



August 2008

Introducing Privilege Guard – A Practical Solution to Applying the Principle of Least Privilege

Mark Austin

This paper outlines the principle of least privilege, the benefits of following this practice and the barriers to implementation. It then goes on to suggest approaches to adopting a least privilege approach and finally introduces Privilege Guard, a solution that enables privileges to be assigned to individual applications, providing a practical implementation of least privilege for corporate environments.

The information contained in this document ("the Material") is believed to be accurate at the time of printing, but no representation or warranty is given (express or implied) as to its accuracy, completeness or correctness. Avecto Ltd, its associated companies and the publisher accept no liability whatsoever for any direct, indirect or consequential loss or damage arising in any way from any use of or reliance placed on this Material for any purpose.

Copyright in the whole and every part of this document belongs to Avecto Ltd ("the Owner") and may not be used, sold, transferred, copied or reproduced in whole or in part in any manner or form or in or on any media to any person other than in accordance with the terms of the Owner's Agreement or otherwise without the prior written consent of the Owner.

Contents

The Principle of Least Privilege	3
Definition of Least Privilege	3
Benefits of Least Privilege	3
Barriers to Implementing Least Privilege	4
Implementing Least Privilege	5
Removing Administrative Rights	5
Windows XP and 'Run As'	5
Windows Vista and User Account Control	6
Introducing Privilege Guard	7
Conclusion	8
About Avecto	8
References	8

The Principle of Least Privilege

Definition of Least Privilege

The principle of least privilege requires that a user should be given no more privileges than is required to perform their job function.

The Department of Defense Trusted Computer System Evaluation Criteria, (DOD-5200.28-STD), also known as the Orange Book, defines least privilege as a principle that “requires that each subject in a system be granted the most restrictive set of privileges (or lowest clearance) needed for the performance of authorized tasks. The application of this principle limits the damage that can result from accident, error, or unauthorized use.” [1]

The principle of least privilege was first put forward as a design principle by Jerry Saltzer and Mike Schroeder over 30 years ago. According to Saltzer and Schroeder, “Every program and every user of the system should operate using the least set of privileges necessary to complete the job. Primarily, this principle limits the damage that can result from an accident or error. It also reduces the number of potential interactions among privileged programs to the minimum for correct operation, so that unintentional, unwanted, or improper uses of privilege are less likely to occur. Thus, if a question arises related to misuse of a privilege, the number of programs that must be audited is minimized. Put another way, if a mechanism can provide ‘firewalls’, the principle of least privilege provides a rationale for where to install the firewalls. The military security rule of ‘need-to-know’ is an example of this principle.”[2]

The precise definition of least privilege, not only deals with users running with minimal privileges, but also processes. In that, a process should only have the rights necessary to perform its function, and that an access right or privilege should only be held while it is required. In practice, most operating systems tend not to have the granularity of control that is required to implement least privilege precisely, but this should not deter organizations from adopting a more practical implementation.

Benefits of Least Privilege

The benefits of least privilege can't be ignored and yet one of the most common problems in organizations is the proliferation of administrative rights on systems. Although the threat associated with this practice is clearly understood, there is an assumption that it will be too difficult to remove administrative rights from users.

Users who logon to their systems with administrative rights are at much greater risk from malware. Malicious software is more effective when its payload runs with administrative rights, as it does not need to find a security flaw to gain privileged access to the system. An administrator has full control over the system, so malware can use these privileges to install drivers, intercept logon passwords, create user accounts, install root kits, replace system files and disable security software. In fact many viruses fail to infect a computer when a user has standard rights, and there is strong evidence that running users with minimal rights greatly reduces the risk of virus infection.

Malicious software is not the only reason to adopt the principle of least privilege. A user with excessive privileges can either deliberately or accidentally cause system configuration problems,

resulting in downtime and increased desktop support. Users with administrative rights are also free to install software, which leads to unlicensed software installed on corporate systems and system stability problems.

Most corporate environments strive to deploy a standard desktop, which is locked down in such a way that the user can still perform their job function. This effort becomes pointless if users run with administrative rights, as they have the privileges necessary to alter the desktop configuration. If an organization is to succeed in an initiative to deploy locked down desktops then they must remove users from the administrators group.

Although least privilege is not limited to removing administrative rights from users, this is a crucial first step, and one that has the biggest security benefit to the majority of organizations. Following least privilege to a greater degree can increase system security further. For instance, data access can be restricted at the application level, and not just at the user level. Although this may not provide the obvious benefits of removing a user from the administrator group, every step that can be taken to restrict access to resources, can only decrease the threat of a security incident. Although in practice, very few operating systems are capable of restricting access at the application level, with access rights typically being assigned to the user at logon and all applications inheriting these rights.

Barriers to Implementing Least Privilege

Given the increase in security associated with removing administrative rights, it could be asked why so many organizations run a large proportion of their user base with administrative rights. It all comes down to practicality, in that there is an assumption that it will be too difficult and that users will become less efficient due to the restrictions of running with minimal rights.

If the user needs administrative rights to run even a single application then they need to be made a member of the local administrators group, and in reality it is often more than one application that requires these rights. The increase in laptop deployments has added to this problem, because users are more likely to carry out basic administration tasks, such as changing network settings, adding printers and installing authorized software.

The inability of most operating systems to provide the granular control necessary to restrict administrative rights to particular applications is the fundamental barrier to most organizations adopting a least privilege approach, as a user must be granted the rights necessary to perform all of their tasks.

Although Windows XP and Windows Vista both have features that enable a user to run without administrative rights, namely 'Run As' and User Account Control (UAC) respectively, both of these features still require the user to have access to an administrator account to perform administrative tasks. Although a welcome addition, this limitation tends to make these built-in features more appropriate for real system administrators, as it enables them to logon as a standard user and only use the administrator account when they need to perform administrative tasks. The benefits and limitations of these operating system capabilities will be covered in more detail later in this paper.

Implementing Least Privilege

Removing Administrative Rights

Removing administrative rights for standard users is the first step to adopting the principle of least privilege. This is often referred to as the least-privileged user account (LUA) approach and can be applied to both standard users and administrators.

However, the LUA approach can be difficult and costly to implement, as applications that require administrative rights to run must either be redeveloped or their activity must be closely monitored in order to relax the permissions on resources, allowing the application to run with standard rights. Unfortunately, this tends to be a complex undertaking, and the end result is to create a different set of security concerns due to the weakening of access control lists on resources.

Where users require access to applications that must have administrative rights, such as performing system configuration tasks, then the only solution available with most operating systems is to provide the user with access to an administrative account. Windows XP 'Run As' and Windows Vista User Account Control can be used to run individual applications with administrative rights, but the user must provide the credentials for a local administrator with both of these approaches. This obviously creates a number of concerns:

- The user has access to the local administrator account and must therefore be trusted not to abuse these privileges, a situation that is less than ideal in a corporate environment.
- Applications running with administrative rights are now running under a separate user account, which creates its own set of problems. For instance, these applications will not have access to the user's profile or network shares, which can result in a less than seamless experience for the user.
- The user must remember two passwords, one for their standard account and a second for the administrator account. Encouraging users to be security conscious with a single account is challenging enough, so introducing a second, more privileged account, is very likely to lead to security issues at some point.

However, even with these problems, making use of these features is much better than allowing users to logon with an administrative account to perform all of their tasks, and so we will take a closer look at how each of these features function before exploring the Privilege Guard solution.

Windows XP and 'Run As'

The Windows XP 'Run As' feature allows a user to run applications with a different user account inside the same session. This is commonly used to run applications with the local administrators account, and is an obvious improvement over the user logging off and then logging on as an administrator to run applications that require administrative rights. In practice, 'Run As' tends to be used by real system administrators, who are security conscious. They login with a standard user account to perform day to day tasks, such as reading email and browsing the web, and only access the local administrators account when performing administrative tasks.

Although 'Run As' could be used to implement a basic least privilege environment, the need for standard users to have access to the administrators account, presents a less than seamless experience to the user and still leaves the user free to use these rights at will.

Windows Vista and User Account Control

Microsoft took a much bolder approach in Windows Vista with the introduction of User Account Control (UAC). Although UAC has been met with a mixed reaction, it is a welcome addition to the security features within the operating system.

On Windows Vista when a user logs on with administrative rights, in addition creating an access token corresponding to this privileged account, a filtered token is also created. The filtered token has the administrative rights removed, and is only granted the privileges of a standard user. Processes launched by the user are assigned the filtered token. Applications that require administrative rights can be marked by the developer as requiring elevation, and many of the system configuration applications in Windows Vista are marked as requiring elevation. When an application requires elevation the user is prompted with a consent dialog. If the user approves the elevation then the application runs with the unfiltered token, which includes their full administrative rights.

UAC also has some rules that it applies to determine whether an application is likely to require elevation even though it may not be marked as requiring elevation. This is aimed at trying to handle applications that were developed before Vista that are not UAC aware. If all else fails the user can run any application with the unfiltered token by using the shell context menu and selecting the 'Run as administrator' option.

It should be noted that elevation can only occur when a process starts, so once an application is running with a filtered token it is not possible for the process to ask for elevation during execution. A prime example of this limitation, which is imposed by the operating system, is task manager. By default task manager will run with the filtered token, but if the user clicks the 'Show processes for all users' button then task manager will prompt for elevation. Task manager achieves this by spawning a new task manager process, which asks the user for consent to elevate. If the user agrees to the consent dialog then the original task manager will exit with the elevated task manager taking over.

Standard users are treated differently on Vista, in that they only have a single access token when they logon, as they do not have administrative rights. When an application requires elevation the user is prompted, but this time they will be asked to enter the credentials of an administrative account. Assuming they have access to an administrative account then the user can enter the credentials and the application will launch with an access token created with these privileged credentials. The end result is not dissimilar to 'Run As' on Windows XP, where the processes running with administrative rights are running under a different account.

Although UAC is a significant step forward it still suffers from the same issue as Windows XP and 'Run As'. Users must either be a member of the local administrators group or have access to an administrative account in order to run applications that require administrative rights. Once a user has these rights or access to an account with these rights they may freely use these privileges.

UAC also suffers from another problem, which is why it has received mixed reactions in corporate environments. Although the consent dialogs are there for good reason, to ensure that a user does

not inadvertently launch an application with elevated privileges, these dialogs can become intrusive. For standard users, who do not have access to an administrators account, they can also be confusing.

As with 'Run As' the most appropriate use of UAC is for real administrators. The major differences are that 'Run As' was optional to administrators, whereas UAC forces administrators to run with standard rights, which has to be a positive stance on security. With UAC an administrator does not require two accounts, so although the experience may be slightly intrusive, the applications run under the same user context, albeit with a different access token.

Where both Windows XP and Windows Vista fall short is in the handling of standard users in corporate environments. Giving standard users access to an administrative account is not secure and the experience is not seamless to the user. The user has to manage both accounts, with applications running under the context of two distinct user accounts, which can also lead to operational issues.

Introducing Privilege Guard

The corporate environment requires a solution that can elevate individual applications based on policies defined by the IT department, and not determined by the user or the operating system. A solution that allows users to logon with standard accounts, and one that does not require users to have access to an administrative account. Avecto Privilege Guard was designed with these goals in mind.

Privilege Guard allows organizations to adopt the principle of least privilege by enabling users to run with standard rights and elevate individual applications based on policy settings. The experience is totally seamless to the end user, and all applications run under the context of a single user account.

Unlike UAC, Privilege Guard also creates multiple tokens for standard users. When a standard user logs on, Privilege Guard will create an elevated token for that user, which will be assigned to applications that have been deemed to require administrative rights. These applications are defined in the Privilege Guard policy, which ensures that all decisions to run applications with elevated rights are under the control of the IT department.

Importantly, users do not have to belong to the administrators group or have access to an administrative account. The Privilege Guard client does not require access to an administrative account either, as the elevated token is based on the user's own token, which ensures that all applications are running under a single user account, avoiding the operational issues that can arise from dual accounts.

Application rules are extremely flexible, with Privilege Guard providing support beyond executable images. Application policies can be defined for the following types of file:

- Executable images (.exe)
- Control panel applets (.cpl)
- Individual management consoles (.msc)
- Windows installer packages (.msi)
- Windows scripting host scripts (.vbs, .js)
- PowerShell scripts (.ps1)

- Batch files (.bat, .cmd)
- Registry settings (.reg)

A variety of application identification rules are also available to ensure the correct applications are elevated. Applications may be matched on any combination of file path, command line, hash (SHA-1) and certificate.

Shell integration provides the ability for users to launch elevated applications on demand via a shell context menu option. This menu option can be restricted, in that it can be limited to a set of applications, as opposed to being available for all applications.

An important aspect of all security products is auditing, and Privilege Guard can audit all or selective application elevations, giving details of how the application was invoked and the policy that applied.

Conclusion

This paper has introduced the principle of least privilege and the benefits of implementing this approach in a corporate environment. The most important step for most organizations is to remove administrative rights for standard users. Although developments in the Windows operating system have made it possible for users to run with minimal rights, the need to logon with administrative credentials to run applications that require administrative rights, introduces its own set of security challenges.

Avecto Privilege Guard was designed to meet the requirements of implementing least privilege in corporate environments. By enabling users to run with minimal rights and elevating individual applications, without giving users access to an administrative account, Privilege Guard provides a practical solution to the issues that prevent most organizations from adopting the least privilege approach.

About Avecto

Avecto is a pioneer in least privilege technology, helping organizations to deploy secure and compliant desktops. With its innovative new product, Privilege Guard, it is no longer necessary to assign admin rights to users, as these rights can now be assigned dynamically to individual applications, tasks and scripts. Privilege Guard enables users to log on with minimal rights and empowers them to perform their day to day role, without compromising the integrity and security of the standard desktop build. Avecto was founded in 2008, and is headquartered in Manchester, England.

References

[1] Department of Defense Trusted Computer System Evaluation Criteria (Orange Book) at <http://zedz.net/rainbow/5200.28-STD.html>.

[2] SALTZER, J.H. AND SCHROEDER, M.D., 'The Protection of information in computer systems,' *Proceedings of the IEEE*, vol. 63, no. 9 (Sept 1975), pp. 1278-1308.