_xxxxxx by shipping the OS with access disabled. This was a compromise between the preferable alternative of disabling the support altogether and retaining legacy compatibility. Similarly, homegroups and drive sharing are more tightly controlled. This is a predictable response to the hacking vectors discovered with earlier NT-based OSs, and isn't unrelated to the major vulnerability produced by the default support of null sessions in early NT systems that allowed enumeration of user accounts and properties without authentication. Indeed, for many years the SANS Institute routinely used hacking into NT systems through anonymous enumeration of Security Account Manager (SAM) accounts and shares as classroom examples of simple hacking techniques. With XP, Microsoft provided controls for both anonymous accounts (default = not allowed) and anonymous shares (default = allowed) and unauthenticated users (also known as everyone; default = allowed). If you want maximum protection from unauthenticated access, all features should be disabled or restricted. Local settings can be found in the registry hive at HKLM\System\CurrentControlSet\Control\lsa under the keys restrictanonymoussam, restrictanonymous, and everyoneincludesanonymous. Of course, the stronger protection of registry key value = 1 will interfere with interprocess communication through the Server Message Block protocol. In general, Microsoft has plugged the null session hole in Windows from XP on.

A casual study of the evolution of Microsoft's local security policies reveals a patch of another vulnerability via reversible encryption that's a technological sibling to the ill-conceived 14-character, case-insensitive, salt-free LM password hash regimen. From an insecurity perspective, both were mistakes carried forth to perfection by Microsoft. The first was resolved in XP and the latter in Vista.

Auditing was always a strength of NT-based systems, so not much has changed with time except the occasional feature that allows the user to require more auditing than demanded by the domain controller. User rights assignment is a different matter. Credential management and trust levels received greater attention, following best practices. The Windows 7 treatment of symbolic links is noteworthy given the Stuxnet experience. Symbolic links and Autorun (incidentally, used in the earliest versions of Stuxnet) were both poorly handled by Microsoft to the peril of their users, including the uranium enrichment facility in Natanz, Iran.

Taken together, these trends are both reassuring and alarming: reassuring because discovered vulnerabilities were patched, but alarming because Microsoft seems to be reactive rather than proactive in patching holes. After disclosure of a major vulnerability—for example, the Autorun and .lnk hacks used in different versions of the Stuxnet injector—Microsoft responds appropriately. But it's unclear how much attention they pay to hacks that don't receive widespread media attention. A brief review of the literature indicates that Microsoft tends to delay patches longer than necessary. Of course, these vulnerabilities are likely due to product design issues rather than poor-quality code, as each of the weaknesses discussed above were actually negative externalities of highly touted Windows "features" (Autorun, null sessions, and so on).

Although narrow in scope, I found my quick longitudinal analysis to be a good indicator of how well one major software developer performed over time. Similar studies about other developers could likewise make valuable contributions to the literature. While such limited analyses aren't very useful in defining security policies, the trends they reveal can be insightful in risk assessment. A second pass might plot such security changes on a timeline along with reported vulnerabilities—the correlations might prove illuminating. But that project is for another time.

If this topic interests you, I recommend consulting David Karp's expansive guides[2–4] for further detail on recent Windows OSs.

REFERENCES

1. H. Berghel, "A Farewell to Air Gaps, Part I," *Computer*, vol. 48, no. 6, 2015, pp. 64–68.
2. D.A. Karp and T. O'Reilly, *Windows XP in a Nutshell*, 2nd ed., O'Reilly Media, 2005.
3. D.A. Karp, *Windows Vista Annoyances: Tips, Secrets and Solutions*, O'Reilly Media, 2008.
4. D.A. Karp, *Windows 7 Annoyances: Tools and Techniques to Improve Your Windows 7 Experience*, O'Reilly Media, 2010.

HAL BERGHEL is an IEEE and ACM Fellow and a professor of computer science at the University of Nevada, Las Vegas. Contact him at hlb@computer.org.