



Access Control Excellence

A Practical Approach to Controlling Privileged Accounts

Organizations must prove that privileged and root operations are suitably controlled, but delegating and managing privileges is not straightforward. Sharing privileged account passwords is a security and compliance no-no, while more complex methods, such as dual controls, can hinder business processes. To effectively and efficiently control privileged accounts, a combination of different access management components is required.





“Administrator accounts have privileges to access any data and execute any application or transaction, typically with little or no tracking or control. These accounts — which in some enterprises number in the hundreds — are frequently not tied to specific individuals, so the accounts can be used to do virtually anything, with little or no possibility of detection.”

Gartner - The Top 10 Risk and Security Audit Findings to Avoid

Containment of Privileged and Root Accounts Is No Longer Optional

Properly defining, controlling and monitoring administrative privileges in IT systems are real challenges for enterprises. And while in the past, controlling privileged accounts made good business sense, today, it is mandated by regulations such as Sarbanes-Oxley (SOX) Section 404.

In addition to the increased potential for failing IT security audits, a lack of proper control over root and privileged accounts can lead to a significant increase in the risk of fraudulent activities by employees.

What’s the Problem?

All computer operating systems require some kind of unrestricted administrative access to enable system management. Security models based on group policies and privileges, overlaid and accumulated through group membership and security principals, as is the case in Microsoft Windows environments, generates one set of challenges. On UNIX and Linux systems, the unrestricted “root” account poses a different and particularly troublesome situation. And similar super user issues can also arise with privileged accounts for database administrators.

While the various security models face similar problems and risks associated with super users and shared passwords, this white paper will primarily focus on the UNIX and Linux specific issues.

The real challenge is that the problems associated with controlling privileged and root accounts increases quickly as the number of people who need powerful administrative access for various job functions grows. Examples of roles and tasks that may require privileged accounts include:

- **System administrators** - Upgrading operating system software
- **Security administrators** - Managing access rules, user IDs, passwords

- **System operators** - Mounting volumes and tapes
- **Database administrators** - Implementing database access controls
- **Network managers** - Configuring network connections
- **Application developers** - Installing and running compilers
- **Consultants** - From hardware and software vendors
- **Help desk and support** - Resolving user-related issues

Considering all of the work that needs to be done using privileged permissions, most organizations find that controlling delegation of these accounts can quickly become difficult to manage and monitor. Furthermore, granted permissions are rarely reviewed or revoked, which means that users unintentionally accumulate more and more privileges over time as their job functions change and new access rights are granted.

Native UNIX and Linux systems usually have poor support for controlling delegation of privileged accounts. The easy way out is to share root passwords with a variety of administrative “controls”:

- Write the password on a piece of paper
- Create multiple user accounts with different passwords that share the root UID 0
- Create emergency superuser accounts, that are protected with dual controls, by splitting passwords in half so two people need to communicate prior to use
- Implement a software-based solution that generates random passwords for emergency superuser accounts

Another common practice is delegation through `suid` or `sgid`, allowing a process to run in the context of the program file's owner or group. However, delegation through `suid` or `sgid` is also a much appreciated door-opener for hackers, since such processes can be readily exploited.



In order to provide accountability that can be traced back to individual users, you should minimize the use of functional accounts in your server environment. Strive to provision each physical user with a universal account that can provide the access that individual needs to various resources across the enterprise. This provides segregation of duties and traceability.

Regardless of which of the above approaches is being used, all fail to provide efficient, effective protection of privileged accounts. Even an elaborate process for password check-out and subsequent password resets can remain dependent on someone with unlimited privileges.

And there are some other considerations to the challenge of controlling privileged accounts:

- **Securing data in transit:** Even if a user has been granted administrative access in a controlled fashion, used in a clear-text telnet session, the password can be intercepted and shared in an uncontrolled way by someone listening in on network traffic. And asking your users to switch to a peer-to-peer solution, such as user-managed SSH, only partially improves the situation; it doesn't scale, you remain vulnerable to man-in-the middle attacks, and you have lost control to individuals who may want to make use of port-forwarding techniques for purposes that are not aligned with your business objectives. The FoxT framework can be easily extended to protect desktops and applications in the enterprise.
- **Passing IT audits:** Even if you trust your process for password sharing, you still won't be able to satisfy your auditors. When the audit log shows that "root" did something, your auditor will want to know the real name behind "root".
- **Capturing roots' actions:** Once root privileges have been acquired, the omnipotent user can do just about anything, including deleting log fields and then re-configuring the system for future private use. You need a way to effectively monitor and record what the root user is doing once they have acquired their privileges.

What's Really Required to Protect Privileged Accounts?

To effectively and efficiently control privileged accounts, a combination of different access management components is required:

- **Centralized management of user accounts on all servers:** Ensures that you can monitor and audit which user has what type of access

on which machine. Centralized management will also facilitate faster disabling of user accounts across the security domain.

- **Enforcement of policies for delegation of privileges:** Instead of allowing functional accounts such as “root” or “sysdba” to login, you need to have enforceable access rules that mandate the use of individual and auditable user accounts. Using access controls, any switch to a privileged functional account for specific job functions or tasks will be tied to the named user. Using centralized management of fine-grained access rules that can be enforced throughout the security domain means that controlled switching to a privileged account will not require password sharing.
- **Secure communications that are centrally managed and explicitly mandated by access rules:** It is not sufficient to provide users with the option to use a secure connection rather than clear-text telnet. You must be able to enforce its usage where needed. As well, it is not sufficient to delegate the task to establish encrypted connections to individual users (e.g. letting them maintain user and host public SSH keys as they find suitable). In reality, delegating the task introduces a new authentication authority which in turn subverts centralized management and enforcement of access rules.
- **Centralized audit logging:** An effective approach to protecting privileged accounts includes centralized audit logging with a detailed record of user activities. It is also important that these logs are kept on a separate security domain so they can be trusted.
- **Keystroke logging:** For sensitive sessions, you must also have the ability to enforce full keystroke logging so administrator activities can be tracked in detail.
- **Fine-grained access controls to specific machines:** A user who needs privileges on machine A should not automatically gain the same type of access on machine B, just because that is how password conventions and configurations were once made.

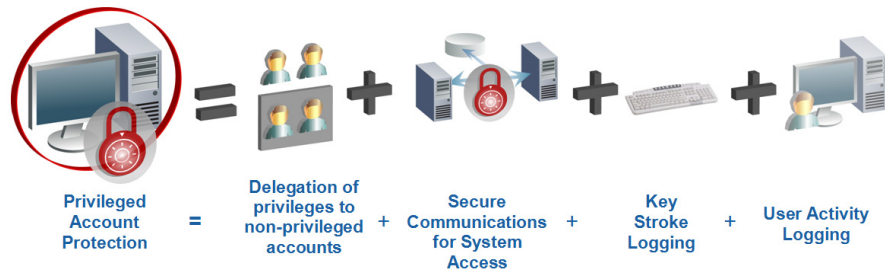


What Options Are Out There?

Organizations struggling to achieve effective protection of privileged and root accounts often evaluate three different alternatives:

1. **Create a home-grown solution** based on operating system capabilities, available utilities such as “sudo”, clever password management procedures, and lots of scripts. Except for in very small organizations, the attempt to create a home-grown solution will often become extremely costly, as well as requiring system administrators to do programming instead of their jobs. The home-grown solution is often found insufficient from an auditor’s perspective as well. Even if they provide an acceptable level of password management capabilities, they often fail to fully address auditing requirements.
2. **Combine various commercial or open source point solutions** to create an operating system environment that provides an effective approach to protecting privileged accounts. This typically involves using one solution for user provisioning, another for centrally managed secure communications (SSH), a third for password management, and possibly another tool for audit log consolidation. While the combined solutions can amount to something powerful, in the end, one important aspect is lost: centralized management on one security system. Combining multiple technical solutions into one leaves conceptual gaps, which in turn lead to security flaws and inefficient management. Ironically, while cost-awareness may well be the primary driver for exploring this option, it typically ends up costing an organization more than other options.
3. **Invest in an Enterprise Access Management solution (EAM).** Several of the leading EAM solutions do provide several of the components required to control super user privileges, but these systems are very expensive and complex to install and operate and still do not fully address the capabilities needed to effectively manage privileged accounts.

There is another option. FoxT ServerControl provides all of the components needed for effectively protecting privileged and root accounts, without all of the overhead, costs, and complexities of full-blown EAM infrastructures.



Protected Delegation of Privileged Accounts

FoxT ServerControl enables you to centralize access controls and authentication for UNIX, Linux, Windows and Virtual servers. Entire domains of servers running heterogeneous operating systems can be securely managed from one central, web-based administration console.

In addition to providing fine-grained access rules to manage which users can access what network or local service on which server, when, and from where, FoxT ServerControl also delivers a unique and powerful combination of functionality for effectively controlling privileged and root accounts.

- First, many of the common management functions in your server domain can be performed using the secure FoxT administration interface via a Web browser. This greatly reduces the amount of operations requiring administrators to know privileged passwords. You can also define sub-administrators who are allowed to work on certain tasks or on specific portions of your server environment and track their actions.



- On FoxT-protected servers, SU can only be performed if the user has a specific access route allowing him or her to do so. Even if a user knows a privileged account password, they cannot use it to become the privileged user unless they have been granted permission to run SU.
- FoxT ServerControl includes programs that enable your administrators to delegate privileged command execution. SUEXEC (for UNIX and Linux servers) and FoxT RUNAS (for Windows servers) allow users to perform specific operations as another user. Again, users can only perform SUEXEC and FoxT RUNAS controlled operations if they have explicitly been assigned permissions to do so. For ease of management, you can specify, down to program argument level, exactly what operation the user can carry out as another user, and group the program command permissions.
- You can also configure the system so that users can run SU and SUEXEC using their own passwords or tokens, which removes the need for any of your administrators to know privileged account passwords.
- If you decide to keep using privileged account passwords, FoxT Password Vault, an optional add-on module, is a password vault that manages the checkout of privileged account passwords and automatically changes passwords after the configurable checkout period has ended. FoxT Password Vault removes the need to share passwords, enforces limits of which users can check out which passwords, and helps you avoid having privileged account passwords active in the system too long.
- SUEXEC operations can be keystroke logged, with a configurable level of keyboard input and on-screen output recorded for reference. Keystroke logging provides a forensic level of traceability, and ensures that no user in your organization can perform any subversive activity in the guise of a privileged user.
- And finally, FoxT ServerControl also enforces use of encrypted communications to protect privileged account passwords.

Organizations using FoxT ServerControl can avoid sharing privileged account passwords, and indeed, once the system is configured, there is no need to use these passwords in day-to-day activities. When operations are traceable back to a specific, physical user, organizations can significantly improve the security over their IT assets and greatly simplify their IT audits.

Copyright © 2010 FoxT. All rights reserved.

The document is provided for informational purposes only and the contents herein are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. The document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior permission.

FoxT logo is a trademark of FoxT, Inc. Other product and company names herein may be registered trademarks and trademarks of their respective owners.

